ANALISIS KARAKTERISTIK *INPUT-OUTPUT* DAN OPTIMASI BIAYA PEMBANGKITAN MENGGUNAKAN METODE *QUADRATIC LEAST SQUARE REGRESSION* DAN METODE *DYNAMIC GENETIC ALGORITHM*

SKRIPSI

oleh

Rina Anggraeni
NIM 121910201110

PROGRAM STUDI STRATA 1 TEKNIK ELEKTRO
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS JEMBER
2016

ANALISIS KARAKTERISTIK *INPUT-OUTPUT* DAN OPTIMASI BIAYA PEMBANGKITAN MENGGUNAKAN METODE *QUADRATIC LEAST SQUARE REGRESSION* DAN METODE *DYNAMIC GENETIC ALGORITHM*

SKRIPSI

diajukan guna melengkapi skripsi dan memenuhi salah satu syarat untuk menyelesaikan Program Studi S1 Teknik Elektro dan mencapai gelar Sarjana Teknik

oleh

Rina Anggraeni
NIM 121910201110

PROGRAM STUDI STRATA 1 TEKNIK ELEKTRO
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS JEMBER
2016

PERSEMBAHAN

Skripsi ini saya persembahkan kepada Alm. Ibu saya tercinta Enik Dwi Rindawati yang telah memberikan kasih sayang, cinta dan seluruh hidupnya untuk saya tanpa pamrih sedikitpun hingga akhir hayat. Skripsi ini juga saya dedikasikan untuk mengenang perjuangan beliau dalam menghadapi penyakit yang diderita serta untuk mengenang seluruh pengorbanan beliau selama hidup dalam membesarkan, merawat, memberikan cinta, kasih sayang serta perngorbanan seluruh jiwa raga kepada saya yang tidak dapat saya balas dengan apapun. Selain itu, skripsi ini juga saya persembahkan kepada :

1. Papa Mistadji Arief Wahyudi yang telah memberikan dukungan materi dan dorongan moral untuk menyelesaikan skripsi ini
2. Kakakku tersayang, Dewi Rifmawati yang selalu memberikan dorongan untuk menjadi pribadi yang lebih baik dan berguna serta menggantikan posisi ibu sebagai tempat berkeluh kesah.
3. Adikku tercinta, Widya Pratiwi yang telah memberikan semangat agar skripsi ini dapat terselesaikan.

MOTO

"*Tuntutlah ilmu walaupun di negeri Cina, karena sesungguhnya menuntut ilmu itu wajib bagi setiap muslim. Sesungguhnya para malaikat meletakkan sayap - sayap mereka kepada para penuntut ilmu karena senang (rela) dengan yang ia tuntut*."

(H.R. Ibnu Abdil Bar)

"What makes you different just might be your greatest strength and the power to change the world has been inside you all along. True courage is pursuing your dream, even when everyone else says it's impossible. Even the smallest person can make a big different and magic will be happens when you believe in yourself. Trust yourself. There is always hope and live your dream."

(Barbie)

"I never dreamed about success, I work for it"

(Estee Lauder)

"*And they lived happily ever after*."

(Disney)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Rina Anggraeni

NIM : 121910201110

Menyatakan dengan sesungguhnya bahwa skripsi yang berjudul Analisis Karakteristik *Input-Output* dan Optimasi Biaya Pembangkitan Menggunakan Metode *Quadratic Least Square Regression* dan Metode *Dynamic Genetic Algorithm*" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan dalam institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak mana pun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 28 Juni 2016

Yang menyatakan,

Rina Anggraeni
NIM 121910201110

SKRIPSI

ANALISIS KARAKTERISTIK *INPUT-OUTPUT* DAN OPTIMASI BIAYA PEMBANGKITAN MENGGUNAKAN METODE *QUADRATIC LEAST SQUARE REGRESSION* DAN METODE *DYNAMIC GENETIC ALGORITHM*

oleh

Rina Anggraeni
NIM 121910201110

Pembimbing

Dosen Pembimbing Utama : Dedy Kurnia Setiawan, S.T., M.T.

Dosen Pembimbing Anggota : Dr. Triwahju Hardianto, S.T., M.T.

PENGESAHAN

Skripsi berjudul Analisis Karakteristik *Input-Output* dan Optimasi Biaya Pembangkitan Menggunakan Metode *Quadratic Least Square Regression* dan Metode *Dynamic Genetic Algorithm*" telah diuji dan disahkan pada:

hari, tanggal : Selasa, 28 Juni 2016

tempat : Fakultas Teknik Universitas Jember.

Tim Penguji:

Pembimbing Utama,	Pembimbing Anggota,
Dedy Kurnia Setiawan, S.T., M.T.	Dr. Triwahju Hardiato, S.T., M.T.
NIP 19800610 200501 1 003	NIP 19700826 199702 1 001
Penguji I,	Penguji II,
Suprihadi Prasetyono, S.T., M.T.	Samsul Bachri M, S.T., M.MT.
NIP 19700404 199601 1 001	NIP 19640317 199802 1 001

Mengesahkan
Dekan Fakultas Teknik,

Dr. Ir. Entin Hidayah, M.UM
NIP 19661215 199503 2 001

Analisis Karakteristik *Input-Output* dan Optimasi Biaya Pembangkitan Menggunakan Metode *Quadratic Least Square Regression* dan Metode *Dynamic Genetic Algorithm*

Rina Anggraeni

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Jember

ABSTRAK

Optimalisasi produksi listrik, khususnya pada pembangkit termal membutuhkan analisis karakteristik *input-output* dan pembebanan yang tepat agar beroperasi dengan baik. Karakteristik *input-output* akan mengawasi pergeseran yang terlihat dari kurva dan mendeteksi perlu adanya *maintenance* atau tidak pada sebuah pembangkit. Karakteristik *input-output* dapat dihitung dengan metode *quadratic least square regression*. Sedangkan pembebanan yang tepat, membuat produksi listrik sesuai maksimal beban yang diinginkan dengan biaya paling murah. Perhitungan pembebanan dilakukan dengan metode *dynamic genetic algorithm*. Metode ini diaplikasikan pada data PT. PJB UP Gresik bulan Juli 2015 didapatkan total biaya bahan bakar yang dihemat sebesar 3.162,9147 KNM^3 dan biaya bahan bakar sebesar $22.773 dibandingkan PJB. Jika diaplikasikan pada data bulan Desember 2012 didapatkan total biaya bahan bakar yang dihemat sebesar 16.532,2189 liter dan biaya bahan bakar sebesar Rp. 1.131.369.852 dibandingkan PJB. Metode *dynamic genetic algorithm* dapat membagi beban sesuai acuan, sementara PSO memiliki selisih -1 hingga 42 MW serta memiliki konsumsi bahan bakar 245,24 liter dan biaya bahan bakar Rp. 1.477.569 lebih hemat dibanding metode PSO. Sementara metode SA memiliki selisih -0,0001 hingga 0,0001 MW. *Dynamic genetic algorithm* memiliki konsumsi bahan bakar 50.511,12 liter dan biaya bahan bakar Rp. 304.329.523 lebih hemat dibanding metode SA.

Kata Kunci: *dynamic genetic algorithm, economic dispatch,* karakteristik *input-output, quadratic least square regression*

Analysis of Input-Output Characteristic and Generation Cost Optimization Using Quadratic Least Square Regression Method and Dynamic Genetic Algorithm Method

Rina Anggraeni

Department of Electrical Engineering, Faculty of Engineering, University of Jember

ABSTRACT

For optimization of electricity production, especially in the thermal power plant required analysis of input-output characteristics and operated load properly. Input-output characteristics will oversee the curve and detected the plant need for maintenance or not. Input-output characteristics can be calculated by least squares quadratic regression method. operated load properly, making electricity production corresponding maximum desired load with lowest cost. Loadi calculations performed by dynamic genetic algorithm method. This method is applied to data from PT. PJB UP Gresik in July 2015 and has saving 3.162,9147 KNM^3 fuel consumption and $22.773 fuel costs compared PJB. While applied to data Desember 2012 has saving 16.532,2189 liters fuel consumption and Rp. 1.131.369.852 fuel costs compared PJB. Method of dynamic genetic algorithm can divide load accordance the reference, while the PSO has a difference of -1 to 42 MW and has 245,24 liter fuel consumption and fuel costs Rp. 1.477.569 more efficient than PSO methode. While the method of SA has a difference of -0.0001 to 0.0001 MW. Dynamic genetic algorithm has 50.511,12 liters of fuel consumption and fuel costs Rp. 304.329.523 more efficient than mtode SA.

***Keywords**: dynamic genetic algorithm, economic dispatch, input-output characteristics, quadratic least square regression*

RINGKASAN

Analisis Karakteristik *Input-Output* dan Optimasi Biaya Pembangkitan Menggunakan Metode *Quadratic Least Square Regression* dan Metode *Dynamic Genetic Algorithm*; Rina Anggraeni, 121910201110; 2016: 99 halaman; Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.

Indonesia merupakan negara kepulauan yang kebanyakan memanfaatkan pembangkit listrik termal sebagai sumber penghasil listrik. Tidak dapat dipungkiri lagi, penggunaan energi listrik saat ini telah meningkat dari tahun–tahun sebelumnya. Total Terra Watt Hour (TWh) yang terjual pada Januari 2016 ialah sebesar 17,57 TWh, lebih tinggi dari Januari 2015 yang penjualannya hanya sekitar 16,34 TWh. Apabila dirupiahkan maka nilai penjualan pada Januari 2016 mencapai 17,6 triliun rupiah, lebih tinggi dari Januari 2015 lalu yang hanya 16,8 triliun rupiah (Benny Marbun, 2016).

Dengan kebutuhan listrik yang semakin meninggi, pemerintah mengoperasikan seluruh pembangkit yang dimiliki, baik pembangkit termal maupun non-termal. Namun saat ini, Indonesia masih lebih banyak menggunakan pembangkit termal untuk memenuhi kebutuhan listrik masyarakat. Seiring pemakaian pembangkit termal secara terus menerus dalam pembangkitan energi listrik, mengakibatkan performa pembangkit mengalami perubahan. Perubahan ini dapat mengakibatkan efisiensi dari pembangkit juga ikut berubah. Perubahan efisiensi ini sangat diperlukan untuk diawasi agar daya yang dihasilkan tetap dapat memenuhi kebutuhan. Untuk mengetahui perubahan performa suatu pembangkit ini, maka digunakanlah kurva karakteristik *input–output* yang dapat dihitung dengan menggunakan metode *quadratic least square regression*,. Kurva *input–output* akan menunjukkan hubungan antara bahan bakar dan daya pada suatu pembangkit. Dengan menggunakan metode *quadratic least square regression*, dapat diketahui kurva dan konstanta konsumsi bahan bakar PLTU 1, PLTU 2, PLTU 3 dan PLTU milik PT. PJB UP Gresik. Karakteristik *input-output* pada bulan Juli 2015 dengan bahan bakar *natural* gas, menunjukkan bahwa performa pembangkit masih baik setelah 34 tahun beroperasi. Perbandingan karakteristik *input-output* pada tahun 2012 dan tahun 1982, menunjukkan bahwa PLTU 1 masih memiliki performa yang baik, sedangkan PLTU 2 telah mengalami pergeseran. Pengambilan perbandingan kurva hanya dilakukan

pada pembangkit dengan periode bahan bakar yang sama. Pada skripsi ini, diambil perbandingan kurva anatara tahun 2012 dan tahun 1982 dengan bahan bakar *High Speed Diesel* (HSD).

Selain performa pembangkit yang harus dikontrol setiap saat, pemilihan pengoperasian pembangkit juga memerlukan pengaturan. Hal ini dilakukan agar mendapatkan daya yang besar dengan biaya pembangkitan paling minimal. Pengaturan pemilihan pegoperasian pembangkit ini dapat dilakukan menggunakan *economic dispatch*. Metode penyelesaian *economic dispatch* ini dapat menggunakan metode *dynamic genetic algorithm*. Metode ini pada saat diaplikasikan pada data sampel milik PT. PJB UP Gresik mendapatkan hasil dapat membagi pembebanan secara presisi tanpa selisih dengan beban yang diinginkan. Untuk optimasi konsumsi bahan bakar dan biaya bahan bakar, pada tugas akhir ini dibagi menjadi dua kategori yaitu pada saat periode bahan bakar gas dan pada saat periode bahan bakar *liquid*. Pada saat periode bahan bakar gas, metode ini dapat menghemat 3.162,9147 KNM^3 serta menghemat biaya bahan bakar sebesar $22.773 dibandingkan PT.PJB UP Gresik. Sedangkan pada saat periode bahan bakar *liquid*, metode ini menghemat 16.532,2189 liter bahan bakar dan Rp. 1.131.269.852 biaya bahan bakar.

Untuk menguji apakah metode *dynamic genetic algorithm* merupakan metode yang efektif, maka likakukan pembandingan dengan metode-metode penyelesaian *economic dispatch* lainnya seperti metode *Particle Swarm Optimization* (PSO) dan metode *Simulated Annealing* (SA). Jika dibandingkan dengan metode PSO, dari segi pembebanan metode *dynamic genetic algorithm* ini memiliki kelebihan dapat membagi pembebanan dengan selisih 0 dengan beban yang diinginkan, sementara metode PSO dalam membagi pembebanan tidak dapat presisi dan menghasilkan selisih antara -1 MW hingga 42 MW. Dari segi konsumsi bahan bakar dan biaya bahan bakar, metode *dynamic genetic algorithm* memiliki konsumsi bahan bakar 245,24 liter dan biaya bahan bakar Rp. 1.477.569 lebih hemat dibanding metode PSO. Jika dibandingkan dengan metode *Simulated Annealing* (SA), dari segi pembebanan maka metode *dynamic genetic algorithm* masih lebih unggul dibandingkan metode SA karena metode *dynamic genetic algorithm* dapat membagi beban sesuai acuan, sementara metode SA memiliki selisih -0,0001 MW hingga 0,0001 MW. Pada perbandingan konsumsi bahan bakar dan biaya bahan bakar, metode *dynamic genetic algorithm* memiliki konsumsi bahan bakar 50.511,12 liter dan biaya bahan bakar Rp. 304.329.532 lebih hemat dibanding metode SA.

PRAKATA

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Analisis Karakteristik *Input-Output* dan Optimasi Biaya Pembangkitan Menggunakan Metode *Quadratic Least Square Regression* dan Metode *Dynamic Genetic Algorithm*". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Ibu Dr. Ir. Entin Hidyah M.UM, selaku dekan Fakultas Teknik Universitas Jember;
2. Bapak Dr. Bambang Sri Kaloko, S.T., M.T., selaku Ketua Jurusan Teknik Elektro Universitas Jember;
3. Bapak Dedy Kurnia Setiawan, S.T., M.T., selaku Dosen Pembimbing Utama dan juga Dosen Pembimbing Akademik yang telah meluangkan waktu, fikiran dan tenaga dalam kelancaran penyusunan skripsi dan selama perkuliahan;
4. Bapak Dr. Triwahju Hardianto, S.T., M.T., selaku Dosen Pembimbing Anggota yang telah memberikan banyak masukan untuk penyempurnaan skripsi ini;
5. Bapak Suprihadi Prasetyono, S.T., M.T., selaku dosen penguji utama dan Bapak Samsul Bachri M, S.T., M.MT, selaku dosen penguji anggota yang telah memberikan kritik dan saran yang membangun sehingga sangat membantu terhadap penyempurnaan skripsi ini;
6. Bapak Ali Harijono, Bapak Ismail Marzuki, Mas Gunadi, Amirullah Satria, yang telah membantu proses pengambilan data di PT. PJB UP Gresik;
7. Kedua orang tuaku tercinta, (Alm) Ibu Enik Dwi Rindawati dan Papa Mistadji Arief Wahyudi yang telah memberikan motivasi, kepercayaan dan dukungan baik materi maupun moral selama penyusunan skripsi ini;

8. Kakaku tercinta Dewi Rifmawati dan Adiku tersayang Widya Pratiwi, yang memberikan semangat dan setia mendengarkan keluh kesah selama penyusunan skripsi ini;
9. Yanu Arif Santoso, yang selalu memberikan kepercayaan untuk bisa melalui setiap masalah dan menemani berproses selama ini,
10. Sahabat dan tim suksesku, Farah Adibah, Aliflah Felen dan Hendro Rosyidi Setiawan yang selalu membantu melewati masa–masa perjuanganan selama merantau untuk memperoleh gelar sarjana;
11. Rekan–rekan seperjuangan dan kakak tingkat di ProjectD yang telah memberikan informasi dan masukan penting;
12. Teman–teman seperjuangan, SATE UJ 2012 yang telah menjadi keluarga kedua selama penulis merantau;
13. Teman–teman asisten laboratorium Sistem Kendali, Ivan, Agam, Cries, Agus, Agus Loka dan seluruh asisten Laboratorium Teknik Elektro Universitas Jember yang secara langsung maupun tidak langsung mendukung penyelesaian skripsi ini;
14. Amirotun Nafissah, Bubble (Iffatur, Putri, Cibol, Ahong), Rempong (Rani, Shinta, Intan, Pujo, Ilmi), Kara (Ifit, Sekar, Firdha, Anis, Rizka, Mega), Maharema Jember, yang senantiasa memberikan semangat;
15. Teman–teman KKN 126, Adinda, Riski, Dwiki yang telah memberikan warna baru selama di Jember;
16. Semua pihak yang tidak dapat disebutkan satu persatu.

Penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, 28 Juni 2016

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan negara kepulauan yang kebanyakan memanfaatkan pembangkit listrik termal sebagai sumber penghasil listrik. Tidak dapat dipungkiri lagi, penggunaan energi listrik saat ini telah meningkat dari tahun–tahun sebelumnya. Total Terra Watt Hour (TWh) yang terjual pada Januari 2016 ialah sebesar 17,57 TWh, lebih tinggi dari Januari 2015 yang penjualannya hanya sekitar 16,34 TWh. Apabila dirupiahkan maka nilai penjualan pada Januari 2016 mencapai 17,6 triliun rupiah, lebih tinggi dari Januari 2015 lalu yang hanya 16,8 triliun rupiah (Benny Marbun, 2016).

Dengan kebutuhan listrik yang semakin meninggi, pemerintah mengoperasikan seluruh pembangkit yang dimiliki, baik pembangkit termal maupun non-termal. Namun saat ini, Indonesia masih lebih banyak menggunakan pembangkit termal untuk memenuhi kebutuhan listrik masyarakat. Pembangkit termal merupakan pembangkitan tenaga listrik yang melibatkan proses termal dalam pembangkitan tenaga listriknya, tipe pembangkitan ini membutuhkan bahan bakar yang berasal dari bahan bakar fosil.

Seiring pemakaian pembangkit termal secara terus menerus dalam pembangkitan energi listrik, mengakibatkan performa pembangkit mengalami perubahan. Perubahan ini dapat mengakibatkan efisiensi dari pembangkit juga ikut berubah. Perubahan efisiensi ini perlu untuk diawasi agar daya yang dihasilkan tetap dapat memenuhi kebutuhan. Untuk mengetahui perubahan performa suatu pembangkit ini, maka digunakanlah kurva karakteristik *input–output*. Kurva *input– output* akan menunjukkan hubungan antara bahan bakar dan daya yang dihasilkan oleh pembangkit.

Ada beberapa cara untuk menghitung karakteristik *input–output,* seperti dengan metode *Lagrange* dan metode *Gauss Jordan*. Penelitian sebelumnya menjelaskan bahwa metode *Lagrange*, memiliki kelemahan yaitu pada ketelitan yang kurang presisi (Khairudin syah dkk, 2013). Sedangkan metode *Gaus Jordan* memiliki kelemahan hanya dapat diaplikasikan pada data–data tertentu, karena pada intinya metode ini

mengharuskan data djadikan matrik eselon yang tereduksi yang tidak bisa diaplikasikan pada semua data (Neny, 2014).

Dari permasalahan ini metode *quadratic least square regression* dapat diaplikasikan untuk menyempurnakan dua metode sebelumnya, karena metode ini dapat memberikan hasil yang lebih presisi dan dengan operasi regresi sederhana sehingga dapat diaplikasikan pada semua data. Metode *least square* menyatakan bahwa "jumlah kuadrat selisih dari nilai sebenarnya dengan nilai yang terhitung, dikalikan jumlah pengukuran adalah minimum". Metode *least square* merupakan metode estimasi parameter system yang meminimumkan fungsi kriteria jumlah kuadrat kesalahan prediksi (Cahyo Adi dkk, 2011). Dari karakteristik ini diharapkan metode *quadratic least square regression* dapat menjadi penyempurnaan dari metode sebelumnya yang akan digunakan untuk memperoleh persamaan dan akhirnya dari persamaan tersebut dapat dibuat sebuah kurva karakteristik *input–output* pembangkit termal. Dari hasil kurva, dapat dilihat performa dari sebuah pembangkit dan juga kurva ini dapat digunakan sebagai parameter untuk menentukan perbaikan/*maintenance* pada pembangkit, baik perbaikan biasa maupun *overhaul.*

Dengan mengamati perubahan–perubahan pada karakteristik *input-output* pembangkit maka daya yang dihasilkan dapat dijaga kestabilannya serta dapat dioptimalkan. Hal ini dilakukan untuk menjamin kebutuhan listrik masyarakat dapat tercukupi. Selain performa pembangkit yang harus dikontrol setiap saat, pemilihan pengoperasian pembangkit juga memerlukan pengaturan. Hal ini dilakukan agar mendapatkan daya yang besar dengan biaya pembangkitan paling minimal. Pengaturan pemilihan pegoperasian pembangkit ini dapat dilakukan menggunakan *economic dispatch*.

Banyak metode yang dapat digunakan untuk menyelesaikan masalah *economic dispatch* ini, salah satunya dengan menggunakan metode *genetic algorithm*. Metode ini merupakan suatu pencarian heuristik yang didasarkan evolusi biologis. Tugas terpenting dari metode ini dalam menyelesaikan masalah *economic dispatch* yaitu

mencari sejumlah solusi yang mungkin bisa dilakukan untuk memperoleh daya paling besar dengan biaya yang paling murah. Akan tetapi, pada *paper* berjudul *Evolution of Appropiate Crossover and Mutation Operator in a Genetic Process* disebutkan bahwa metode ini hanya bisa melakukan satu kali persilangan dan mutasi untuk mendapatkan generasi baru (Tzung-Pei Hong, 2002). Karena hanya dapat melakukan persilangan dan mutasi tidak lebih dari satu kali, maka hasil yang didapatkan menjadi kurang detail. Sehingga kelemahan pada *genetic algorithm* ini disempurnakan dengan solusi menambahkan perhitungan kecocokan jumlah rasio setiap kali melakukan persilangan dan mutasi. Dengaan melakukan perhitungan rasio maka persilangan dan mutasi dilakukan lebih dari satu kali untuk mendapatkan hasil yang lebih detail dan presisi. Metode penyempurnaan dari *genetic algorithm* dengan lebih dari satu kali mutasi ini disebut *dynamic genetic algorithm*.

Dengan menggunakan metode *dynamic genetic algorithm* ini diharapkan akan didapatkan populasi yang semakin besar dengan melakukan persilangan dan mutasi lebih dari satu kali. Dengan semakin banyaknya populasi yang besar maka diharapkan solusi yang didapatkan juga semakin banyak untuk dipilih mana yang paling baik. Sehingga, metode *dynamic genetic algorithm* ini dapat memberikan solusi yang paling baik untuk penyelesaian *economic dispatch*. Untuk perbandingan, maka pada akhir tugas akhir ini metode *dynamic genetic algorithm* akan dibandingkan dengan beberapa metode penyelesaian *economic dispatch* lainnya, yaitu akan dibandingkan dengan metode *Particle Swarm Optimization* (PSO) dan metode *Simulated Annealing* (SA). Dan diharapkan metode *dynamic genetic algorithm* ini dapat memberikan hasil yang lebih baik dari dibandingkan metode–metode yang sudah digunakan pada penyelesaian *economic dispatch* ketika diterapkan dalam penyelesaian *economic dispath* pada pembangkit termal milik PT. PJB UP Gresik.

1.2 Rumusan Masalah

Berdasarkan permasalahan yang dikemukakan, maka dapat dirumuskan permasalahan sebagai berikut :

1. Bagaimana karakteristik *input–output* sistem tenaga listrik di PT. PJB UP Gresik pada periode saat ini dengan menggunakan metode *quadratic least square regression* ?
2. Bagaimana mengaplikasikan metode *dynamic genetic algorithm* untuk menyelesaikan *economic dispatch* pada sistem tenaga listrik di PT. PJB UP Gresik pada periode gas dan periode *liquid*?
3. Bagaimana menentukan pembagian pembebanan yang optimal pada setiap unit pembangkit sehingga kebutuhan sistem dapat terpenuhi?
4. Bagaimana keefisiensian metode *dynamic genetic algorithm* dibandingkan metode-metode penyelesaian *economic dispatch* lain, terutama jika dibandingkan metode *Particle Swarm Optimization* (PSO) dan metode *Simulated Annealing* (SA) ?

1.3 Batasan Masalah

Untuk menghindari meluasnya masalah, maka diberikan beberapa batasan masalah, yaitu :

1. Penelitian ini hanya dilakukan pada data sistem tenaga listrik di PT. PJB UP Gresik.
2. Jenis dan harga bahan bakar yang digunakan pada tiap pembangkit pada periode yang sama adalah sama.
3. Sampel data yang digunakan pada analisis pembebanan bahan bakar gas diambil sampel selama satu bulan, yaitu pada bulan Juli 2015, sedangkan pada analisis pembebanan bahan *bakar liquid* digunakan sampel selama lima belas hari pada bulan Desember 2012.

4. Perbandingan karakteristik *input-output* hanya dilakukan pada data dengan bahan bakar yang sama, yaitu pada bahan bakar *High Speed Diesel* (HSD).
5. Rugi–rugi diabaikan.

1.4 Tujuan

Tujuan dilakukannya penelitian adalah :

1. Mengetahui kurva karakteristik *input–output* terbaru pada sistem tenaga listrik di PT. PJB UP Gresik serta kemungkinan pergeserannya menggunakan metode *quadratic least square regression*.
2. Menentukan total biaya bahan bakar minimum dan pembagian pembebanan yang optimal pada setiap pembangkit khususnya di PT. PJB UP Gresik menggunakan *economic dispatch* dengan metode *dynamic genetic algorithm.*
3. Mengetahui pembebanan optimal yang dapat diaplikasikan pada sistem tenaga listrik di PT. PJB UP Gresik.
4. Mengetahui keefisiensian metode *dynamic genetic algorithm* dibandingkan dengan metode penyelesaian *economic dispatch* lainnya seperti metode PSO dan metode SA.

1.5 Manfaat

Diharapkan dengan dilakukannya penelitian ini, diharapkan dapat memberikan manfaat sebagai berikut :

1. Dapat membantu menjaga performa pembangkit termal yang ada pada sistem tenaga listrik di PT. PJB UP Gresik.
2. Menyelesaikan permasalahan *economic dispatch* dengan biaya pembangkitan yang minimum pada sistem tenaga listrik di PT. PJB UP Gresik.

1.6 Sistematika Penulisan

BAB 1. PENDAHULUAN

Mecakup latar belakang masalah, tujuan, rumusan masalah, batasan masalah dan sistematika pembahasan.

BAB 2. TINJAUAN PUSTAKA

Berisi tentang teori dasar pembangkit termal, biaya pembangkitan, karaktristik *input-output*, metode penyelesaian karakteristik *input-output* berupa metode *quadratic least square regression*, *economic dispatch*, metode *dynamic genetic algorithm* sebagai metode penyelesaian *economic dispatch*.

BAB 3. METODOLOGI PENELITIAN

Menjelaskan tentang metode yang digunakan untuk menyelesaikan skripsi meliputi waktu dan tempat penelitian, alat dan bahan, diagram/*flowchart* alur penelitian, diagram/*flowchart* pembuatan *Graphical User Interface* (GUI) pada perhitungan karakteristik *input-output* menggunakan metode *quadratic least square regression*, diagram/*flowchart* penyelesaian *economic dispatch* serta diagram/*flowchart dynamic genetic algorithm*.

BAB 4. HASIL DAN PEMBAHASAN

Bab ini membahas tentang hasil penelitian yang dilakukan yang berupa hasil kurva karakteristik *input-output* dan hasil *economic dispatch* menggunakan metode *dynamic genetic algorithm* beserta analisisnya serta perbandingan dengan metode penyelesaian *economic dispatch* lainnya.

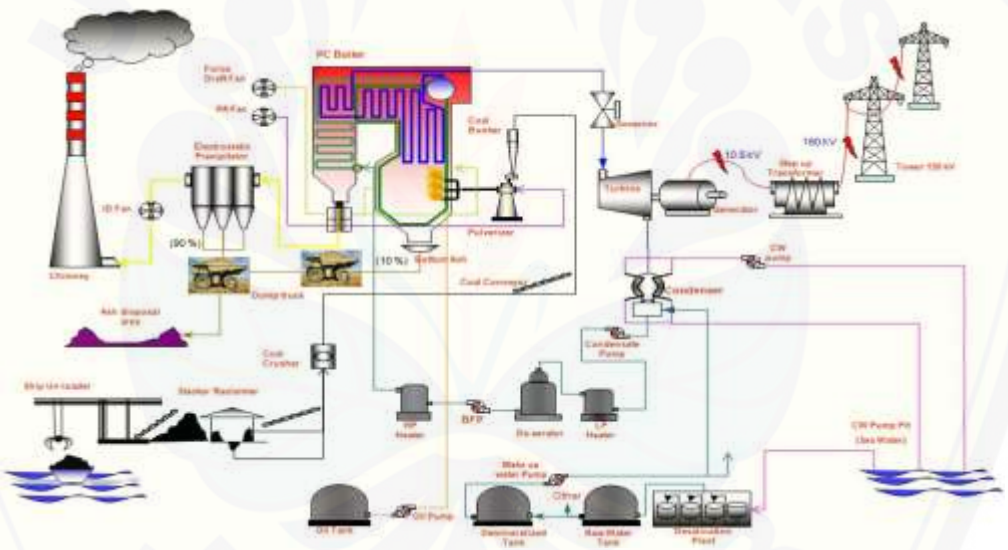
BAB 5. KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan dari analisis yang telah dilakukan dan saran untuk pengembangan skripsi ini lebih lanjut.

BAB 2. TINJAUAN PUSTAKA

2.1 Pembangkit Termal

Pembangkit Termal merupakan pembangkit tenaga listrik yang melibatkan proses panas dalam pembangkitan tenaga listriknya. Tipe pembangkitan ini membutuhkan bahan bakar yang berasal dari bahan bakar fosil. Pembangkit Listrik Tenaga Gas (PLTG), Pembangkit Listrik Tenaga Uap (PLTU) dan Pembangkit Listrik Tenaga Diesel (PLTD) merupakan beberapa contoh dari pembangkit–pembangkit yang merupakan pembangkit termal.



Gambar 2.1 Siklus Pembangkit Termal (PLTU)

sumber : tapakpakulangit.wordpress.com

Salah satu pembangkit termal yang digunakan di Indonesia adalah PLTU yang dioperasikan oleh PJB UP Gresik. Terdapat empat buah PLTU yang dioperasikan oleh PJB UP Gresik saat ini yang digunakan untuk memenuhi kebutuhan listrik. Empat PLTU ini dapat memberikan sumbangan daya sebesar ± 600 MW. Siklus umum pada pembangkit termal khususnya pada PLTU ditunjukkan pada gambar 2.1. Secara

sederhana siklus PLTU sama seperti proses memasak air. Mula-mula air ditampung dalam tempat memasak (*hotwell*) dan kemudian diberi panas dari sumbu api dalam *boiler*. Akibat pembakaran menimbulkan air terus mengalami kenaikan suhu sampai pada batas titik didihnya. Karena pembakaran terus berlanjut maka air yang dimasak melampaui titik didihnya sampai timbul uap panas. Uap ini lah yang digunakan untuk memutar turbin dan generator yang nantinya akan menghasilkan energi listrik.

2.2 Biaya Pembangkitan

Dalam pembangkitan tenaga listrik ada beberapa komponen biaya yang biasanya harus diperhitungkan, yaitu (Khairudin dkk, 2012) :

1. *Fixed Cost*

   yakni biaya yang harus tetap dikeluarkan dalam keadaan suatu pembangkit sedang beroperasi maupun tidak beroperasi. Komponen ini umumnya terdiri dari biaya konstruksi pembangkit, biaya pembelian turbin, generator, dan lain-lain.

2. *Variable Cost*

   yaitu biaya yang dikeluarkan untuk operasi dan perbaikan pada pembangkit. *Variable cost* ini mencakup biaya gaji pegawai atau karyawan, biaya manajemen, biaya untuk pelumas, serta kebutuhan perbaikan dan perawatan lainnya. Semakin sering dan berat kerja pembangkit, semakin dibutuhkan pula perbaikan atau perawatan pada beberapa bagian pada pembangkit, sehingga mengakibatkan biaya *variable cost* juga akan meningkat.

3. *Fuel Cost*

   Fuel cost atau biasa disebut sebagai biaya bahan bakar yang diperlukan oleh pembangkit untuk memproduksi listrik. Beberapa faktor yang mempengaruhi harga bahan bakar ini misalnya banyaknya konsumsi bahan bakar yang diperlukan, jenis bahan bakarnya, lama waktu pembangkit beroperasi, dan beberapa hal lainnya.

4. Biaya optional

Biaya ini merupakan biaya tidak wajib yang harus ada dalam komponen biaya pembangkitan. Namun, saat berada dalam posisi *Independent Power Producer* (IPP) atau penyedia listrik non-PLN (pemerintah), terkadang komponen biaya ini turut kita perhitungkan. Misalnya biaya saluran dari trafo *step-up* yang ada di pembangkit kita ke gardu induk PLN terdekat.

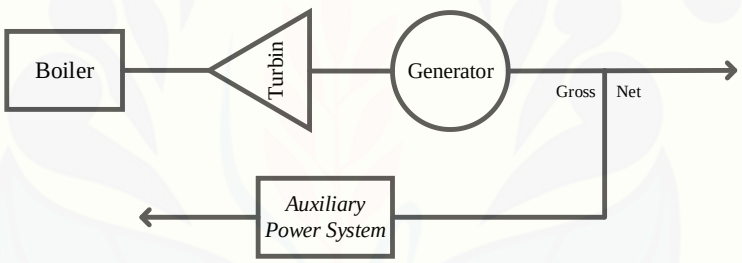
Biaya operasi dari suatu sistem tenaga listrik merupakan biaya terbesar dalam pengoperasian suatu perusahaan listrik. Biaya yang dikeluarkan oleh suatu perusahaan listrik untuk menghasilkan energi listrik ditentukan oleh biaya operasi dan biaya investasi. Besarnya biaya investasi tidak bergantung pada besar daya keluaran pembangkit, tetapi bergantung pada besar kapasitas daya yang terpasang pada pembangkit. Biaya investasi sendiri mencakup biaya pembangunan pusat pembangkit listrik, jaringan transmisi dan distribusi serta peralatan sistem lainnya. Sedangkan biaya operasi atau biaya produksi lebih condong kepada biaya pengoperasian pembangkit. Salah satu cara optimasi pembangkitan adalah dengan cara meminimalkan biaya operasi pembangkitan, dengan kata lain optimasi pembangkitan merupakan suatu proses untuk meminimalkan biaya pembangkitan namun dapat menghasilkan daya yang paling maksimal (Neny, 2014).

2.3 Karakteristik *Input-Output*

Dalam pengoperasian sebuah pembangkit secara terus menerus, pastinya akan mempengaruhi performa dari pembangkit itu sendiri. Pembangkit dapat mengalami penurunan performa baik sedikit demi sedikit maupun secara drastis. Untuk itu dibutuhkan suatu acuan untuk mengetahui apakah performa pembangkit masih dalam keadaan stabil atau mengalami penurunan. Dalam hal ini, karakteristik *input–output* dapat diterapkan untuk mengetahui keadaan performa pembangkit termal.

Dasar dari permasalahan pengoperasian pembangkit secara ekonomis adalah mengatur karakteristik *input-output* dari unit pembangkit termal itu sendiri.

Karakteristik unit pembangkit termal terdiri dari boiler, turbin, generator dan *auxiliary power system* seperti yang ditunjukkan pada gambar 2.2. *Output* pembangkit selain disalurkan melalui transmisi juga akan digunakan oleh *auxiliary power system* atau sistem tenaga bantu untuk beroperasi. Pada gambar tersebut juga terdapat *gross* atau *input* kotor dan juga *net* yang direpresentasikan sebagai *output* bersih. *Input* kotor merupakan *input* total yang diukur dalam biaya/jam sedangkan *output* bersih merupakan daya *output* yang dihasilkan. Dengan menggunakan karakteristik *input–output* pada masing–masing pembangkit, maka dapat ditentukan pengoperasian optimum secara ekonomis sejumlah unit pembangkit untuk menentukan total biaya operasinya paling minimal.



Gambar 2.2 Karakteristik Unit Pembangkit

Karakteristik *input–output* pembangkit termal merupakan suatu karakteristik yang menggambarkan hubungan antara *input* dan *output* yang dimiliki oleh pembangkit. *Input* yang dimaksudkan adalah bahan bakar (liter/jam) yang dipakai oleh pembangkit. Sedangkan *output* yang dihasilkan adalah berupa daya (MW). Pada umumnya karakteristik *input–output* pembangkit termal didekati dengan fungsi polinominal orde dua, yaitu (AM. Ilyas dkk, 2010) :

$$Hi = \propto i + \beta i Pi + \gamma i Pi^{2} \quad \text{..... (2.1)}$$

Dengan :

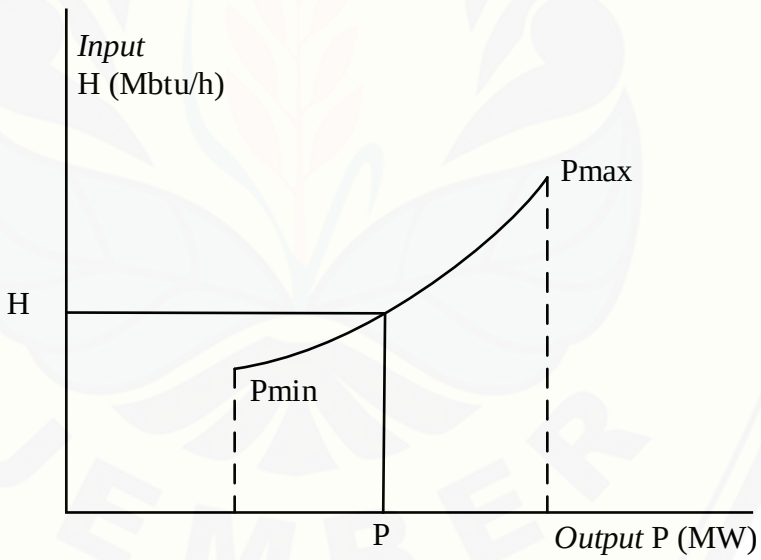
Hi = *input* bahan bakar pembangkit termal ke-*i* (liter/jam)

Pi = *output* pembangkit termal kw-*i* (MW)

$\propto i, \beta i, \gamma i$ = konstanta *input* - *output* pembangkit termal ke-*i*.

Penentuan parameter $\propto i, \beta i, \gamma i$ membutuhkan data yang berhubungan dengan *input* bahan bakar *Hi* dan *output* pembangkit *Pi*.

Gambar 2.3 merupakan karakteristik *input–output* dari unit pembangkit termal yang ideal. Kurva ideal digambarkan sebagai kurva non-linier yang kontinyu. Pada grafik terdapat variabel *input* H yang direpresentasikan sebagai energi panas yang dibutuhkan dengan satuan Mbtu/h. Sedangkan *output* simbolkan dengan P yang berarti daya dengan satuan MW. Pada gambar ditunjukkan sebuah kurva hubungan *input* dan *output* yang dibatasi oleh Pmin dan Pmax.



Gambar 2.3 Karakteristik *Input-Output* Unit Pembangkit Termal Ideal

Data karakteristik *input–output* diperoleh dari perhitungan desain atau dari pengukuran. Jika digunakan data pengukuran akan diperoleh kurva yang tidak kontinyu. Dan pada pembangkit termal mempunyai batas operasi minimum dan

maksimum. Batasan ini biasanya disebabkan oleh kestabilan pembakaran dan masalah desain generator. Pada umumnya unit pembangkit termal tidak dapat beropersi dibawah 30% dari kapaitas desain (Neny, 2014).

2.4 Metode Penyelesaian Karakteristik *Input–Output*

Metode penyelesaian karakteristik *input-output* merupakan metode-metode yang dapat menentukan konstanta konsumsi bahan bakar. Metode-metode yang digunakan seperti metode *Lagrange*, metode *Gauss Jordan* dan metode *quadratic least square regression*.

2.4.1 Metode *Lagrange*

Salah satu metode konvensional yang umum digunakan untuk menyelesaikan masalah optimisasi biaya atau *economic dispatch* adalah metode *Lagrange.* Metode *Lagrange* terbagi menjadi dua yaitu *losses* diabaikan dan *losses* diperhitungkan. Dalam sistem tenaga, kerugian transmisi merupakan kehilangan daya yang harus ditanggung oleh sistem pembangkit. Jadi kerugian transmisi ini merupakan beban bagi sistem tenaga (Khairudin Syah dkk, 2013).

Pendekatan yang khas pada metoda *Lagrange* untuk ditambahkan dalam fungsi objektif disebut dengan faktor pengali *Lagrange*. Persamaan faktor pengali *Lagrange* dituliskan pada persamaan (2.2).

$$L = F_T + \lambda(P_R - \sum_{i=1}^{n} P_1) \quad \text{..... (2.2)}$$

Persamaan *Lagrange* tersebut merupakan fungsi dari *output* pembangkit, keadaan optimum dapat diperoleh dari persamaan *Lagrange* sama dengan nol.

$$\frac{\partial L}{\partial P_i} = \frac{\partial F_T}{\partial P_i} + \lambda\left(\frac{\partial P_R}{\partial P_i} - \frac{\partial P_i}{\partial P_i}\right) = 0 \quad \text{..... (2.3)}$$

$$\frac{\partial F_T}{\partial P_i} + \lambda(0 - 1) = 0 \quad \text{..... (2.4)}$$

$$\frac{\partial F_T}{\partial P_i} = \lambda \quad \text{(2.5)}$$

Kondisi operasi ekonomis adalah:

$$2a_1P_1 + b_1 = \lambda \quad \text{(2.6)}$$

$$\sum_{i=1}^{n} P_i = P_R \quad \text{(2.7)}$$

$$P_{imin} \leq P_i \leq P_{imax} \quad \text{(2.8)}$$

dengan keterangan:

L : faktor pengali *Lagrange*

F_T : total biaya pembangkitan (Rp)

P_i : *output* pembangkit ke-i (MW)

P_R : total kebutuhan beban pada sistem (MW)

a_i, b_i : konstanta *input* pembangkit ke-i

Dengan batasan menggunakan persamaan kesetimbangan daya (*equality constraint*) dimana total daya yang dibangkitkan oleh masing-masing unit pembangkit harus sama dengan total kebutuhan beban. Penggunaan batasan pertidaksamaan (*inequality constraint*), daya *output* dari tiap unit harus lebih besar atau sama dengan daya minimum yang dibolehkan dan harus juga kurang dari atau sama dengan daya maksimum yang diperbolehkan (Khairudin Syah dkk, 2013).

2.4.2 Metode *Gauss Jordan*

Salah satu metode yang dapat digunakan untuk menyelesaikan sistem persamaan linier adalah metode eliminasi *Gauss Jordan*. Metode ini diberi nama *Gauss Jordan* untuk menghormati CarlFriedrich Gauss dan Wilhelm Jordan. Metode ini sebenarnya

adalah modifikasi dari metode eliminasi *Gauss*, yang dijelaskan oleh Jordan di tahun 1887 (Bangkit, 2014).

Metode *Gauss Jordan* ini menghasilkan matriks dengan bentuk baris eselon yang tereduksi (*reduced row echelon form*), sementara eliminasi *Gauss* hanya menghasilkan matriks sampai pada bentuk baris eselon (*row echelon form*). Selain untuk menyelesaikan sistem persamaan linier, metode eliminasi *Gauss Jordan* ini dapat dikembangkan dari metode eliminasi, yaitu menghilangkan atau mengurangi jumlah variabel sehingga dapat diperoleh nilai dari suatu variabel yang bebas.

Eliminasi *Gauss Jordan* adalah pengembangan dari eliminasi *Gauss* yang hasilnya lebih sederhana lagi. Caranya adalah dengan meneruskan operasi baris dari eliminasi *Gauss* sehingga menghasilkan matriks yang eselon baris. Metode ini juga dapat digunakan sebagai salah satu metode penyelesaian persamaan linear dengan menggunakan matriks. Metode ini digunakan untuk mencari invers dari sebuah matriks (Neny, 2014).

Prosedur umum untuk metode eliminasi *Gauss Jordan* ini adalah :

a) Ubah sistem persamaan linier yang ingin dihitung menjadi matriks augmentasi.
b) Lakukan operasi baris elementer pada matriks augmentasi (A|b) untuk mengubah matriks A menjadi dalam bentuk baris eselon yang tereduksi. Pengubahan dilakukan dengan membuat matriks yang elemen-elemennya adalah koefisien - koefisien dari sistem persamaan linier. Sedangkan langkah-langkah pada operasi baris elementer yaitu :
    1. Menukar posisi dari 2 baris.
    $$A_i \leftrightarrow A_j$$
    2. Mengalikan baris dengan sebuah bilangan skalar positif.
    $$A_i = k * A_j$$
    3. Menambahkan baris dengan hasil kali skalar dengan baris lainnya.

Algoritma metode eliminasi *Gauss* adalah :

1. Masukkan matrik A, dan vektor B beserta ukurannya n.

2. Buat augmented matrik [A|B] namakan dengan A.
3. Untuk baris ke i dimana i=1 s/d n, perhatikan apakah nilai ai,i = 0 :

   Bila ya :

   pertukarkan baris ke i dan baris ke i+k=n, dimana ai+k ,i ≠0, bila tidak ada berarti perhitungan tidak bisa dilanjutkan dan proses dihentikan dengan tanpa penyelesaian. Bila tidak : lanjutkan
4. Untuk baris ke j, dimana j = i+1 s/d n

Prinsip eliminasi Gauss-Jordan :

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} a_{11} & 0 & 0 & b'_1 \\ 0 & a_{22} & 0 & b'_2 \\ 0 & 0 & a_{33} & b'_3 \end{array}\right]$$

Sehingga :

$$\begin{aligned} x_1 \quad\quad\quad &= b'_1 \\ x_2 \quad\quad &= b'_2 \\ x_3 &= b'_3 \end{aligned}$$

2.4.3 Metode *Quadratic Least Square Regression*

Metode *least square* menyatakan bahwa "jumlah kuadrat selisih dari nilai sebenarnya dengan nilai yang terhitung, dikalikan jumlah pengukuran adalah minimum". Metode *least square* merupakan metode estimasi parameter sistem yang meminimumkan fungsi kriteria jumlah kuadrat kesalahan prediksi (Cahyo Adi dkk, 2011). Salah satu cabang dari metode *least square* adalah metode *quadratic least square regression* merupakan salah satu metode pendekatan yang dapat digunakan untuk menyelesaikan persoalan pada karakteristik *input–output* pembangkit termal. Metode ini banyak digunakan untuk regresi ataupun pembentukan persamaan dari

titik–titik data diskrit dalam suatu pemodelan dan juga dapat digunakan untuk analisis sesatan pengukuran dalam validasi model. Metode ini termasuk dalam metode–metode pendekatan sesatan terdistribusi. Berdasarkan karakteristik kerjanya yang melakukan pengurangan sesatan menyeluruh yang terukur berdasarkan interval pendekatan keseluruhan sesuai dengan order pendekatan yang meningkat. Metode *quadratic least square regression* juga memainkan peranan penting dalam teori statistik, karena metode ini seringkali digunakan dalam penyelesaian masalah–masalah yang melibatkan kumpulan data yang tersusun secara acak.

Pada tugas akhir ini, metode *quadratic least square regression* digunakan untuk melakukan regresi atau pencocokan kurva pada permasalahan karakteristik *input-output* pembangkit termal yang diharapkan dapat membentuk persamaan matematis tertentu. Karakteristik *input–output* pembangkit termal merupakan fungsi polinominal orde dua, sehingga digunakan persamaan berikut :

a. Untuk mencari koefisien (a,b,c)

$$a = \frac{\{[\sum(x^2y) \times \sum(x^2) - \sum(xy) \times \sum(x^3)]\}}{\{[\sum(x^2) \times \sum(x^4) - [\sum(x^3)]^2} \quad \text{..... (2.9)}$$

$$b = \frac{\{[\sum(xy) \times \sum(x^4) - \sum(x^2y) \times \sum(x^3)]\}}{\{[\sum(x^2) \times \sum(x^4) - [\sum(x^3)]^2} \quad \text{..... (2.10)}$$

$$c = \left[\frac{\sum y_i}{n}\right] - \left\{b \times \left[\frac{\sum x_i}{n}\right]\right\} - \left\{a \times \left[\frac{\sum x_i^2}{n}\right]\right\} \quad \text{..... (2.11)}$$

b. Persamaan Kuadrat Regresi Statistik

$$\sum x^2 = \left(\sum x_i^2\right) - \left[\frac{\sum x_i^2}{n}\right] \quad \text{..... (2.12)}$$

$$\sum xy = \left(\sum x_i y_i\right) - \left[\frac{\sum x_i \times \sum y_i}{n}\right] \quad \text{..... (2.13)}$$

$$\sum x^3 = \left(\sum x_i^3\right) - \left[\frac{\sum x_i \times \sum x_i^2}{n}\right] \quad \text{..... (2.14)}$$

$$\sum x^2 y = (\sum x_i^2 y_i) - \left[\frac{\sum x_i^2 \times \sum y_i}{n}\right] \quad \text{..... (2.15)}$$

$$\sum x^4 = (\sum x_i^4) - \left[\frac{(\sum x_i^2)^2}{n}\right] \quad \text{..... (2.16)}$$

Dengan keterangan :

y_i = nilai individu untuk setiap gabungan variabel

x_i = nilai individu untuk setiap gabungan variabel

n = jumlah data

2.5 *Economic Dispatch*

Economic dispatch adalah pembagian pembebanan pada setiap unit pembangkit sehingga diperoleh kombinasi unit pembangkit yang dapat memenuhi kebutuhan beban dengan biaya yang optimum. Dengan kata lain, *economic dispatch* digunakan untuk mencari nilai optimum dari *output* daya dari kombinasi beberapa unit pembangkit yang bertujuan untuk meminimalkan total biaya pembangkitan dan dapat memenuhi batasan *equality* dan *inequality*. Secara umum, fungsi biaya dari tiap pembangkit dapat diformulasikan sebagai suatu fungsi obyektif seperti pada persamaan berikut (Bangkit, 2014):

$$F_T = \sum_{I-1}^{N} F_i(P_i) \quad \text{..... (2.17)}$$

$$Fi(Pi) = a_1 + b_i P_i + c_i P_i^2 \quad \text{..... (2.18)}$$

Dengan keterangan :

F_T = total biaya pembangkitan (Rp)

Fi(Pi) = fungsi biaya *input-output* dari pembangkit *i* (Rp/jam)

a_1, b_i, c_i = koefisien biaya dari pembangkit *i*

Pi = *output* pembangkit *i* (MW)

n = jumlah unit pembangkit

i = indeks dari unit pembangkit ke-i

Ada dua hal mendasar yang dilakukan *economic dispatch*, yaitu perencanaan hari berikutnya dan pembagian beban pada hari-H (FERC Staff, 2005).

a. Perencanaan *economic dispatch* untuk hari berikutnya meliputi penjadwalan unit pembangkit berdasarkan beban perkiraan untuk hari berikutnya, memilih unit pembangkit yang akan dijalankan, mengatur tingkat *ramp* atau seberapa cepat *output* generator dapat diubah, mengatur level maksimum ataupun minimum generator dan juga mengatur kapan waktu generator harus menyata ataupun padam. *Economic dispacth* juga akan mengenali dan mengkalkulasi biaya pembangkitann berdasarkan efisiensi dan biaya operasi.
b. *Economic dispatch* ada hari-H. Dapat dikatakan bahwa tugas *economic dispatch* pada hari-H adalah memantau keseimbangan beban antara pemasokan dan kebutuhan menjadi seimbang. Selain itu juga, *economic dispatch* dapat memantau aliran daya pada sistem transmisi untuk menjaga tegangan yang dikirimkan stabil.

2.6 Metode Penyelesaian *Economic Dispatch*

Metode penyelesaian *economic dispatch* merupakan metode yang digunakan untuk pembagian pembebanan secara optimal. Metode ini harus dapat melakukan optimasi untuk mencari nilai konsumsi bahan bakar paling murah dengan kombinasi-kombinasi daya pada masing-masing pembangkit. Metode-metode penyelesaian *economic dispatch* diantaranya adalah metode *Particle Swarm Optimizatio* (PSO), metode *Simulated Annealing* (SA) dan metode *dynamic genetic algorithm.*

2.6.1 Metode *Partikel Swarm Optimization* (PSO)

Algoritma PSO diperkenalkan oleh Kennedy dan Eberhart pada tahun 1995, proses algoritmanya diinspirasikan oleh perilaku sosial dari binatang, seperti sekumpulan burung dalam suatu kumpulan (*swarm).* Perilaku social terdiri atas

tindakan individu dan pengaruh individu-individu lain dalam suatu kelompok. Setiap partikel di dalam PSO juga berhubungan dengan suatu kecepatan (*velocity*). Partikel–partikel mempunyai kecenderungan untuk bergerak ke area penelusuran yang lebih baik setelah melewati proses penelusuran (Franki dkk, 2014).

PSO adalah salah satu dari teknik komputasi *evolusioner*, yang mana populasi pada PSO didasarkan pada penelusuran algoritma dan diawali dengan suatu populasi yang random yang disebut dengan *partikel*. Berbeda dengan teknik komputasi *evolusioner* lainnya, setiap partikel di dalam PSO juga berhubungan dengan suatu *velocity*. Partikel–partikel tersebut bergerak melalui penelusuran ruang dengan *velocity* yang dinamis yang disesuaikan menurut perilaku historisnya.

Partikel Swarm Optimization (PSO) mempunyai kesamaan dengan *Genetic Algorithm* (GA) yang mana dimulai dengan suatu populasi yang random dalam bentuk matriks. Namun PSO tidak memiliki operator evolusi yaitu *crossover* dan mutasi seperti yang ada pada *genetic algorithm* sebagai kromosom yang terdiri dari nilai suatu variabel. Setiap partikel berpindah dari posisinya semula ke posisi yang lebih baik dengan suatu *velocity* (Bangkit, 2014).

Pada algoritma PSO, vektor kecepatan diperbaharui untuk masing-masing partikel, kemudian menjumlahkan vektor kecepatan tersebut ke posisi partikel. Proses memperbaharui kecepatan dipengaruhi oleh kedua solusi, yaitu melakukan penyesuaian posisi terbaik dari partikel (*particle best*) dan penyesuaian terhadap partikel terbaik dari seluruh kawanan (*global best*). Pada tiap iterasi, setiap solusi direpresentasikan oleh posisi partikel dievaluasi dengan cara memasukkan solusi tersebut ke dalam *fitness function* (Santosa dan Willy, 2011).

Kesederhanaan algoritma dan performansinya yang baik, menjadikan PSO telah menarik banyak perhatian di kalangan para peneliti dan telah diaplikasikan dalam berbagai persoalan optimisasi sistem tenaga seperti *economic dispatch*, design kontrol PID pada sistem AVR, kontrol tegangan dan daya reaktif, *unit commitment* dan lain sebagainya. PSO telah populer menjadi optimisasi global dengan sebagian besar

permasalahan dapat diselesaikan dengan baik dimana variabel – variabelnya adalah bilangan riil (Bangkit, 2014).

Beberapa istilah umum yang biasa digunakan dalam PSO dapat didefinisikan sebagai berikut :

a) *Swarm* : populasi dari suatu algoritma.
b) *Partikel* : anggota (individu) pada suatu *swarm.*

Setiap partikel merepresentasikan suatu solusi yang potensial pada permasalahan yang diselesaikan. Posisi dari suatu partikel adalah ditentukan oleh representasi solusi saat itu.

c) *Pbest (Personal best)* : posisi partikel yang dipersiapkan untuk mendapatkan suatu solusi yang terbaik.
d) *Gbest (Global best)* : posisi terbaik partikel pada *swarm.*
e) *Velocity (vector)* : vektor yang menggerakkan proses optimisasi yang menentukan arah di mana suatu partikel diperlukan untuk berpindah *(move)* untuk memperbaiki posisinya semula.
f) *Inertia weight* : disimbolkan *w*, parameter ini digunakan untuk mengontrol dampak dari adanya *velocity* yang diberikan oleh suatu partikel. Perubahan *velocity* pada algoritma PSO terdiri atas tiga bagian yaitu *social part, cognitive part dan momentum part.* Ketiga bagian tersebut menentukan keseimbangan antara kemampuan penelusuran *global* dan *local*, oleh karena itu dapat memberikan performansi yang baik pada PSO. Parameter *inertia weight* digabungkan dengan *sosial part* di dalam algoritma PSO standar.

Prosedur algoritma PSO adalah sebagai berikut :

1. Inisialisasi populasi dari partikel-partikel dengan posisi dan kecepatan secara acak
2. Evaluasi nilai *fitness* untuk masing-masing pastikel
3. Pembandingan dan pembaharuan *particle best* dan *global best* untuk tiap-tiap partikel berdasarkan fungsi *fitness*. Tahap selanjutnya adalah pengulangan langkah berikut sampai *stopping criteria* terpenuhi :

a. Menggunakan *particle best* dan *global best* yang ada, perbaharui kecepatan setiap partikel. Dari kecepatan yang diperoleh, perbaharui posisi partikel dengan menjumlahkan kecepatan baru dan posisi sebelumnya menggunakan persamaan 2.19 dan 2.20.

$$V_i(t) = wv_i(t-1) + c_i r_i\left(x_{pi} - x_i\right) + c_2 r_2\left(x_{gi} - x_i\right) \quad \text{..... (2.19)}$$

dan

$$X_i(t) = x_i(t-1) + V_i(t) \quad \text{..... (2.20)}$$

dengan keterangan :

i = indeks partikel

t = iterasi

w = *inertia*

v_i = kecepatan partikel ke-*i*

x_i = posisi partikel ke-*i*

x_{gi} = posisi terbaik dari smua partikel (*gbest*)

x_{pi} = posisi terbaik dari partikel ke-*i* (*pbest*)

$c_{1,2}$ = *learning rate*

$r_{1,2}$ = bilangan acak [0,1]

b. Evaluasi *fitness* dari setiap partikel.

c. Bandingkan dan perbaharui *particle best* dan *global best* tiap-tiap partikel berdasarkan *fungsi fitness*.

d. Cek *stopping criteria*. Jika terpenuhi, berhenti. Jika tidak maka kembali ke (a).

2.6.2 Metode *Simulated Annealing* (SA)

Simulated annealing (SA) adalah salah satu algoritma untuk optimasi yang bersifat *generic*. Berbasis probabilitas dan mekanika statistik, algoritma ini dapat digunakan untuk mencari pendekatan terhadap solusi optimum global dari suatu

permasalahan. Masalah yang membutuhkan pendekatan SA adalah masalah–masalah optimasi kombinatorial, dimana ruang pencarian solusi yang ada terlalu besar, sehingga hampir tidak mungkin ditemukan solusi eksak terhadap permasalahan tersebut. Publikasi pendekatan ini pertama kali dilakukan oleh S. Krikpatrick, C. D Gelatt dan M. P. Vecchi, diaplikasikan pada desain optimal *hardware computer*, dan juga pada salah satu masalah klasik ilmu komputer yaitu *Travelling Salesman Problem* (Kirkpatrick, 1983).

Simulated annealing adalah salah satu teknik yang dikenal dalam bidang metalurgi, digunakan dalam mempelajari proses pembentukan kristal dalam suatu materi. Agar dapat terbentuk suatu susunan kristal yang sempurna, diperlukan persamaan sampai suatu tingkat tertentu, kemudian dilanjutkan dengan pendinginan yang perlahan-lahan dan terkendali dari materi tersebut. Pemanasan materi di awal proses *annealing*, memberikan kesempatan pada atom–atom dalam materi itu untuk bergerak secara bebas, mengingat tingkat energi dalam kondisi panas ini cukup tinggi. Proses pendiginan yang perlahan–lahan memungkinkan atom–atom yang tadinya bergerak bebas itu, pada akhirnya menemukan tempat yang optimum, dimana energi internal yang dibutuhkan atom itu untuk mempertahankan posisinya adalah minimum. *Simulated annealing* berjalan berdasarkan analogi dengan proses *annealing* tersebut. *Simulated annealing* memanfaatkan analogi antara pendinginan dan pembekuan metal menjadi sebuah struktur *crystal* dan energy yang minimal (proses penguatan) dan proses pencarian untuk tujuan minimal (Dhany Indrawan, 2008).

Berikut adalah algoritma metode *simulated annealing* :

a. Evaluasi keadaan awal. Jika tujuan telah terpenuhi maka pencarian solusi selesai. Jika tidak lanjutkan dengan keadaan awal sebagai keadaan sekarang.
b. Inisialisasi *best_so_far* untuk keadaan sekarang.
c. Inisialisasi suhu (T) sesuai dengan *annealing schedule*.
d. Kerjakan hingga solusi ditemukan atau sudah tidak ada operator baru lagi akan diaplikasikan ke kondisi sekarang.

1. Gunakan operator yang belum pernah digunakan untuk menghasilkan keadaan baru.
2. Evaluasi kondisi baru dengan menghitug :

   ΔE = nilai sekarang – nilai keadaan baru ..... (2.22)

   - Jika kondisi baru merupakan tujuan yang dicari maka pencarian solusi selesai.
   - Jika bukan tujuan, namun nilainya lebih baik dari sekarang, maka jadikan keadaan tersebut sebagai keadaan sekarang.
   - Jika nilai kondisi baru tidak lebih baik dari sekarang, maka tetapkan kondisi baru sebagai keadaan sekarang dengan probabilitas :

   $$p' = e^{\frac{-\Delta E}{kT}} \quad \text{..... (2.23)}$$

   dengan keterangan : ΔE = perubahan suhu

   kT = temperatur saat ini

   dalam kondisi ini generate random number dengan range [0,1]. Jika random number lebih kecil dari p' maka solusi diterima. Jika random number lebih besar dari p' maka solusi tidak di terima.
3. Perbaiki T sesuai dengan *annealing schedule*.

e. *best_so_far* adalah solusi yang dicari.

2.6.3 Metode *Genetic Algorithm*

Metode *genetic algorithm* memiliki beberapa turunan dan penyempurnaan seiring dengan perkembangannya. Sehingga pada materi ini metode *genetic algorithm* dibagi menjadi dua bagian, yaitu :

A. *Genetic Algorithm* Konvensional

Genetic algorithm sebagai cabang dari *evolution algorithm* merupakan metode *adaptive* yang biasa digunakan untuk memecahkan suatu pencarian nilai dari sebuah

masalah optimasi. Algoritma ini didasarkan pada proses genetik yang ada dalam makhluk hidup, yaitu perkembangan generasi dalam sebuah populasi yang alami, secara lambat laun mengikuti prinsip seleksi alam atau "siapa yang kuat, dia yang bertahan (*survive*)". Dengan meniru evolusi ini, *genetic algorithm* dapat digunakan untuk mencari solusi permasalahan–permasalahan dunia nyata (Eka Risky dkk, 2012).

Peletak prinsip dasar sekaligus pencipta *genetic algorithm* adalah John Holland. *Genetic algorithm* menggunakan analogi secara langsung dari kebiasaan yang alami yaitu seleksi alam. Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu–individu yang masing–masing individu merepresentasikan sebuah solusi yang mungkin bagi persoalan yang ada. Dalam kaitan ini, individu dilambangkan dengan sebuah nilai *fitness* yang akan digunakan untuk mencari solusi terbaik dari persoalan yang ada.

Pertahanan yang tinggi dari individu memberikan kesempatan untuk melakukan reproduksi melalui perkawinan silang dengan individu yang lain dalam populasi tersebut. Individu baru yang dihasilkan dalam hal ini dinamakan keturunan, yang membawa beberapa sifat induknya. Sedangkan individu dalam populasi yang tidak terseleksi dalam reproduksi akan mati dengan sendirinya. Dengan jalan ini, beberapa generasi dengan karakteristik yang bagus akan bermunculan dalam populasi tersebut untuk kemudian dicampur dan ditukar dengan karakter yang lain. Dengan mengawinkan semakin banyak individu, maka akan semakin banyak kemungkinan terbaik yang dapat diperoleh (Kusumaningtyas, 2016).

Genetic algorithm sudah banyak digunakan pada masalah praktis yang berfokus pada pencarian parameter-parameter atau solusi yang optimal. Hal ini membuat banyak orang mengira bahwa *genetic algorithm* hanya dapat digunakan untuk menyelesaikan masalah optimasi saja. Namun, pada kenyataanya *genetic algorithm* juga memiliki kemampuan untuk menyelesaikan masalah-masalah selain optimasi. Tapi perlu diakui, bahwa kemampuan *genetic algorithm* dalam menyelesaikan masalah optimasi

merupakan metode yang paling efisien saat ini. *Genetic algorithm* banyak diaplikasikan untuk berbagai macam permasalahan, yaitu (Eka Risky dkk, 2012) :

1. Optimasi

Beberapa penggunaan *genetic algorithm* untuk optimasi antara lain untuk optimasi numerik dan optimasi kombinatorial seperti *Traveling Salesmen Problem* (TSP), Perancangan *Integrated Circuit* atau IC, *Job Scheduling*, dan Optimasi video dan suara.

2. Pemrograman Otomatis

Genetic algorithm untuk pemrograman otomatis antara lain untuk melakukan proses evolusi terhadap program komputer dalam merancang struktur komputasional, seperti *cellular automata* dan *sorting networks*.

3. *Machine Learning*

Genetic algorithm juga telah berhasil diaplikasikan untuk memprediksi struktur protein. *Genetic algorithm* juga berhasil diaplikasikan dalam perancangan *neural networks* (jaringan syaraf tiruan) untuk melakukan proses evolusi terhadap aturan–aturan pada *learning classifier system* atau *symbolic production system* dan dapat digunakan untuk mengontrol robot.

4. Model Ekonomi

Dalam bidang ekonomi, *genetic algorithm* digunakan untuk memodelkan prosesproses inovasi dan pembangunan *bidding strategies*.

5. Model Sistem Imunisasi

Contoh penggunaan *genetic algorithm* dalam bidang ini untuk memodelkan berbagai aspek pada sistem imunisasi alamiah, termasuk *somatic mutation* selama kehidupan individu dan menemukan keluarga dengan gen ganda (*multi gen families*) sepanjang waktu evolusi.

6. Model Ekologis

Genetic algorithm juga dapat digunakan untuk memodelkan fenomena ekologis seperti *host-parasite co evolutions*, simbiosis dan aliran sumber di dalam ekologi.

Genetic algorithm memiliki keunggulan-keunggulan dibandingkan dengan metode-metode heuristik yang lain. Keuntungan penggunaan *genetic algorithm* terlihat dari kemudahan implementasi dan kemampuannya untuk menemukan solusi yang optimal dan bisa diterima secara cepat untuk masalah-masalah berdimensi tinggi. *Genetic algorithm* sangat berguna dan efisien untuk masalah dengan karakteristik sebagai berikut :

a) Dapat menyelesaikan masalah dengan ruang masalah sangat besar, kompleks, sulit dipahami, kurang atau bahkan tidak ada pengetahuan yang memadai untuk merepresentasikan masalah ke dalam ruang pencarian yang lebih sempit
b) Dapat menyelesaikan masalah yang tidak tersedia analisis matematika yang memadai.
c) *Genetic algorithm* menyelesaikan masalah dengan mengkodekan permasalah menjadi kromosom, bukan dengan menyelesaikan permasalahan itu sendiri. Karena itu diperlukan pemodelan kromosom yang baik dan efektif yang dapat mewakili solusi dari permasalahan yang dihadapi.
d) *Genetic algorithm* memulai prosesnya dengan sekumpulan *initial solutions*, berbeda dengan metaheuristic lain yang memulai proses dengan sebuah solusi tunggal, dan berlanjut ke solusi lainnya melalui suatu transisi.
e) Hanya diperlukan sebuah fungsi evaluasi tunggal yang berbeda untuk tiap permasalahan.

Struktur umum *Genetic algorithm* yaitu:

a) Populasi, istilah pada teknik pencarian yang dilakukan sekaligus atas sejumlah kemungkinan solusi.
b) Kromosom, individu yang terdapat dalam satu populasi dan merupakan suatu solusi yang masih berbentuk simbol.
c) Generasi, populasi awal dibangun secara acak sedangkan populasi selanjutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi.

d) Fungsi *fitness*, alat ukur yang digunakan untuk proses evaluasi kromosom. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut.

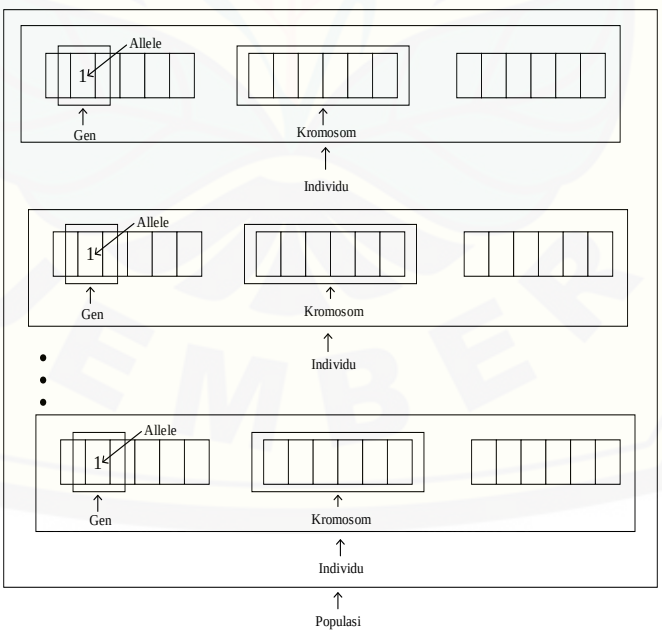
e) Generasi berikutnya dikenal dengan anak (*offspring*) yang terbentuk dari gabungan dua kromosom generasi sekarang yang bertindak sebagai induk (*parent*) dengan menggunakan operator penyilang (*crossover*).

f) Mutasi, operator untuk memodifikasi kromosom.

Ada beberapa hal yang perlu dilakukan dalam *genetic algorithm* pada saat digunakan sebagai metode penyelesaian suatu permasalahan. Beberapa hal yang perlu dilakukan tersebut adalah (Kusumaningtyas, 2016) :

1. Mendefinisikan individu.

Individu dalam hal ini menyatakan salah satu solusi (penyelesaian) yang mungkin dari permasalahan yang diangkat. Individu juga dapat dikatakan sama dengan kromosom, yang merupakan kumpulan gen. gen dapat berupa biner, float dan kombinatorial.



Gambar 2.4 Ilustrasi Penyelesaian Permasalahan Dalam *Genetic Algorithm*

Beberapa definisi penting yang perlu diperhatikan dalam mendefinisikan individu untuk membangun penyelesaian permasalahan dengan *genetic algorithm* dijelaskan pada gambar 2.4 dan keterangan berikut :

a) ***Genotype (Gen)***, sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam *genetic algorithm*, gen ini bias berupa nilai biner, float, integer maupun karakter, atau kombinatorial.
b) *Allele*, nilai dari gen.
c) Kromosom¸gabungan gen–gen yang membentuk nilai tertentu.
d) Individu, menyatakan satu nilai atau keadaan salah satu solusi yang mungkin dari permasalahan yang diangkat.
e) Populasi, merupakan sekumpulan individu yang akan diproses bersama satu siklus proses evolusi.
f) Generasi, menyatakan satu siklus proses evolusi atau satu iterasi dalam *genetic algorithm.*

2. Mendefinisikan nilai *fitness*.

Nilai *fitness* adalah nilai yang menyatakan baik tidaknya suatu solusi (individu). Nilai *fitness* ini yang dijadikan acuan dalam mencapai nilai optimal dalam *genetic algorithm*. *Genetic algorithm* bertujuan mencari individu dengan nilai *fitness* paling tinggi (Eka Risky dkk, 2012).

3. Menentukan proses pembangkitan populasi awal

Sebelum menentukan pembangkitan populasi awal, dilakukan terlebih dahulu teknik pengkodean. Teknik pengkodean adalah bagaimana mengkodekan gen dari kromosom, dimana gen merupakan bagian dari kromosom. Satu gen biasanya mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk bit, bilangan real, daftar aturan, elemen permutasi, elemen program atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

Membangkitkan populasi adalah proses membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Ukuran untuk populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal. Syarat–syarat yang harus dipenuhi untuk menunjukkan suatu solusi harus benar–benar diperhatikan dalam pembangkitan setiap individunya. Teknik pembangkitan populasi awal ini ada beberapa cara, diantaranya adalah sebagai berikut:

a) *Random generator*

   Inti dari cara ini adalah melibatkan pembangkitan bilangan *random* untuk nilai setiap gen sesuai dengan representasi kromosom yang digunakan.

b) Pendekatan tertentu (memasukkan nilai terterntu ke dalam gen)

   Cara ini adalah dengan memasukkan nilai tertentu ke dalam gen dari populasi awal yang di bentuk.

c) Permutasi gen

   Salah satu cara permutasi gen dalam pembangkitan populasi awal adalah penggunaan permutasi Josephus dalam permasalahan kombinatorial.

4. Menentukan seleksi yang akan digunakan.

Seleksi digunakan untuk memilih individu–individu mana saja yang akan dipilih untuk proses kawin silang (*crossover*) dan mutasi. Seleksi digunakan untuk mendapatkan calon induk yang baik. “Induk yang baik akan menghasilkan keturunan yang baik”. Semakin tinggi nilai *fitness* suatu individu semakin besar kemungkinannya untuk terpilih.

Langkah pertama yang dilakukan dalam seleksi ini adalah pencarian nilai *fitness*. Nilai *fitness* ini yang nantinya akan digunakan pada tahap–tahap seleksi berikutnya. Masing–masing individu dalam wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dalam wadah seleksi tersebut.

Ada beberapa definisi yang bisa digunakan untuk melakukan perbandingan terhadap beberapa metode yang akan digunakan, antara lain :

a) *Selective Pressure* : probabilitas dari individu terbaik yang akan diseleksi dibandingkan dengan rata-rata probabilitas dari semua individu yang diseleksi.
b) *Bias* : perbedaan absolut antara *fitness* ternormalisasi dari suatu individu dan probabilitas reproduksi yang diharapkan.
c) *Spread* : *range* nilai kemungkinan untuk sejumlah *offspring* dari suatu individu.
d) *Loss of diversity*: proposi dari individu-individu dalam suatu populasi yang tidak terseleksi selama fase seleksi.
e) *Selection intensity* : nilai *fitness* rata-rata yang diharapkan dalam suatu populasi setelah dilakukan seleksi (menggunakan distribusi *Gauss* ternormalisasi).
f) *Selection variance* : variansi yang diharapkan dari distribusi *fitness* dalam populasi setelah dilakukan seleksi (menggunakan distribusi *Gauss* ternormalisasi).

Terdapat beberapa metode seleksi yang dapat digunakan, diantaranya:

a) *Rank-based fitness assignment*

Populasi diurutkan menurut nilai objektifnya. Nilai *fitness* dari tiap-tiap individu hanya tergantung pada posisi individu tersebut dalam urutan, dan tidak dipengaruhi oleh nilai objektifnya.

b) *Local selection*

Setiap individu yang berada di dalam konstrain tertentu disebut dengan nama lingkungan lokal. Interaksi antar individu hanya dilakukan di dalam wilayah tersebut. Lingkungan tersebut ditetapkan sebagai struktur dimana populasi tersebut terdistribusi. Lingkungan tersebut juga dapat dipandang sebagai kelompok pasangan-pasangan yang potensial. Langkah pertama yang dilakukan adalah menyeleksi separuh pertama dari populasi yang berpasangan secara *random*. Kemudian lingkungan baru tersebut diberikan pada setiap individu yang terseleksi.

Struktur lingkungan pada seleksi lokal dapat berbentuk : linear (*full ring* dan *half ring*), dimensi-2 (*full cross* dan *half cross*, *full star* dan *half star*), dan

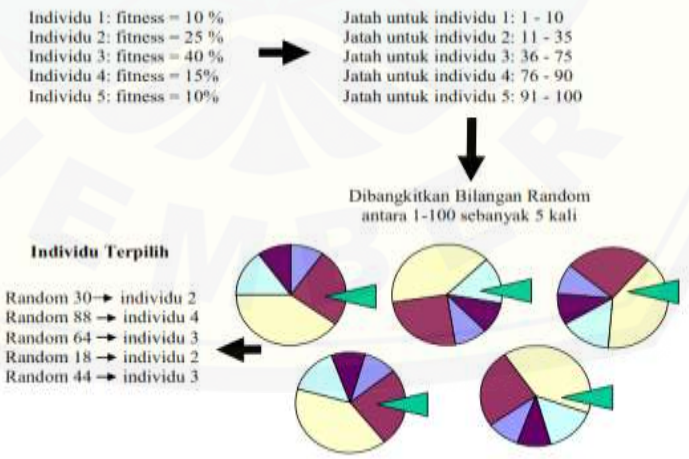
dimensi-3 dan struktur yang lebih kompleks yang merupakan kombinasi dari kedua struktur diatas. Jarak antara individu dengan struktur tersebut akan sangat menentukan ukuran lingkungan. Individu yang terdapat dalam lingkungan dengan ukuran yang lebih kecil, akan lebih terisolasi dibandingkan dengan individu yang terletak pada lingkungan dengan ukuran yang lebih besar.

c) *Truncation selection*

Merupakan seleksi buatan yang digunakan oleh populasi yang jumlahnya sangat besar. Individu - individu diurutkan berdasarkan nilai *fitness*. Hanya individu yang terbaik saja yang akan diseleksi sebagai induk. Parameter yang digunakan adalah suatu nilai ambang *trunc* yang mengindikasikan ukuran populasi yang akan diseleksi sebagai induk yang berkisar antara 50% -10%. Individu-individu yang ada dibawah nilai ambang tidak akan menghasilkan keturunan.

d) Seleksi dengan mesin *roulette*

Metode seleksi dengan mesin *roulette* ini merupakan metode paling sederhana dan sering dikenal dengan nama *stochastic sampling with replacement*.



Gambar 2.5 Ilustrasi Seleksi Dengan Mesin *Roulette*

Sumber : http://entin.lecturer.pens.ac.id/

Cara kerja metode ini adalah:

1. Dihitung nilai *fitness* dari masing–masing individu (f_i, dimana i adalah individu ke-1 s/d ke-n)
2. Dihitung total *fitness* semua individu
3. Dihitung probabilitas masing–masing individu
4. Dari probabilitas tersebut, dihitung jatah masing–masing individu pada angka 1 sampai 100
5. Dibangkitkan bilangan random antara 1 sampai 100
6. Dari bilangan *random* yang dihasilkan, ditentukan individu mana yang terpilih dalam proses seleksi.

e) Seleksi dengan *tournament*

Ditetapkan suatu nilai *tour* untuk individu - individu yang dipilih secara *random* dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk. Parameter yang digunakan adalah ukuran *tour* yang bernilai antara 2 sampai N (jumlah individu dalam populasi).

5. Menentukan perkawinan silang (*crossover*) dan mutasi gen yang akan digunakan.

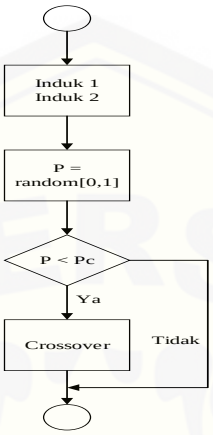
a) Perkawinan silang/pindah silang (*crossover*)

Crossover adalah operator dari *genetic algorithm* yang melibatkan dua induk untuk membentuk kromosom baru. *Crossover* menghasilkan titik baru dalam ruang perncarian yang siap untuk diuji. Operasi ini tidak selalu dilakukan pada semua individu yang ada. Individu dipilih secara acak untuk dilakukan crossing dengan probabilitas *crossover* (*Pc*) antara 0,6 hingga 0,95.

Prinsip dari *crossover* ini adalah melakukan operasi (pertukaran langsung dan pertukaran aritmatika) pada gen–gen yang bersesuaian dari dua induk untuk menghasilkan individu baru. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *crossover* yang ditentukan.

Prinsip dari *crossover* ini adalah melakukan operasi (pertukaran langsung dan pertukaran aritmatika) pada gen–gen yang bersesuaian dari dua induk untuk

menghasilkan individu baru. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *crossover* yang ditentukan.

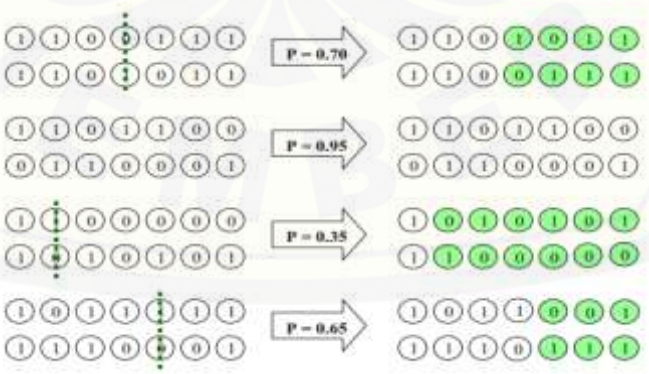


Gambar 2.6 Diagram Alir Proses *Crossover*

Ada beberapa jenis *crossover* yang dapat diaplikasikan dlam menyelesaikan suatu permasalahan, diantaranya yaitu:

1. *Crossover* satu titik

   Crossover satu titik dan banyak titik biasanya digunakan untuk representasi kromosom dalam biner. Pada *crossover* satu titik, posisi *crossover* k (k=1,2,….N-1) dengan N adalah panjang kromosom diseleksi secara *random*. Variabel – variabel ditukar antar kromosom pada titik tersebut untuk menghasilkan anak.



Gambar 2.7 Ilustrasi *Crossover* Satu Titik

Sumber : http://entin.lecturer.pens.ac.id/

2. *Crossover* banyak titik

   Pada *crossover* banyak titik, m posisi penyilangan k_i (k=1,2…….N-1, i=1,2….m) dengan N adalah panjang kromosom diseleksi secara random dan tidak diperbolehkan ada posisi yang sama, serta diurutkan naik. Variabel-variabel ditukar antar kromosom pada titik tersebut untuk menghasilkan anak.

3. *Crossover* aritmatika

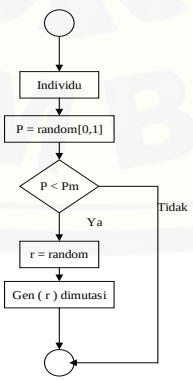
   Crossover aritmatika digunakan untuk merepresentasikan kromosom berupa bilangan *float* (pecahan). *Crossover* ini dilakukan dengan menentukan nilai r sebagai bilangan random lebih dai 0 dan kurang dari 1. Selain itu juga ditentukan posisi dari gen yang dilakukan *crossover* menggunakan bilangan random. Nilai baru dari gen pada anak mengikuti rumus (2.24) dan (2.25).

   $x_1{}'(k) = r.x_1(k) + (l - r).x_2(k)$ ..... (2.24)

   $x_2{}'(k) = r.x_2(k) + (l - r).x_1(k)$ ..... (2.25)

b) Mutasi

Operator mutasi berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak mucul pada inisialisasi populasi. Kromosom anak dimutasi dengan menambahkan nilai *random* yang sangat kecil dengan probabilitas yang rendah.



Gambar 2.8 Diagram Alir Proses Mutasi

Probabilitas mutasi (*Pm*) didefinisikan sebagai presentasi dari jumlah total gen pada populasi yang mengalami mutasi. Peluang mutasi mengendalikan banyaknya gen baru yang akan dimunculkan untuk evaluasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi. Tetapi jika peluang mutasi tertalu besar, maka akan terlalu banyak gangguan acak sehingga anak akan kehilangan kemiripan dari induknya, dan juga algoritma kehilangan kemampuan untuk belajar dari histori pencarian. Ada beberapa pendapat mengenai laju mutasi ini. Ada yang berpendapat bahwa laju mutasi sebesar 1/N akan mendapatkan hasil yang cukup baik, namun ada juga yang beranggapan bahwa laju mutasi tidak tergantung pada ukuran populasi. Kromosom hasil mutasi harus diperiksa apakah masih berada pada domain solusi, dan bila perlu bias dilakukan perbaikan. Ada beberapa jenis mutasi yang dapat digunakan dalam menyelesaikan masalah, yaitu

a) Mutasi biner
b) Mutasi bilangan real

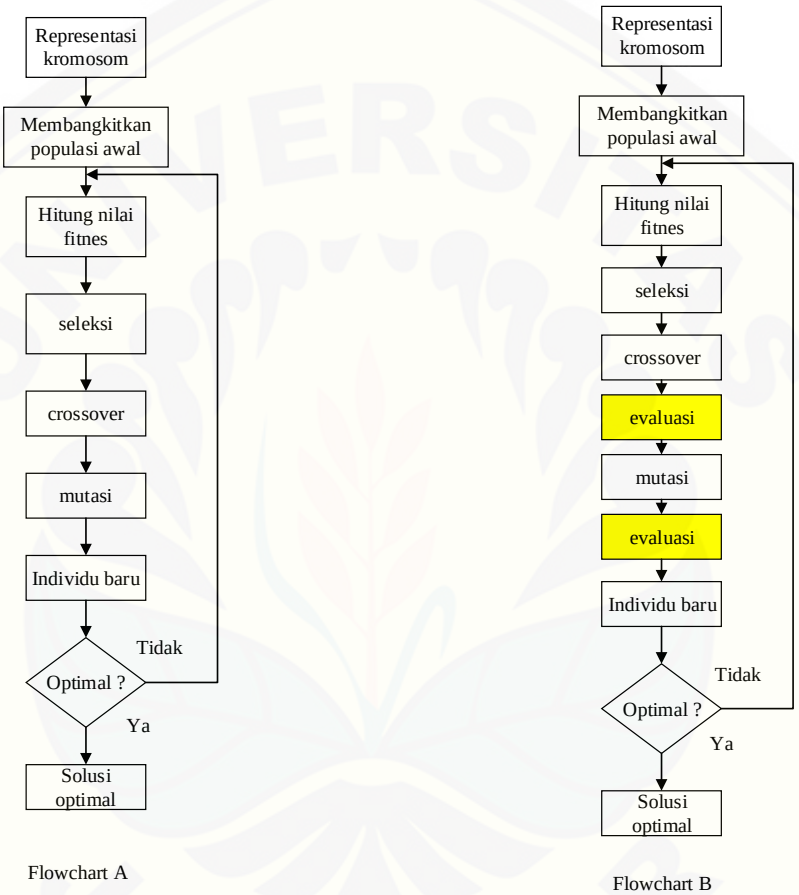
B. *Dynamic Genetic Algorithm*

Seperti yang telah disebutkan pada latar belakang, bahwa metode *genetic algorithm* konvensional memiliki keterbatasan hanya bisa melakukan persilangan dan mutasi sebanyak satu kali untuk menghasilkan generasi baru, maka ditemukanlah suatu metode baru yang disebut *dynamic genetic algorithm* (Tzung-Pei Hong, 2002).

Dynamic genetic algorithm memiliki kelebihan yang dapat menyempurnakan *genetic algorithm*, yaitu dapat melakukan persilangan dan mutasi lebih dari satu kali untuk menghasilkan generasi baru. Perbedaan yang dilakukan oleh *dynamic genetic algorithm* terletak pada evaluasi yang dilakukan setelah melakukan operator *crossover* dan mutasi.

Diharapkan dengan melakukan persilangan dan mutasi lebih dari satu kali dapat membuat kapasitas populasi semakin besar. Sehingga didapatkan variasi kromosom

yang muncul juga semakin banyak. Dengan banyaknya kromosom, maka hasil buruk dapat diminimalisir. Kapasitas populasi yang besar biasanya memberikan hasil yang lebih baik, dan mencegah terjadinya konvergensi yang prematur.



Gambar 2.9 Flowchart A. *Genetic Algorithm*, B. *Dynamic Genetic Algorithm*

Dynamic genetic algorithm secara bersamaan akan melakukan beberapa persilangan dan mutasi pada setiap proses genetik dan secara otomatis menyesuaikan rasio mereka. Dengan demikian, *dynamic genetic algorithm* dapat dianggap sebagai generalisasi dari *genetic algorithm* konvensional dengan menetapkan rasio awal

persilangan dan mutasi sebagai nilai-nilai yang ditetapkan dengan rasio awal untuk operator yang lain dianggap nol (Tzung-Pei Hong, 2002).

Dalam bahasa yang sederhana, cara kerja metode *dynamic genetic algorithm* ini dimulai dengan membentuk sebuah populasi. Kemudian populasi tersebut dievaluasi untuk menentukan individu yang memiliki kualitas yang baik. Dilanjutkan dengan memetapkan rasio awal persilangan dan mutasi. Kemudian membentuk *mating set* yang akan diambil secara acak oleh *parent.* Selanjutnya melakukan persilangan untuk menghasilkan keturunan baru yang akan dievaluasi *fitnes value* nya. Setelah rasio persilangan memenuhi, maka dilakukan mutasi dan kembali di evaluasi *fittness value*nya sesuai rasio mutasi. Akhirnya, individu–individu baru akan membentuk populasi. Populasi inilah yang akan menjadi solusi dari suatu permasalahan.

BAB 3. METODOLOGI PENELITIAN

3.1 Tempat Pelaksanaan

Proses penelitian tentang analisis karakteristik *input-output* dan optimasi biaya pembangkitan menggunakan metode *dynamic genetic algorithm* dilaksanakan di PT. PLN PJB UP Gresik. Unit pembangkit Gresik merupakan salah satu unit pembangkit tenaga listrik milik PT. PJB yang terletak di provinsi Jawa Timur. Unit pembangkit ini berlokasi di kota Gresik, kira–kira 20 km arah barat laut kota Surabaya, tepatnya di desa Sidorukun, Jl. Harun Tohir no.1 Gresik, Jawa Timur. Total luas wilayah PT. PJB UP Gresik mencapai $\pm$ 78Ha, termasuk wilayah pembuangan lumpur dan luas bangunan.

Kapasitas daya yang dimiliki PLTGU Gresik $\pm$ 600MW yang terdiri dari 4 unit dengan rincian sebagai berikut :

- Unit 1 sebesar 100 MW
- Unit 2 sebesar 100 MW
- Unit 3 sebesar 200 MW
- Unit 4 sebesar 200 MW

Sumber : PT. PLN PJB UP Gresik tahun 2012

3.2 Alat dan Bahan

Alat dan bahan yang digunakan pada penelitian ini yaitu :

- Alat

  Laptop

  Perangkat lunak berupa aplikasi MATLAB
- Bahan

  Data primer dari PT. PJB UP Gresik, data dari internet, buku dan hasil konfigurasi yang digunakan dalam pembuatan program.

3.3 Metodologi Pelaksanaan Penelitian

Dalam melakukan penelitian ini beberapa langkah harus dilakukan secara runtut. Hal ini dilakukan agar dapat menganalisa karakteristik *input–output* serta mencari solusi dari permasalahan optimasi biaya pembangkitan yang optimal. Langkah–langkah yang dilakukan yaitu :

1. Melakukan pengambilan data terbaru *input–output* pada pembangkit.
2. Membuat persamaan matematis menggunakan metode *quadratic least square regression.*
3. Membuat kurva *input–output* dari hasil metode *quadratic least square regression.*
4. Menganalisis perubahan kurva karakteristik *input–output* sehingga diketahui sejauh mana pergeseraan kurva hasil pengukuran dibandingkan dengan kurva ideal dengan menggunakan *Graphical User Interface* (GUI) Matlab.
5. Hasil karakteristik *input–output* dan data pembangkit serta pembebanan dijadikan masukan proses optimasi biaya pembangkitan menggunakan metode *dynamic genetic algorithm.*
6. Bandingkan *economic dispatch* hasil metode *dynamic genetic algoritm* dengan data pembangkitan dari pembangkit.

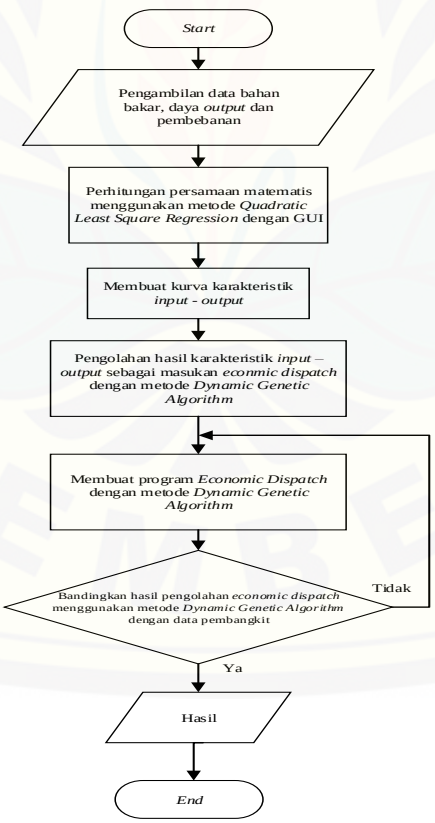
Metodologi pelaksanaan penelitian ini secara lebih rinci dijelaskan pada dua buah diagram alir pada gambar 3.1, gambar 3.2 dan gambar 3.3.

3.3.1 Diagram Alir *Economic Dispatch*

Diagram alir *economic dispatch* merupakan langkah kerja dalam menentukan optimasi pembebanan pada masing–masing pembangkit. yang dijelaskan pada gambar 3.1. Langkah–langkah dalam menyelesaikan permasalahan *economic dispatch* yaitu :

1. Melakukan pengambilan data terbaru *input–output* pada pembangkit.
2. Membuat persamaan matematis menggunakan metode *quadratic least square regression* menggunakan *Graphical User Interface* (GUI).
3. Membuat kurva *input–output* dari hasil metode *quadratic least square regression.*

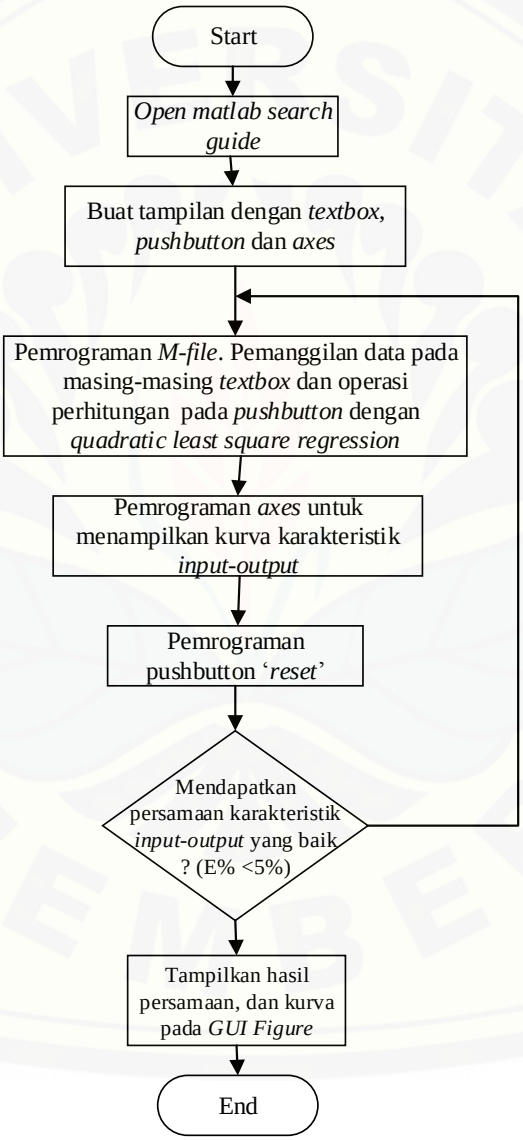
4. Menganalisis perubahan kurva karakteristik *input–output* sehingga diketahui sejauh mana pergeseraan kurva hasil pengukuran dibandingkan dengan kurva ideal.
5. Hasil karakteristik *input–output* dan data pembangkit serta pembebanan dijadikan masukan proses optimasi biaya pembangkitan menggunakan metode *dynamic genetic algorithm.*
6. Membuat program *economic dispatch* menggunakan metode *dynamic genetic algorithm.*
7. Bandingkan *economic dispatch* hasil metode *dynamic genetic algoritm* dengan data pembangkitan dari pembangkit. Jika mendapatkan hasil yang memiliki baik (lebih baik dari acuan) maka tampilkan hasil, jika tidak evaluasi program.



Gambar 3.1 Diagram *Economic Dispatch*

3.3.2 Diagram Alir *Graphical User Interface* (GUI)

Diagram alir GUI merupakan diagram yang menjelaskan langkah kerja dari perhitungan *quadratic least square regression*. Proses pembuatan GUI ditunjukkan pada gambar 3.2 berikut :



Gambar 3.2 Diagram alir GUI untuk perhitungan *quadratic least square regression*

Langkah - langkah dari pembuatan GUI yaitu :

1. Membuka aplikasi matlab dan menuliskan '*guide*' pada *command window* untuk membuka dialog GUI Matlab.
2. Kemudian membuat tampilan GUI *figure* dengan menambahkan *textbox, static text, pushbutton* dan *axes*.
3. Membuat program pada M-*file* Matlab. Program pertama mengenai pemanggilan data *input* yang dimasukkan pada *textbox* agar dapat diolah oleh M-*file*. Selanjutnya menambahkan program perhitungan karakteristik *input-output* dengan metode *quadratic least square regression*.
4. Membuat program untuk menampilkan kurva karakteristik *input-output* pada *axes*.
5. Pemrograman *pushbutton 'reset'* untuk menghapus atau membersihkan GUI *figure*.
6. Apakah hasil dari perhitungan *quadratic least square regression* mendapatkan persamaan yang baik? Persamaan yang baik adalah persamaan yang memiliki *error persen* kurang dari 5%.
7. Jika hasil persamaan sudah baik maka tampilkan hasil dan program berhenti, jika tidak maka berarti ada program yang harus diperbaiki.

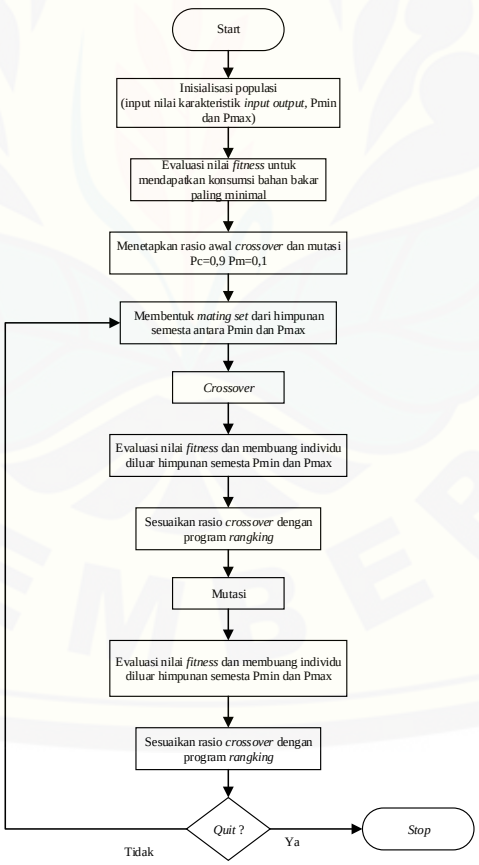
3.3.3 Diagram Alir *Dynamic Genetic Algorithm*

Metode *dynamic genetic algorithm* ini digunakan sebagai metode pendekatan algoritma yang diimplmentasikan untuk menyelesaikan masalah *economic dispatch* atau optimasi biaya pembangkitan sebagai solusi baru yang lebih baik. Metode ini dioperasikan dengan data yang dihasilkan dari perhitungan dan analisa karakteristik *input–output* yang dihitung dengan metode *quadratic least square regression*. Setelah memasukkan data karakteritik *input–output* dalam rangkaian operasi metode *dynamic genetic algorithm* ini, diharapkan dapat menghasilkan nilai *output* berupa daya *output* tiap generator, total daya dan biaya pembangkitan yang digunakan sebagai solusi permasalahan *economic dispatch* atau optimasi biaya pembangkitan yang optimal.

Prosedur metode *dynamic genetic algorithm* ditunjukkan pada gambar 3.3. Langkah-langkah dari metode *dynamic genetic algorithm* yaitu :

1. Buat set populasi awal dari N individu untuk evolusi dengan memasukkan *input* nilai koefisien karakteristik *input output* (a,b,c), Pmin dan Pmax
2. Evaluasi nilai *fitness*. Pada tahap ini, *range* antara nilai antara Pmin hingga Pmax akan dimasukkan pada persamaan karakteristik *input output*. Nilai *fitness* yang baik direpresentasikan pada nilai konsumsi bahan bakar paling minimal.
3. Menetapkan rasio awal persilangan dan mutasi. Pada tahap ini akan ditentukan nilai n, Pc, dan Pm. n adalah matrik yang dibentuk dari masukan a,b,c, Pmin dan Pmax. Pc bernilai 0,9 dan Pm bernilai 0,1.
4. Membentuk *mating set*. *Sample* orang tua yang akan dikembangbiakkan akan diambil secara acak dari evaluasi kebugaran individu pada langkah 2. Orang tua diambil secara acak dari *range* Pmin dan Pmax.
5. Terapkan operator *crossover* untuk menghasilkan keturunan. Dari orang tua yang telah dipilih dari langkah 4, maka *sample-sample* ini kan disilangkan secara acak untuk menghasilkan keturunan baru. *Crossover* dilakukan dengan menkorversi Pmin dan Pmax menjadi bilangan biner kemudian akan di *random* oleh program.
6. Evaluasi nilai *fitness* dari setiap individu yang dihasilkan oleh masing-masing operator *crossover*. Evaluasi ini juga akan membuang nilai-nilai yang tidak dalam himpunan semesta Pmin hingga Pmax.
7. Sesuaikan rasio *crossover* menurut rata-rata nilai progres yang didapatkan. Pada tahap ini, terdapat program *rangking* untuk mengurutkan keturunan dengan nilai *fitness* paling baik.
8. Terapkan operator mutasi untuk menghasilkan keturunan. Pada tahap ini, program akan menyisipkan biner secara acak pada keturunan yang memiliki nilai mendekati solusi.

9. Evaluasi nilai *fitness* dari setiap individu dihasilkan oleh operator mutasi. Evaluasi ini juga akan membuang nilai-nilai yang tidak dalam himpunan semesta Pmin hingga Pmax.
10. Sesuaikan rasio *crossover* menurut rata-rata nilai *progress* yang didapatkan. Pada tahap ini, terdapat program *rangking* untuk mengurutkan keturunan dengan nilai *fitness* paling baik
11. Didapatkan populasi baru (solusi). Solusi yang didapatkan belum tentu sebagai solusi paling baik. Jika kurang baik, maka akan dilakukan proses kembali yang dimulai dari langkah 4. Parameter baik yang digunakan adalah apabila individu dengan nilai fitness paling bagus (nilai konsumsi bahan bakar paling rendah).



Gambar 3.3 Diagram *Dynamic Genetic Algorithm*

BAB 5. PENUTUP

5.1 Kesimpulan

Dari penelitian yang telah dilakukan pada tugas akhir ini, dapat diambil beberapa kesimpulan, yaitu :

1. Karakteristik *input-output* pada saat menggunakan bahan bakar *natural* gas, menunjukkan bahwa performa pembangkit masih baik setelah 34 tahun beroperasi.
2. Perbandingan karakteristik *input-output* saat menggunkan bahan bakar *liquid*, menunjukkan bahwa PLTU 1 masih memiliki performa yang baik, sedangkan PLTU 2 telah mengalami pergeseran.
3. Dengan menggunakan metode *dynamic genetic algorithm* yang digunakan pada pembangkit menggunakan bahan bakar gas, didapatkan didapatkan pembebanan yang optimal yang menghasilkan total biaya bahan bakar dapat dihemat sebesar 3.162,9147 KNM^3 dan biaya bahan bakar dapat dihemat sebesar $22.773 dibandingkan PJB.
4. Dengan menggunakan metode *dynamic genetic algorithm* yang digunakan pada pembangkit menggunakan bahan bakar *liquid*, didapatkan pembebanan yang optimal yang menghasilkan total biaya bahan bakar dapat dihemat sebesar 16.532,2189 liter dan biaya bahan bakar dapat dihemat sebesar Rp. 1.131.369.852 dibandingkan PJB
5. Perbandingan metode *dynamic genetic algorithm* dengan metode PSO dari segi pembebanan, maka metode *dynamic genetic algorithm* dapat membagi beban sesuai acuan, sementara PSO memiliki selisih -1 hingga 42 MW. *Dynamic genetic algorithm* memiliki konsumsi bahan bakar 245,24 liter dan biaya bahan bakar Rp. 1.477.569 lebih hemat dibanding metode PSO.
6. Perbandingan metode *dynamic genetic algorithm* dengan metode SA dari segi pembebanan, maka metode *dynamic genetic algorithm* dapat membagi beban sesuai acuan, sementara SA memiliki selisih -0,0001 hingga 0,0001 MW. *Dynamic*

genetic algorithm memiliki konsumsi bahan bakar 50.511,12 liter dan biaya bahan bakar Rp. 304.329.523 lebih hemat dibanding metode SA.

5.2 Saran

Dari penelitian yang telah dilakukan pada tugas akhir ini, dapat diambil beberapa saran, yaitu :

1. Pada penelitian ini, perbandingan konsumsi bahan bakar pada PLTU hanya dilakukan pada bahan bakar HSD, untuk penelitian selanjutnya dapat membandingkan konsumsi bahan bakar dengan bahan bakar lain yang digunakan oleh PLTU.
2. Dalam tugas akhir ini, rugi-rugi tidak diperhitungkan, untuk penelitian selanjutnya rugi-rugi bisa diperhitungkan.

DAFTAR PUSTAKA

Agustin, Neny. 2014. Optimasi biaya pembangkitan pada PT. PJB UP Gresik menggunakan *Economic Dispatch* (ED) dengan metode *Simulated Annealing* (SA). Jember: Skripsi. Universitas Jember.

Ardiana, Bangkit Basovi. 2014. Analisis Karakteristik *Input-Output* dan *Economic Dispatch* Menggunakan Metode *Particle Swarm Optimization* (PSO) Pada PT. PLN PJB UP Gresik. Jember: Skripsi. Universitas Jember.

Basuki, Cahyo Adi, Ir. Agung Nugroho & Ir. Bambang Winardi. 2011. Analisis Konsumsi Bahan Bakar Pada Pembangkit Listrik Tenaga Uap Dengan Menggunakan Metode *Least Square.* Semarang : Universitas Diponegoro.

Bisilsin, Franki Yusuf, Yeni Herdiyeni & Bib Paruhum Silalahi. 2014. Optimasi *K-Means Clustering* Menggunakan *Particle Swarm Optimization* pada Sistem Identifikasi Tumbuhan Obat Berbasis Citra. Bogor : Institut Pertanian Bogor.

FERC Staff. 2005. *Economic Dispatch : Concepts, Practices and Issues*. California: Palm Springs.

Firmansyah, Eka Risky, Syukri Sayyid Ahmad & Nurul Hikmah Agustin. 2012. Algoritma Genetika. Jakarta : Universitas Islam Negeri Syarif Hidayatullah Jakarta.

Glover, J. Duncan, Mulukutla S. Sarma, & Thomas J. Overbye. 2010. *Power System Analysis and Design*. USA: Global Engineering.

Joshi, Gopesh. 2014. *Review of Genetic Algorithm: An optimization Technique. International Journal of Advanced Research in Compuer Science and Software Engineering. Volume 4, Issue 4, April 2014*

Kusumaningtyas, E. M. (2016, June 5). *Buku Kecerdasan Buatan*. Retrieved June 5, 2016, from pens.ac.id: entin.lecturer.pens.ac.id

Ouiddir, R, M. Rahli & L. Abdelhakem-Koridak. 2005. *Economic Dispatch using a Genetic Algorithm: Aplication to Western Algeria's Electrical Power Network. Journal of Information Science and Engineering University of Oran.*

Sivanandam, S.N & S.N Depa. 2007. *Introduction to Genetic Algorithm*. Jerman: Springer Science & Business Media

Syah, Khairudin, Harry Soekotjo Dachlan & Mahfudz Shidiq. 2013. *Econmic Dispatch* Pembangkit Menggunakan Metode *Coonstriction Factor Particle Swarm Optimization* (CPSO). Malang: Universitas Brawijaya

Thiang & Dhany Indrawan. 2008. Implementasi Metode *Simulated Annealing* pada Robot Mobil untuk Mencari Rute Terpendek. Surabaya : Universitas Kristen Petra.

Tzung-Pei Hong. 2002. *Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process*. *Journal of Departmen Information Management I-Shou Universitty & Institute of Electrical Engineering Chung-Hua University*.

LAMPIRAN

A. Data Hubungan Daya Dan Bahan Bakar Periode Pembangkit Menggunakan Bahan Bakar Gas

Date	Time	PLTU 1		PLTU 2		PLTU 3		PLTU 4	
		Load	Gas Flow	Load	Gas Flow	Load	Gas Flow	Load	Gas Flow
1-May-15	4:00	45	12.23	45	11.54	108	23.64	107	25.5
	18:00	45	12.23	45	12.05				
2-May-15	6:00	45	12.14	45	11.9	108	24.16	107	25.5
3-May-15	6:00	45	12.23	45	11.64	108	24.05	107	25.4
4-May-15	6:00	45	12.26	42	12.1	108	23.51	107	25.5
	20:00	45	12.27	45	12.02	109	24.35	107	25.6
5-May-15	4:00	45	12.37	45	12.16	108	24.05	107	25.2
6-May-15	0:00	45	12.8	45	12.16			107	25.6
	14:00	45	12.73	45	12.32	108	23.43		
	18:00	45	12.02	45	11.84	108	23.83	90	23
7-May-15	4:00	45	12.3	45	12.04	108	24.67	102	24.9
	8:00	45	12.65	45	12.11	108	24.02	80	21.2
	10:00	45	12.65	45	12.48	108	24.01		
	22:00	45	12.02	45	10.87	108	23.23	110	27.2
8-May-15	2:00	46	12.58	45	11.09	108	24.21	110	26.5
	16:00	45	12.91	45	11.89			110	26.5
9-May-15	6:00	46	13.12	45	12.01	108	23.7	110	27.3
10-May-15	2:00	46	12.82	45	11.95			110	27.1
11-May-15	14:00	46.4	12.73	45	11.89	109.3	24.169	110	26.71
12-May-15	2:00	46	12.38	45	11.8				
	8:00	46	13.59	45	12.23			160	38
	14:00	46	12.69	45	12.29	108	24.21	160	38
13-May-15	2:00	46	13.23	45	12.01	108	23.6		
	10:00	45	13.24	44	12.25	108	23.78	110	27.4
	16:00	45	12.48	45	11.78	108	24.33	160	38
14-May-15	18:00	67	17.46	67	17.23	108	23.247		
15-May-15	8:00	45	12.4	45	11.44	108	23.6	110	26.1
16-May-15	0:00	46	12.37			108	23.1		
	4:00	46	12.78			108	24.339	160	36.5

	14:00	45	12.45					170	39.33
	16:00	46	12.63					110	26.47
	18:00	46	12.49					170	39.35
	20:00	46	12.59						
17-May-15	4:00	45	12.27			108	24.21		
	6:00	45	12.52					110	26.3
	8:00	46	12.51			108	23.78	110	26.25
18-May-15	4:00	45	12.43			108	24.35	160	35.87
	6:00	45	12.9			108	23.11		
	8:00			46	12.67	109.6	24.373		
19-May-15	8:00	45	12.97			108	24.21		
	14:00	85	22.15			108	23.79	107	24.88
	18:00	84	21.64.			108	23.56		
20-May-15	6:00	46	12.55			108	23.83		
	12:00	45	12.95			108	23.12	107	24.77
21-May-15	8:00	45	12.56					175	38.8
	10:00	85	22.94			108	23.407	108	24.8
	12:00	45	12.21			108	23.439		
22-May-15	6:00	46	12.55			108	23.51	175	39.9
	12:00	84	22.41			108	23.83	175	40.2
	16:00	66	17.31			108	24.37		
23-May-15	0:00	45	12.26			108	23.107	108	25.4
	2:00	45	12.4			108	24.4	175	39.1
	6:00	45	12.72					175	39.9
24-May-15	6:00	45	12.38			108	23.56	175	39.9
25-May-15	0:00	45	12.38			108	23.14	107	25.2
	4:00	45	12.69			108	24.611	107	25.8
	8:00	45	12.38			108	24.46	160	37.6
	12:00	85	23.27					107	26.29
	14:00	85	23					160	37.55
26-May-15	6:00	45	12.69					107	25.8
	8:00	85	22.91			108	23.82	107	25.8
	16:00	85	22.65					160	37.2
	18:00	85	22.58			108	23.51	160	37.4
27-May-15	8:00	67	22.42			109.1	24.645	107	24.97
28-May-15	10:00	85	23			108	24.679	176	40
	8:00	45	12.94			108	24.21	108	25.6

	10:00	45	12.67			108	24.21	175	40.2
	14:00	85	23.89			108	23.83	175	39.6
29-May-15	12:00	45	12.78			108	23.16	175	39.9
	14:00	84	23.34			109.11	24.713	175	39.9
30-May-15	12:00	85	22.98			108	23.695	108	25.7
	14:00	85	23.65			108	24.21		
31-May-15	8:00	45	12.53			108	23.43		25.3
1-Jun-15	4:00	45	12.32	0	0	108	24.67		
2-Jun-15	6:00	45	12.38	0	0	108	23.56	160	36.3
3-Jun-15	8:00	45	12.61	0	0	108	24.7	107	25.2
	20:00	45	12.23	0	0	108	23.14	107	25.4
4-Jun-15	6:00	46	13.12	0	0	108	23.94	107	25.9
	18:00	85	22.67	0	0	108	23.14	107	25.5
5-Jun-15	0:00	45	12.51	0	0			107	25.3
	18:00	46	12.88	0	0	108	23.27	107	25.2
6-Jun-15	8:00	45	12.75	0	0	160	33.87	160	36.6
	12:00	45	12.6	0	0	170	35.77	175	40.01
7-Jun-15	6:00	45	13.06	0	0	160	33.85	160	37.1
8-Jun-15	6:00	45	12.28	0	0	160	33.91	107	25.4
	20:00	79	20.56	0	0			107	25.46
	22:00	61	16.21	0	0				
9-Jun-15	12:00	81	22.41	0	0	108	23.65	107	25.8
	14:00	78	20.33	0	0	108	23.91	107	26.1
	16:00	80	21.07	0	0	108	33.79	107	26.1
10-Jun-15	10:00	85	22.72	0	0	108	23.61	107	25.5
11-Jun-15	14:00	84	22.73	0	0	108	23.59	107	28.18
	16:00	84	22.49	0	0	160	33.93	160	36.81
	18:00	84	23.01	0	0	170	35.87	175	40.2
12-Jun-15	6:00	45	12.37	0	0	170	35.28	175	40.6
	8:00	45	12.39	0	0	108	22.87	108	25.4
	16:00	85	23.4	0	0	170	36	170	39.2
13-Jun-15	2:00	46	12.64	0	0	170	35.6	175	39.9
	20:00	85	22.53	0	0	170	35.77	175	39.6
14-Jun-15	4:00	46	12.35	0	0	170	36.08	175	40.5
	12:00	45	12.94	0	0	108	23.03	107	25.28
16-Jun-15	2:00	45	12.34	0	0	170	35.52	175	40.1
	18:00	45	12.7	0	0	108	23.05	108	25.4

17-Jun-15	8:00	45	12.42	0	0	160	33.28	175	39.6
	10:00	45	12.18	0	0	170	35.46	175	39.5
18-Jun-15	8:00	45	12.79	0	0	108	23.85	107	25.5
	10:00	45	13	0	0	160	33.41	160	36.6
19-Jun-15	8:00	45	12.4	0	0	160	33.98	160	36.8
	10:00	85	22.7	0	0	160	33.82	160	36.3
20-Jun-15	6:00	45	12.64	0	0	140	31.11	107	26.3
	8:00	45	12.43	0	0	114	24.09	107	26.1
	10:00	45	13	0	0	160	34.19	107	26.3
21-Jun-15	6:00	45	12.92	0	0	108	23.12	107	25.5
	8:00	46	12.74	0	0	160	33.51	160	37.2
22-Jun-15	8:00	45	12.35	45	12.09	108	23.33	107	25.57
	10:00	46	12.65	45	12.52	108	23.32	107	25.87
23-Jun-15	18:00	85	22.44	84	20.84	170	35.72	160	36.97
	22:00	45	12.09	46	11.65	170	35.68	175	40.42
24-Jun-15	0:00	45	12.7	45	12.32	108	23.11	107	25.58
25-Jun-15	12:00	46	13.12	45	12.32	108	23.06	107	25.5
26-Jun-15	14:00	46	12.45	45	12.35	170	35.55	107	25.12
	22:00	45	12.3	45	12.32	170	35.56	107	25.75
27-Jun-15	6:00	45	12.78	45	12.32	170	35.7	107	25.2
28-Jun-15	8:00	45	12.86	45	12.56	108	22.85	108	25.5
29-Jun-15	10:00	45	12.13	85	21.64	108	22.85	107	25.2
30-Jun-15	18:00	45	12.22	45	12.32	108	23.54	108	24.8
1-Jul-15	0:00	45	12.45	45	12.12	108	23.2	108	24.7
2-Jul-15	8:00	45	12.27	45	12.32	108	23.75	107	24.8
3-Jul-15	16:00	46	12.59	45	12.15	108	23.2	107	24.59
4-Jul-15	2:00	45	12.86	45	12.44	108	23.06	107	24.5
5-Jul-15	8:00	45	12.95	45	12.53			107	24.85
6-Jul-15	10:00	45	13.36	45	12.53	108	23.49	107	24.63
7-Jul-15	12:00	45	12.62	45	12.32	108	23.23	107	24.7
8-Jul-15	12:00	45	12.71	45	12.48	108	22.29	107	24.6
9-Jul-15	16:00	45	12.23	45	12.52	108	23.22	107	24.68
10-Jul-15	0:00	45	12.68	45	12.24	108	22.86	107	24.9
11-Jul-15	2:00	45	12.69	45	12.3	105	22.62	102	24.11
	20:00	85	22.45	85	21.93	105	22.62	107	4.09
12-Jul-15	8:00	45	12.93	45	12.32	108	22.66	107	25.42
	20:00	45	12.17	45	12.19	170	35.68	175	39.8

13-Jul-15	20:00	45	12.35	45	12.19	170	35.11	175	40.27
14-Jul-15	8:00	45	12.21	45	12.64	108	22.19	90	22.3
15-Jul-15	0:00	45	12.71	45	12.06	106	22.79	107	24.94
	18:00	45	12.65	45	12.45	107	23.01	107	
16-Jul-15	20:00	45	12.93	45	12.33	160	32.98	160	36.06
17-Jul-15	10:00	45	12	45	11.05	108	22.52	107	25.05
18-Jul-15	12:00	45	12.21	45	12.01	108	23.27	175	40
19-Jul-15	12:00	45	12.63	45	11.92	108	23.06	107	25.3
20-Jul-15	0:00	45	12.05	45	11.56	141	28.58	160	34.4
	20:00	45	12.4	45	11.57	134	28.05	175	39.4
21-Jul-15	12:00	45	12.73	45	12.52	108	23.16	107	24.93
22-Jul-15	20:00	65	17.73	65	16.61			155	24.13
	22:00	45	12.81	45	12.59			175	39.33
23-Jul-15	18:00	45	12.86	45	11.65			107	24.75
	20:00	85	22.45	85	21.92			175	38.87
24-Jul-15	14:00	45	12.23	45	11.64			107	24.62
	18:00	67	17.54	67	17.09			107	23.99
25-Jul-15	10:00	45	12.39	45	11.74	108	26.4	107	24.1
26-Jul-15	10:00	45	12.72	45	11.66	107	26	107	24.8
27-Jul-15	6:00	45	11.91	45	11.29	107	25.3	107	24.7
	20:00	45	12.08	46	12.48	169	39.08	160	36.73
28-Jul-15	6:00	45	12.05	45	11.55			107	25.03
29-Jul-15	8:00	46	12.51	46	11.36			107	25.1
30-Jul-15	10:00	46	12.61	45	11.55				
31-Jul-15	16:00	46	12.65	45	12.18			107	25.29

B. Listing Program Graphical User Interface untuk Metode Quadratic Least Square Regression

```
function varargout = skripsi_qslr(varargin)
% SKRIPSI_QSLR M-file for skripsi_qslr.fig
%      SKRIPSI_QSLR, by itself, creates a new SKRIPSI_QSLR or raises the existing
%      singleton*.
%
%      H = SKRIPSI_QSLR returns the handle to a new SKRIPSI_QSLR or the handle to
%      the existing singleton*.
%
%      SKRIPSI_QSLR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SKRIPSI_QSLR.M with the given input arguments.
%
%      SKRIPSI_QSLR('Property','Value',...) creates a new SKRIPSI_QSLR or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before skripsi_qslr_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to skripsi_qslr_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help skripsi_qslr

% Last Modified by GUIDE v2.5 15-May-2016 20:37:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @skripsi_qslr_OpeningFcn, ...
                   'gui_OutputFcn',  @skripsi_qslr_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
  % --- Executes just before skripsi_qslr is made visible.
function skripsi_qslr_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to skripsi_qslr (see VARARGIN)

% Choose default command line output for skripsi_qslr
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes skripsi_qslr wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = skripsi_qslr_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr1_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr1 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr1 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr2_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr2 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr2 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr3_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr3 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr3 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function bahanbkr3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function bahanbkr4_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr4 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr4 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr5_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr5 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr5 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr6_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr6 as text
```

```
%        str2double(get(hObject,'String')) returns contents of bahanbkr6 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr7_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr7 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr7 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr8_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr8 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr8 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double

function bahanbkr9_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr9 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr9 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr10_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr10 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr10 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr11_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr11 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr11 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr12_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr12 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr12 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a double

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr13_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr13 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr13 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a double

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr14_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr14 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr14 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a double

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr15_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr15 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr15 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16 as a double

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr16_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr16 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr16 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a double

% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr17_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr17 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr17 as a
double


% --- Executes during object creation, after setting all properties.
function bahanbkr17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18 as a double

% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bahanbkr18_Callback(hObject, eventdata, handles)
% hObject    handle to bahanbkr18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bahanbkr18 as text
%        str2double(get(hObject,'String')) returns contents of bahanbkr18 as a
double

% --- Executes during object creation, after setting all properties.
function bahanbkr18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bahanbkr18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit40_Callback(hObject, eventdata, handles)
% hObject    handle to edit40 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit40 as text
%        str2double(get(hObject,'String')) returns contents of edit40 as a double


% --- Executes during object creation, after setting all properties.
function edit40_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit40 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function hasil1_Callback(hObject, eventdata, handles)
% hObject    handle to hasil1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hasil1 as text
%        str2double(get(hObject,'String')) returns contents of hasil1 as a double

% --- Executes during object creation, after setting all properties.
function hasil1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hasil1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function hasil2_Callback(hObject, eventdata, handles)
% hObject    handle to hasil2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hasil2 as text
%        str2double(get(hObject,'String')) returns contents of hasil2 as a double

% --- Executes during object creation, after setting all properties.
function hasil2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hasil2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
--- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x = [...
    str2num(get(handles.edit1,'String'))
    str2num(get(handles.edit2,'String'))
    str2num(get(handles.edit3,'String'))
    str2num(get(handles.edit4,'String'))
    str2num(get(handles.edit5,'String'))
    str2num(get(handles.edit6,'String'))
    str2num(get(handles.edit7,'String'))
    str2num(get(handles.edit8,'String'))
    str2num(get(handles.edit9,'String'))
    str2num(get(handles.edit10,'String'))
    str2num(get(handles.edit11,'String'))
    str2num(get(handles.edit12,'String'))
    str2num(get(handles.edit13,'String'))
    str2num(get(handles.edit14,'String'))
    str2num(get(handles.edit15,'String'))
    str2num(get(handles.edit16,'String'))
    str2num(get(handles.edit17,'String'))
    str2num(get(handles.edit18,'String'))];

y = [...
    str2num(get(handles.bahanbkr1,'String'))
    str2num(get(handles.bahanbkr2,'String'))
    str2num(get(handles.bahanbkr3,'String'))
    str2num(get(handles.bahanbkr4,'String'))
    str2num(get(handles.bahanbkr5,'String'))
    str2num(get(handles.bahanbkr6,'String'))
    str2num(get(handles.bahanbkr7,'String'))
    str2num(get(handles.bahanbkr8,'String'))
    str2num(get(handles.bahanbkr9,'String'))
    str2num(get(handles.bahanbkr10,'String'))
    str2num(get(handles.bahanbkr11,'String'))
    str2num(get(handles.bahanbkr12,'String'))
    str2num(get(handles.bahanbkr13,'String'))
    str2num(get(handles.bahanbkr14,'String'))
    str2num(get(handles.bahanbkr15,'String'))
    str2num(get(handles.bahanbkr16,'String'))
    str2num(get(handles.bahanbkr17,'String'))
    str2num(get(handles.bahanbkr18,'String'))];

N = [...
    str2num(get(handles.edit40,'String'))];

%% Sx dan Sy
Sx=sum(x);
Sy=sum(y);

%% Sx^2
for i=1:N
    a(i)=x(i)*x(i);
    b(i)=x(i)*y(i);
```

```
    c(i)=x(i)*x(i)*x(i);
    d(i)=x(i)*x(i)*x(i)*x(i);
    e(i)=x(i)*x(i)*y(i);
end

 %% Mencari Sxx Sxy dll
 Sxx=Sx2-(Sx^2/N);
 Sxy=Sxy-(Sx*Sy/N);
 Sxx2=Sx3-(Sx*Sx2/N);
 Sx2y=Sx2y-(Sx2*Sy/N);
 Sx2x2=Sx4-(Sx2*Sx2/N);

 %% Koefisien a, b, dan c
 a=((Sx2y*Sxx)-(Sxy*Sxx2))/((Sxx*Sx2x2)-Sxx2*Sxx2);
 b=((Sxy*Sx2x2)-(Sx2y*Sxx2))/((Sxx*Sx2x2)-Sxx2*Sxx2);
 c=Sy/N - (b*(Sx/N))-(a*(Sx2/N));

 fprintf('Persamaannya QSLR : %.4f x^2 + %.4f x + %.4f\n', a,b,c);
 set(handles.hasil1,'String',a)
 set(handles.hasil2,'String',b)
 set(handles.hasil3,'String',c)

 %% Ploting
axes(handles.axes1)

%Data
%figure(1)
hold on
h1=scatter(x , y , 50,'r','filled');
hold off


%Hasil Regresi
p=[a b c];
x=linspace(0,700);
y=polyval(p,x);
plot(x,y)
xlabel('Bahan Bakar (lt/jam)');
ylabel('Daya (MW)');


%Tombol RESET
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.hasil1,'string',num2str(0))
set(handles.hasil2,'string',num2str(0))
set(handles.hasil3,'string',num2str(0))

cla(handles.axes1,'reset')
```

C. Listing Program Dynamic Genetic Algorithm

```
disp('                                                                    ')
disp('____________________________________________________________________')
disp('                                                                    ')
disp('                          SKRIPSI                                   ')
disp('                PERHITUNGAN ECONOMIC DISPATH                        ')
disp('          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM              ')
disp('             Rina Anggraeni          (121910201110)                 ')
disp('____________________________________________________________________')
disp('                                                                    ')

% Kolom 1,2,3 berisi koefisien karakteristik input output pembangkit
% Kolom 4,5 berisi limit daya pembangkit
% Kolom 1 = koefisien a (ax^2) karakteristik input output pembangkit
% Kolom 2 = koefisien b (bx) karakteristik input output pembangkit
% Kolom 3 = koefisien c (c) karakteristik input output pembangkit
% Kolom 4 = daya minimal (MW)
% Kolom 5 = daya maksimal (MW)

data=[...
0.0057   0.5038    264.4896   45   45
0.0707   -7.9486   516.9360   45   45
-0.0005  1.6189    399.152    110  110
0.0266   -5.6917   905.7788   110  18];
n=4;
lb=data(:,4)'; % jumlah baris
ub=data(:,5)'; % jumlah kolom
 gaDat.FieldD=[lb; ub];
 gaDat.Objfun='ELDCost';

% Execute GA
gaDat=ga(gaDat);
% Hasil

x1=gaDat.xmin;
[F  P1]=ELDCost(x1)
% F = harga minimal
% P1 = pembebanan pada masing-masing pembangkit
% lam the deviation os sum of genration minus demend+losses i.e sum(P1)-Pd-
P1*Bmn*P1'

% Matrik Rugi - Rugi = diabaikan
B=0*[...
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0];

%Pd = Jumlah beban
Pd=340;

n=length(data(:,1));
[m n1]=size(x);
P=x(1:m,2:n);
B11=B(1,1);
B1n=B(1,2:n);
Bnn=B(2:n,2:n);
A=B11;
BB1=2*B1n*P';
B1=(BB1-1)';
C1=(P*Bnn*P');
C1=diag(C1);
```

```
C=Pd-(sum(P'))'+C1;
 A=A*ones(m,1);
for i=1:m
    y=[A(i) B1(i) C(i)];
x1(i,:)=roots(y);
 x2(i)=(abs(min(x1(i,:))))';
 if x2(i)>data(1,5)
     x2(i)=data(1,5);
 else
 end
   if x2(i)<data(1,4)
x2(i)=data(1,4);
   else
   end
end
 P1=[x2' P];
 a1=data(:,1);
 b1=data(:,2);
 c1=sum(data(:,3));
 F=P1.*P1*a1+P1*b1+c1;
 Pl1=0;
 Pl=0;
 lam=abs(sum(P1')'-Pd-Pl);
 F=(F)+1000*lam;
 x=P1;
 end

function gaDat=ga(g)
%
% Dynamic Genetic Algorithm
%    gaDat=ga(gaDat)
%    gaDat : Struktur Data Algoritma
%
% gaDat.FieldD=[lb; ub]; % (lb) jumlah baris data matrik
                         % (up) jumlah kolom data matrik
%
% gaDat.Objfun='costFunction'; % Name of the Objective function to be minimize
%
% Parameter standar Dynamic Genetic Algorithm:
% gaDat.MAXGEN={gaDat.NVAR*20+10}; % Jumlah generasi
% gaDat.NIND={gaDat.NVAR*50} ;     % Ukuran populasi
% gaDat.alfa={0};                  % Parameter untuk linear crossover
% gaDat.Pc={0.9};                  % Probabilitas Crossover
% gaDat.Pm={0.1};                  % Probabilitas Mutasi
% gaDat.ObjfunPar={[]};            % Parameter tambahan fungsi objektif
% gaDat.indini={[]};               % Inisialisasi anggota dari inisial populasi
%


% ---------------------- Parameter Internal -----------------------------
gaDat.Chrom=[];
gaDat.ObjV=[];
gaDat.xmin=[];
gaDat.fxmin=inf;
gaDat.xmingen=[];
gaDat.fxmingen=[];
gaDat.rf=(1:gaDat.NIND)';
gaDat.gen=0;

%% ---------------------------- MAIN LOOP ---------------------------------
%% ------------------- Penghitungan Jumlah Generasi ----------------------
gen=0;

%% -------------------- Inisialisasi Populasi ----------------------------
```

```
gaDat.Chrom=crtrp(gaDat.NIND,gaDat.FieldD);   % Real codification

% Individu dari gaDat.indini ditambahkan acak dari inisialisasi populasi
if not(isempty(gaDat.indini))
    nind0=size(gaDat.indini,1);
    posicion0=ceil(rand(1,nind0)*gaDat.NIND);
    gaDat.Chrom(posicion0,:)=gaDat.indini;
end

while (gaDat.gen<gaDat.MAXGEN),
    gaDat.gen=gen;
    gaDat=gaevolucion(gaDat);
    % Penurunan penghitungan generasi
    gaDat.xmingen(gen+1,:)=gaDat.xmin;
    gaDat.fxmingen(gen+1,:)=gaDat.fxmin;
    gen=gen+1;
end
%% ------------------------ END MAIN LOOP --------------------------------

%% --------------------------- Hasil -------------------------------------
disp('_____________________________________________________________')
disp('                                                             ')
disp('                          HASIL                              ')
disp('                                                             ')
disp('_____________________________________________________________')
disp('-------------------------------------------------------------')
disp('                                                             ')
disp(['   Fungsi Objektif untuk Pmin : ' num2str(gaDat.fxmin)])
disp('                                                             ')
disp('-------------------------------------------------------------')
disp('                                                             ')
disp('_____________________________________________________________')
disp('                                                            !')
disp('                  PEMBAGIAN PEMBEBANAN                      !')
disp('                                                            !')
disp('------------------------------------------------------------!')
disp(['   Pmin : ' mat2str(gaDat.xmin)])
disp('_____________________________________________________________')


%% ---------------------- GENERASI PERTAMA -----------------------------
function gaDat=gaevolucion(gaDat)
Chrom=gaDat.Chrom;
nind=size(Chrom,1);
ObjV=inf(nind,1);
for i=1:nind
    if isempty(gaDat.ObjfunPar)
        ObjV(i)=feval(gaDat.Objfun,Chrom(i,:));
    else
        ObjV(i)=feval(gaDat.Objfun,Chrom(i,:),gaDat.ObjfunPar);
    end
end
gaDat.ObjV=ObjV;

%% ---------------- Individu Terbaik dari Generasi ---------------------
[v,p]=min(gaDat.ObjV);
if v<=gaDat.fxmin
    gaDat.xmin=Chrom(p,:);
    gaDat.fxmin=v;
end

%% -------------------- GENERASI SELANJUTNYA -----------------------------
%% ------------------------ RANKING --------------------------------------
FitnV = ranking(gaDat.ObjV,gaDat.rf);
```

```
%% ------------------------ SELEKSI ----------------------------------
SelCh = select('sus',Chrom,FitnV,1);

%% ------------------------ CROSSOVER ----------------------------------
% Crossover menggunakan uniform crossover
SelCh = lxov(SelCh,gaDat.Pc,gaDat.alfa);

%% ------------------------ MUTATION ----------------------------------
Chrom = mutbga(SelCh,gaDat.FieldD,[gaDat.Pm 1]); % Codificación Real.
% Reinsert the best individual
Chrom(round(gaDat.NIND/2),:) = gaDat.xmin;
gaDat.Chrom=Chrom;
disp('_____________________________________________________________')
disp(['Iterasi ke-: ' num2str(gaDat.gen)])
disp(['   Pmin: ' mat2str(gaDat.xmin) ' -- f(Pmin): ',num2str(gaDat.fxmin)])
disp('_____________________________________________________________')

% Ranking function
function FitV=ranking(ObjV,RFun)
if nargin==1
    error('Ranking function needs two parameters');
end

if ~(length(ObjV)==length(RFun))
    error('RFun have to be of the same size than ObjV.');
end

[val,pos]=sort(ObjV);
FitV(pos)=flipud(RFun);
FitV=FitV';

%% ------------------------------------------------------------------------
function [SelCh]=select(SEL_F, Chrom, FitnV, GGAP)
% ------------------ Selection Function ---------------------------------
if (nargin==3) %  No overlap
    if (SEL_F=='rws')
        % Metode Seleksi Roulette Wheel
        indices=rws(FitnV,length(FitnV));
        SelCh=Chrom(indices,:);
    elseif (SEL_F=='sus')
        indices=sus(FitnV,length(FitnV));
        SelCh=Chrom(indices,:);
    else
        error('Incorrect selection method');
    end
elseif (nargin==4) % With overlap
    % Indexes of new individuals
    if (SEL_F=='rws')
        indices=rws(FitnV,round(length(FitnV)*GGAP));
    elseif (SEL_F=='sus')
        indices=sus2(FitnV,round(length(FitnV)*GGAP));
    else
        error('Incorrect selection method');
    end

    if (GGAP<1) % there is overlap
        % Members of the population to overlap
        oldpos=(1:length(FitnV))';
        for k=1:length(FitnV)
            pos=round(rand*length(FitnV)+0.5);
            % exchange indexes
            oldpos([pos k])=oldpos([k pos]);
        end
```

```
        oldpos=oldpos(1:round(length(FitnV)*GGAP));
        SelCh=Chrom;
        SelCh(oldpos,:)=Chrom(indices,:);
    else % more childs than parents
        SelCh=Chrom(indices,:);
    end
else
    error('Incorrect number of paramenters');
end

% Disorder the population.
[kk,indi]=sort(rand(length(FitnV),1));
SelCh=SelCh(indi,:);

%% ------------------------------------------------------------------------
function NewChrom =lxov(OldChrom, XOVR, alpha)
% ------------------------ Linear crossover ------------------------------
% Membentuk populasi baru dengan linier crossover
%   NewChroms =lxov(OldChrom, XOVR, alpha, FieldDR)
%
% Rekombinasi linier
%   Child1 = beta1*Parent1+(1-beta1)*Parent2
%   Child2 = beta2*Parent1+(1-beta2)*Parent2

if nargin==1
    XOVR = 0.7;
    alpha = 0;
elseif nargin==2
    alpha = 0;
end

n = size(OldChrom,1);    % Jumlah individu dan kromosom
npares = floor(n/2);     % Jumlah pasangan
cruzar = rand(npares,1)<= XOVR;     % Pasangan untuk crossover
NewChrom=OldChrom;

for i=1:npares
    pin = (i-1)*2+1;
    if ~(cruzar(i)==0)
        betas=rand(2,1)*(1+2*alpha)-(0.5+alpha);
        A=[betas(1) 1-betas(1); 1-betas(2) betas(2)];
        NewChrom(pin:pin+1,:)=A*OldChrom(pin:pin+1,:);
    end
end

% aux = ones(n,1);
% auxf1=aux*FieldDR(1,:);
% auxf2=aux*FieldDR(2,:);
% NewChrom = (NewChrom>auxf2).*auxf2+(NewChrom<auxf1).*auxf1+(NewChrom<=auxf2 &
NewChrom>=auxf1).*NewChrom;

%% ------------------------------------------------------------------------
function NewChrom=mutbga(OldChrom,FieldDR,MutOpt)
% --------------------- Mutation function ---------------------------------
% OldChrom: Initial population.
% FieldChrom: Upper and lower bounds.
% MutOpt: mutation options,
%         MutOpt(1)=mutation probability (0 to 1).
%         MutOpt(2)=compression of the mutation value (0 to 1).
%         default MutOpt(1)=1/Nvar y MutOpt(2)=1

if (nargin==3)
    pm=MutOpt(1);
    shr=MutOpt(2);
```

```
elseif (nargin==2)
    pm=1/size(FieldDR,2);
    shr=1;
else
    error('Incorrect number of parameters');
end

Nind=size(OldChrom,1);
m1=0.5-(1-pm)*0.5;
m2=0.5+(1-pm)*0.5;
aux=rand(size(OldChrom));
MutMx=(aux>m2)-(aux<m1);
range=[-1 1]*FieldDR*0.5*shr;
range=ones(Nind,1)*range;
index=find(MutMx);
m=20;
alpha=rand(m,length(index))<(1/m);
xx=2.^(0:-1:(1-m));
aux2=xx*alpha;
delta=zeros(size(MutMx));
delta(index)=aux2;
NewChrom=OldChrom+(MutMx.*range.*delta);

%% -------------------------------------------------------------------------
function NewChrIx=sus2(FitnV, Nsel)
suma=sum(FitnV);
% Posisi roulette pointers
j=0;
sumfit=0;
paso=suma/Nsel; % jarak antar pointer
flecha=rand*paso; % keturunan pertama dari pointer pertama
NewChrIx(Nsel,1)=0;
for i=1:Nsel
    sumfit=sumfit+FitnV(i);
    while (sumfit>=flecha)
        j=j+1;
        NewChrIx(j)=i;
        flecha=flecha+paso;
    end
end

%% -------------------------------------------------------------------------
```

D. Hasil Program Dynamic Genetic Algorithm Pada Data 2012

Beban 308

```
_____________________________________________________________________
                          SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________
_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
_____________________________________________________________________
.
.
.
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45 45 108 110] -- f(Pmin): 1773.2983
_____________________________________________________________________
                          HASIL
_____________________________________________________________
-------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1773.2983
-------------------------------------------------------------
_____________________________________________________________
                  PEMBAGIAN PEMBEBANAN                      !
-------------------------------------------------------------!
   Pmin : [45 45 108 110]
_____________________________________________________________
F =
  1.7733e+003
P1 =
    45    45   108   110
```

Beban 310

```
_____________________________________________________________________

                          SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
.
.
.
Iterasi ke-: 175
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 176
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45 45 110 110] -- f(Pmin): 1768.6995
_____________________________________________________________________
                          HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1768.6995
---------------------------------------------------------------------
_____________________________________________________________________
                   PEMBAGIAN PEMBEBANAN                             !
---------------------------------------------------------------------!
   Pmin : [45 45 110 110]
_____________________________________________________________________
F =
  1.7687e+003
P1 =
    45    45   110   110
```

Beban 452

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)

_____________________________________________________________________
_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [39.7775143553084 52.0446930930799 138.513302360139 143.359572988865]
-- f(Pmin): 34907.0443
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [44.0210635279676 60.7472854551082 160 140.776959467756] -- f(Pmin):
7335.7074
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [37.0586530141034 61.4233316570913 160 160] -- f(Pmin): 1884.9646
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [42.1739980007963 66.9844623895734 160 140.209684530947] -- f(Pmin):
1865.3285
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [42.1739980007963 66.9844623895734 160 140.209684530947] -- f(Pmin):
1865.3285
_____________________________________________________________________
•
•
•
Iterasi ke-: 178
   Pmin: [0 70.2242819251285 150.604514691923 146.17150033924] -- f(Pmin):
1859.3476
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [0 70.2242819251285 150.604514691923 146.17150033924] -- f(Pmin):
1859.3476
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [0 70.2294346695285 150.598147183461 146.172742104608] -- f(Pmin):
1859.3476
_____________________________________________________________________
                          HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1859.3476
---------------------------------------------------------------------
_____________________________________________________________________
                  PEMBAGIAN PEMBEBANAN                              !
---------------------------------------------------------------------!
   Pmin : [0 70.2294346695285 150.598147183461 146.172742104608]
_____________________________________________________________________
F =
  1.8593e+003
P1 =
   84.9997     0.2294  150.5981  146.1727
```

Beban 525

```
_____________________________________________________________________

                          SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [45.0164689702424 84.9418473243549 177.2382418828 183.55395093477] --
f(Pmin): 2100.541
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [45.0164689702424 84.9418473243549 177.2382418828 183.55395093477] --
f(Pmin): 2100.541
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [46.0992349260191 84.9492965869305 172.739374564357 182.815274898726]
-- f(Pmin): 2084.6558
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [46.0992349260191 84.9492965869305 172.739374564357 182.815274898726]
-- f(Pmin): 2084.6558
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [46.0992349260191 84.9492965869305 172.739374564357 182.815274898726]
-- f(Pmin): 2084.6558
_____________________________________________________________________
•
•
•
Iterasi ke-: 178
   Pmin: [47.7169852002348 84.999999482288 170.90790815511 184.092092363255] -
- f(Pmin): 2083.1018
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [47.7169852002348 84.999999482288 170.90790815511 184.092092363255] -
- f(Pmin): 2083.1018
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [47.7169852002348 84.999999482288 170.90790815511 184.092092363255] -
- f(Pmin): 2083.1018
_____________________________________________________________________
                          HASIL
_____________________________________________________________
-------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 2083.1018
-------------------------------------------------------------
_____________________________________________________________
                  PEMBAGIAN PEMBEBANAN                      !
-------------------------------------------------------------!
   Pmin : [47.7169852002348 84.999999482288 170.90790815511 184.092092363255]
_____________________________________________________________
F =
  2.0831e+003
P1 =
   85.0   5.0000  170.9079  184.0921
```

Beban 360

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [45 45 150.944528740043 119.398123808036] -- f(Pmin): 2111.9675
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [45 45 110 160] -- f(Pmin): 1843.2195
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [45 45 160 110] -- f(Pmin): 1788.1495
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [45 45 160 110] -- f(Pmin): 1788.1495
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [45 45 160 110] -- f(Pmin): 1788.1495
_____________________________________________________________________
Iterasi ke-: 5
   Pmin: [45 45 153.208588148422 116.791411851578] -- f(Pmin): 1772.6526
_____________________________________________________________________
•
•
•
   Pmin: [45 45 142.032468619311 127.967531380689] -- f(Pmin): 1762.8738
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [45 45 142.032468619311 127.967531380689] -- f(Pmin): 1762.8738
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 142.032468676615 127.967531323385] -- f(Pmin): 1762.8738
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45 45 142.032468758241 127.967531241759] -- f(Pmin): 1762.8738
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45 45 142.032468758241 127.967531241759] -- f(Pmin): 1762.8738
_____________________________________________________________________
                          HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1762.8738
---------------------------------------------------------------------
_____________________________________________________________________
                   PEMBAGIAN PEMBEBANAN                             !
---------------------------------------------------------------------!
   Pmin : [45 45 142.032468758241 127.967531241759]
_____________________________________________________________________
F =
  1.7629e+003
P1 =
    45.00  5.0000  142.0325  127.9675
```

Beban 353

```
_____________________________________________________________________
                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________
_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [87.1704982266304 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [66.520849616495 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [87.1704982266304 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [59.9937899056429 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [59.3198251531912 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 5
   Pmin: [84.6337759070002 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
•
•
•
Iterasi ke-: 176
   Pmin: [81.6723409168354 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [63.5822313866079 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [63.1987903183946 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [64.2016535632215 45 110 110] -- f(Pmin): 1822.3893
_____________________________________________________________________
                          HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1822.3893
---------------------------------------------------------------------
_____________________________________________________________________
                  PEMBAGIAN PEMBEBANAN                             !
---------------------------------------------------------------------!
   Pmin : [64.2016535632215 45 110 110]
_____________________________________________________________________
F =
  1.8224e+003
P1 =
    88    45   110   110
```

Beban 282

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [45 45 96.4038077329672 97.9712818678031] -- f(Pmin): 4185.1228
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [45 45 96.4038077329672 97.9712818678031] -- f(Pmin): 4185.1228
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [45 45 96.4038077329672 97.9712818678031] -- f(Pmin): 4185.1228
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [45 45 96 97.8783669268939] -- f(Pmin): 3689.9078
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [45 45 97.5326070840287 96] -- f(Pmin): 3339.689
_____________________________________________________________________
Iterasi ke-: 5
   Pmin: [45 45 97.3582733641439 96] -- f(Pmin): 3165.9644
_____________________________________________________________________
•
•
•
Iterasi ke-: 176
   Pmin: [45 45 96 96] -- f(Pmin): 1812.5447
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [45 45 96 96] -- f(Pmin): 1812.5447
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 96 96] -- f(Pmin): 1812.5447
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45 45 96 96] -- f(Pmin): 1812.5447
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45 45 96 96] -- f(Pmin): 1812.5447
_____________________________________________________________________
                          HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1812.5447
---------------------------------------------------------------------
_____________________________________________________________________
                   PEMBAGIAN PEMBEBANAN                            !
---------------------------------------------------------------------!
   Pmin : [45 45 96 96]
_____________________________________________________________________
F =
  1.8125e+003
P1 =
    45    45    96    96
```

Beban 410

```
_____________________________________________________________________

                          SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [48.7200664734162 45 145.781806708436 133.534425664402] -- f(Pmin):
1825.094
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [48.7200664734162 45 145.781806708436 133.534425664402] -- f(Pmin):
1825.094
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [48.7200664734162 45 145.781806708436 133.534425664402] -- f(Pmin):
1825.094
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [48.7200664734162 45 145.781806708436 133.534425664402] -- f(Pmin):
1825.094
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [48.7200664734162 45 145.391181708436 133.534425664402] -- f(Pmin):
1825.087
_____________________________________________________________________
•
•
•
Iterasi ke-: 178
   Pmin: [70.3509757717545 45 145.344577106977 134.402806322869] -- f(Pmin):
1825.063
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [70.3509757717545 45 145.344577106977 134.402806322869] -- f(Pmin):
1825.063
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [70.3509757717545 45 145.344577106977 134.402806322869] -- f(Pmin):
1825.063
_____________________________________________________________________
                          HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1825.063
---------------------------------------------------------------------
_____________________________________________________________________
                   PEMBAGIAN PEMBEBANAN                             !
---------------------------------------------------------------------!
   Pmin : [70.3509757717545 45 145.344577106977 134.402806322869]
_____________________________________________________________________
F =
  1.8251e+003
P1 =
   85.2526     5.0000  145.3446  134.4028
```

Beban 455

```
_____________________________________________________________________

                          SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)

_____________________________________________________________________

_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [45 45 180.467214210004 179.525157689144] -- f(Pmin): 7018.0341
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [45 45 180.467214210004 179.525157689144] -- f(Pmin): 7018.0341
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
Iterasi ke-: 5
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
.
.
.
Iterasi ke-: 176
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45 45 185 180] -- f(Pmin): 2036.38
_____________________________________________________________________
                          HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 2036.38
---------------------------------------------------------------------
_____________________________________________________________________
                   PEMBAGIAN PEMBEBANAN                             !
---------------------------------------------------------------------!
   Pmin : [45 45 185 180]
_____________________________________________________________________
F =
  2.0364e+003
P1 =
    45.00  5.0000  185.0000  180.0000
```

Beban 285

```
_____________________________________________________________________
                          SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [45 45 110 85.5287363575565] -- f(Pmin): 2309.4399
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [45 45 110 85.3122014685193] -- f(Pmin): 2093.1534
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
Iterasi ke-: 5
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
•
•
•

Iterasi ke-: 176
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45 45 110 85] -- f(Pmin): 1781.3145
_____________________________________________________________________
                          HASIL
_____________________________________________________________
-------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 1781.3145
-------------------------------------------------------------

_____________________________________________________________
                  PEMBAGIAN PEMBEBANAN                      !
-------------------------------------------------------------!
   Pmin : [45 45 110 85]

_____________________________________________________________
F =
  1.7813e+003
P1 =
   45.0000   45.0000  110.0000   85.0000
```

E. Hasil Program Dynamic Genetic Algorithm Pada Data 2015

Beban 305

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)

Iterasi ke-: 0
   Pmin: [56.2594336593158 54.6979960467531 114.055890535916 122.793337223363] --
f(Pmin): 31627.5727
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [56.0112380758673 52.9806255619186 100 123.061584043889] -- f(Pmin):
16120.5767
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [56.0112380758673 52.9806255619186 100 123.061584043889] -- f(Pmin):
16120.5767
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [45 49.7481457432934 100 116.667591561792] -- f(Pmin): 6492.1809
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [45.8424880397917 45.8522465781578 100 115.457203779955] -- f(Pmin):
1384.9676
_____________________________________________________________________
.
.
.
Iterasi ke-: 177
   Pmin: [45.0673072766713 45.0000000039475 114.999999995998 100] -- f(Pmin):
74.1856
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45.0673072766713 45.0000000039475 114.999999995998 100] -- f(Pmin):
74.1856
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45.0505305447431 45.0000000039167 114.999999996061 100] -- f(Pmin):
74.1856
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45.0505305447431 45.0000000039167 114.999999996061 100] -- f(Pmin):
74.1856

                        HASIL
_____________________________________________________________________
---------------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 74.1856
---------------------------------------------------------------------
                                                                     !
                  PEMBAGIAN PEMBEBANAN                               !
                                                                     !
---------------------------------------------------------------------!
   Pmin : [45.0505305447431 45.0000000039167 114.999999996061 100]
_____________________________________________________________________
F =
   74.1856
P1 =
   45.0000   45.0000  115.0000  100.0000
```

- Beban 346

```
                    SKRIPSI
            PERHITUNGAN ECONOMIC DISPATH
      MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
         Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [53.0156378937311 47.4083713078106 123.203015284449 127.063565059053] --
f(Pmin): 80.6997
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [53.0156378937311 47.4083713078106 123.203015284449 127.063565059053] --
f(Pmin): 80.6997
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [53.0156378937311 47.4083713078106 123.203015284449 127.063565059053] --
f(Pmin): 80.6997
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [53.0156378937311 47.4083713078106 123.203015284449 127.063565059053] --
f(Pmin): 80.6997
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [53.0156378937311 47.4083713078106 123.203015284449 127.063565059053] --
f(Pmin): 80.6997
.
.
.
Iterasi ke-: 177
   Pmin: [72.6344006619942 49.1881478729163 125.861928270802 122.003135813304] --
f(Pmin): 80.6706
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [72.6344006619942 49.1881478729163 125.861928270802 122.003135813304] --
f(Pmin): 80.6706
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [73.7272077577129 49.1881642416977 125.861930438288 122.003115260096] --
f(Pmin): 80.6706
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [69.083690616771 49.1881056284291 125.861891291303 122.003164227975] --
f(Pmin): 80.6706
_____________________________________________________________________
                        HASIL
_____________________________________________________________
-------------------------------------------------------------

   Fungsi Objektif untuk Pmin : 80.6706

-------------------------------------------------------------

                PEMBAGIAN PEMBEBANAN                        !
-------------------------------------------------------------!
   Pmin : [69.083690616771 49.1881056284291 125.861891291303 122.003164227975]
_____________________________________________________________
F =
   80.6706
P1 =
   48.9468   49.1881  125.8619  122.0032
```

- Beban 385

```
_____________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________

Iterasi ke-: 0
   Pmin: [66.9212122373552 65.563092278179 136.22466488446 134.627032046914] --
f(Pmin): 88.7125
_____________________________________________________________
Iterasi ke-: 1
   Pmin: [66.9212122373552 65.563092278179 136.22466488446 134.627032046914] --
f(Pmin): 88.7125
_____________________________________________________________
Iterasi ke-: 2
   Pmin: [66.9914593207845 64.584328834434 134.880473430317 135.315187551573] --
f(Pmin): 88.6272
_____________________________________________________________
Iterasi ke-: 3
   Pmin: [61.6313880604739 61.5269110398607 129.938807577374 138.131346953844] --
f(Pmin): 88.5204
_____________________________________________________________
Iterasi ke-: 4
   Pmin: [61.6313880604739 61.5269110398607 129.938807577374 138.131346953844] --
f(Pmin): 88.5204
.
.
.
Iterasi ke-: 178
   Pmin: [77.2746585952211 55.2233854398789 131.228634334233 146.153648554163] --
f(Pmin): 88.4028
_____________________________________________________________
_____________________________________________________________
Iterasi ke-: 179
   Pmin: [76.6964522822795 55.2230834213933 131.228673392314 146.153597640613] --
f(Pmin): 88.4028
_____________________________________________________________
Iterasi ke-: 180
   Pmin: [76.6564816435345 55.2229892408523 131.228682220215 146.153593570996] --
f(Pmin): 88.4028
_____________________________________________________________
                        HASIL
_____________________________________________________________
-------------------------------------------------------------

   Fungsi Objektif untuk Pmin : 88.4028

-------------------------------------------------------------
                 PEMBAGIAN PEMBEBANAN                      !
-------------------------------------------------------------!
   Pmin : [76.6564816435345 55.2229892408523 131.228682220215 146.153593570996]

F =

   88.4028
P1 =
   52.3947   55.2230  131.2287  146.1536
```

- Beban 297

```
_____________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)

_____________________________________________________________
Iterasi ke-: 0
   Pmin: [49.7425616303467 47.3509853769151 112.617350007929 107.482872366641] --
f(Pmin): 15526.8858
_____________________________________________________________
Iterasi ke-: 1
   Pmin: [80.0272090428285 56.8608629278088 102.8138966819 100] -- f(Pmin):
7749.6055
_____________________________________________________________
Iterasi ke-: 2
   Pmin: [80.6982956821401 52.2327522915312 100.914298545286 100] -- f(Pmin):
1220.7748
_____________________________________________________________
Iterasi ke-: 3
   Pmin: [77.6414640560937 47.1457203636855 104.16389107335 100] -- f(Pmin):
73.2543
_____________________________________________________________
Iterasi ke-: 4
   Pmin: [77.6414640560937 47.1457203636855 104.16389107335 100] -- f(Pmin):
73.2543
.
.
.
Iterasi ke-: 176
   Pmin: [80.3281564073177 45 107 100] -- f(Pmin): 73.1048
_____________________________________________________________
Iterasi ke-: 177
   Pmin: [79.7098829339185 45 107 100] -- f(Pmin): 73.1048
_____________________________________________________________
Iterasi ke-: 178
   Pmin: [81.4935921410948 45 107 100] -- f(Pmin): 73.1048
_____________________________________________________________
Iterasi ke-: 179
   Pmin: [80.9430239348159 45 107 100] -- f(Pmin): 73.1048
_____________________________________________________________
Iterasi ke-: 180
   Pmin: [81.184173743801 45 107 100] -- f(Pmin): 73.1048
_____________________________________________________________
_____________________________________________________________
                        HASIL
_____________________________________________________________
-------------------------------------------------------------

   Fungsi Objektif untuk Pmin : 73.1048

-------------------------------------------------------------
                  PEMBAGIAN PEMBEBANAN                      !
-------------------------------------------------------------!
   Pmin : [81.184173743801 45 107 100]
F =
   73.1048
P1 =
    45    45   107   100
```

- Beban 435

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [79.5079583370761 64.8585667787275 131.452816001812 180.400222894703] --
f(Pmin): 99.5138
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [66.3051030877434 65.5033586543589 142.685236691149 169.03332033516] --
f(Pmin): 99.4947
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [66.3051030877434 65.5033586543589 142.685236691149 169.03332033516] --
f(Pmin): 99.4947
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [67.0094023614823 65.7556786296895 142.603334113529 169.437326576004] --
f(Pmin): 99.4908
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [78.3128149482985 66.9958719242178 136.456686000425 173.905326491396] --
f(Pmin): 99.4549
.
.
.
Iterasi ke-: 177
   Pmin: [46.9720809272299 62.9646386993903 138.107453000776 177.108836030247] --
f(Pmin): 99.418
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [49.0679992819332 62.9646486710951 138.10745409208 177.108792206446] --
f(Pmin): 99.418
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45 62.9646505808475 138.107454301084 177.108783813437] -- f(Pmin):
99.418
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45 62.9646505808475 138.107454301084 177.108783813437] -- f(Pmin):
99.418
_____________________________________________________________________
                          HASIL
_____________________________________________________________
-------------------------------------------------------------

   Fungsi Objektif untuk Pmin : 99.418

-------------------------------------------------------------
                                                             !
                  PEMBAGIAN PEMBEBANAN                       !
-------------------------------------------------------------!
   Pmin : [45 62.9646505808475 138.107454301084 177.108783813437]
F =
   99.4180
P1 =
   56.8191   62.9647  138.1075  177.1088
```

- Beban 288

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [54.7767388122334 48.5573228787432 103.255197955559 118.417976982082] --
f(Pmin): 27307.4136
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [54.572601981209 48.0778173054933 101.703370933138 117.376459489458] --
f(Pmin): 24234.1042
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [55.0351178160952 45 100 107.664058467476] -- f(Pmin): 9737.8839
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [55.0351178160952 45 100 107.664058467476] -- f(Pmin): 9737.8839
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [55.941520449291 45.3417195829763 100.775380446195 90] -- f(Pmin):
71.9012
.
.
.
Iterasi ke-: 175
   Pmin: [45.0109175892042 45 108 90] -- f(Pmin): 71.3703
_____________________________________________________________________
Iterasi ke-: 176
   Pmin: [45.014126539252 45 108 90] -- f(Pmin): 71.3703
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [45 45 108 90] -- f(Pmin): 71.3703
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [45 45 108 90] -- f(Pmin): 71.3703
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [45.0089435159376 45 108 90] -- f(Pmin): 71.3703
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [45.0327983545457 45 108 90] -- f(Pmin): 71.3703
_____________________________________________________________________
_____________________________________________________________
                          HASIL
_____________________________________________________________
-------------------------------------------------------------

   Fungsi Objektif untuk Pmin : 71.3703

-------------------------------------------------------------
                  PEMBAGIAN PEMBEBANAN                      !
-------------------------------------------------------------!
   Pmin : [45.0327983545457 45 108 90]
F =
   71.3703
P1 =
    45    45   108    90
```

- Beban 410

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [81.6289317584367 47.1159506849313 144.396916571079 158.807322338355] --
f(Pmin): 94.2308
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [79.5441138523505 47.4313156366972 137.634694001724 161.871282345382] --
f(Pmin): 94.1893
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [76.6666383113568 48.044750349629 137.201312624245 161.756474795161] --
f(Pmin): 94.1596
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [78.3494783758936 49.7058184578999 136.787724990131 161.556525313461] --
f(Pmin): 94.0557
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [77.0026661433348 55.1676126653592 134.802059369306 159.338104517546] --
f(Pmin): 93.8861
.
.
.
Iterasi ke-: 177
   Pmin: [75.5181537882172 59.0950052921791 134.668168404223 161.628725163106] --
f(Pmin): 93.7556
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [75.5181537882172 59.0950052921791 134.668168404223 161.628725163106] --
f(Pmin): 93.7556
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [75.5181537882172 59.0950052921791 134.668168404223 161.628725163106] --
f(Pmin): 93.7556
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [75.13713421532 59.0949978013784 134.668170283094 161.628734160875] --
f(Pmin): 93.7556
_____________________________________________________________________
                        HASIL
---------------------------------------------------------------

   Fungsi Objektif untuk Pmin : 93.7556

---------------------------------------------------------------

                  PEMBAGIAN PEMBEBANAN                        !
---------------------------------------------------------------!
   Pmin : [75.13713421532 59.0949978013784 134.668170283094 161.628734160875]
_____________________________________________________________________
F =
   93.7556
P1 =
   54.6081   59.0950  134.6682  161.6287
```

- Beban 373

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [54.5287472337745 81.1613179614158 112.238560672269 129.038036744612] --
f(Pmin): 87.7597
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [66.3221671465161 54.4827393064177 125.431633375502 136.48814217171] --
f(Pmin): 86.0559
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [66.3221671465161 54.4827393064177 125.431633375502 136.48814217171] --
f(Pmin): 86.0559
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [85 47.7269546771617 130.116583452229 143.898701672523] -- f(Pmin):
86.0056
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [85 47.7269546771617 130.116583452229 143.898701672523] -- f(Pmin):
86.0056
.
.
.
Iterasi ke-: 177
   Pmin: [84.7732856717237 53.3677333424984 129.57676925346 138.720366081498] --
f(Pmin): 85.9434
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [84.827660938817 53.3677335914091 129.576772814459 138.72036506229] --
f(Pmin): 85.9434
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [84.828579649223 53.3677335956146 129.576772874625 138.720365045069] --
f(Pmin): 85.9434
_____________________________________________________________________
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [84.827660938817 53.3677335914091 129.576772814459 138.72036506229] --
f(Pmin): 85.9434
_____________________________________________________________
                        HASIL
-------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 85.9434
-------------------------------------------------------------
                 PEMBAGIAN PEMBEBANAN                       !
-------------------------------------------------------------!
   Pmin : [84.827660938817 53.3677335914091 129.576772814459 138.72036506229]
_____________________________________________________________
F =
   85.9434
P1 =
   51.3351   53.3677  129.5768  138.7204
```

- Beban 420

```
_________________________________________________________________

                        SKRIPSI
              PERHITUNGAN ECONOMIC DISPATH
        MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
           Rina Anggraeni          (121910201110)
_________________________________________________________________

Iterasi ke-: 0
   Pmin: [83.1715483283948 46.2753432887252 151.833222229304 176.217411739947] --
f(Pmin): 97.0606
_________________________________________________________________
Iterasi ke-: 1
   Pmin: [83.1715483283948 46.2753432887252 151.833222229304 176.217411739947] --
f(Pmin): 97.0606
_________________________________________________________________
Iterasi ke-: 2
   Pmin: [83.4432453662424 47.4339080250809 144.131113849577 177.522452092912] --
f(Pmin): 96.4767
_________________________________________________________________
Iterasi ke-: 3
   Pmin: [83.4432453662424 47.4339080250809 144.131113849577 177.522452092912] --
f(Pmin): 96.4767
_________________________________________________________________
Iterasi ke-: 4
   Pmin: [72.522923590549 65.1477069099882 146.689915382942 153.961030771667] --
f(Pmin): 96.3014
.
.
.
Iterasi ke-: 176
   Pmin: [82.6941419962473 60.6428001616008 136.043565466511 167.821285070421] --
f(Pmin): 95.9834
_________________________________________________________________
Iterasi ke-: 177
   Pmin: [82.6941419962473 60.6428001616008 136.043565466511 167.821285070421] --
f(Pmin): 95.9834
_________________________________________________________________
Iterasi ke-: 178
   Pmin: [82.6941419962473 60.6428001616008 136.043565466511 167.821285070421] --
f(Pmin): 95.9834
_________________________________________________________________
Iterasi ke-: 179
   Pmin: [82.6941419962473 60.6428001616008 136.043565466511 167.821285070421] --
f(Pmin): 95.9834
_________________________________________________________________
Iterasi ke-: 180
   Pmin: [82.6941419962473 60.6428001616008 136.043565466511 167.821285070421] --
f(Pmin): 95.9834
_____________________________________________________________
                         HASIL
-------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 95.9834
-------------------------------------------------------------
                 PEMBAGIAN PEMBEBANAN                       !
-------------------------------------------------------------!
   Pmin : [82.6941419962473 60.6428001616008 136.043565466511 167.821285070421]
_____________________________________________________________
F =
   95.9834
P1 =
   55.4923   60.6428  136.0436  167.8213
```

- Beban 265

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________
Iterasi ke-: 0
   Pmin: [80.1017466548282 62.3010533570503 71.7444502504878 81.263135453422] --
f(Pmin): 71.606
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [80.1017466548282 62.3010533570503 71.7444502504878 81.263135453422] --
f(Pmin): 71.606
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [85 66.2081031402507 86.9580276938905 52.7001859709668] -- f(Pmin):
70.7933
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [85 61.1533624926996 103.323759284834 45] -- f(Pmin): 69.0301
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [85 59.2805782167316 109.387272118688 45] -- f(Pmin): 68.433
.
.
.
Iterasi ke-: 175
   Pmin: [85 45 117.147713445256 57.8243808375052] -- f(Pmin): 67.2491
_____________________________________________________________________
Iterasi ke-: 176
   Pmin: [85 45 117.147713445256 57.8243808375052] -- f(Pmin): 67.2491
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [85 45 117.160382234015 57.8281651452337] -- f(Pmin): 67.2489
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [85 45 117.160311418587 57.8304628215663] -- f(Pmin): 67.2488
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [85 45 117.163230332689 57.8348871060827] -- f(Pmin): 67.2487
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [85 45 117.163230332689 57.8348871060827] -- f(Pmin): 67.2487
_____________________________________________________________________
                          HASIL
---------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 67.2487
---------------------------------------------------------------
                  PEMBAGIAN PEMBEBANAN                        !
---------------------------------------------------------------!
   Pmin : [85 45 117.163230332689 57.8348871060827]
_____________________________________________________________________
F =
   67.2487
P1 =
   45.0019   45.0000  117.1632   57.8349
```

- Beban 197

```
_____________________________________________________________

                      SKRIPSI
            PERHITUNGAN ECONOMIC DISPATH
        MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
           Rina Anggraeni          (121910201110)
_____________________________________________________________

Iterasi ke-: 0
   Pmin: [48.5298601795123 80.1736153649187 49.0380773296724 64.7128186780308] --
f(Pmin): 41998.5033
_____________________________________________________________
Iterasi ke-: 1
   Pmin: [65.3685574277802 84.2800612737578 45 57.1889430935975] -- f(Pmin):
34543.5379
_____________________________________________________________
Iterasi ke-: 2
   Pmin: [63.2043441134704 85 45 47.7135722034734] -- f(Pmin): 25787.0634
_____________________________________________________________
Iterasi ke-: 3
   Pmin: [69.9823830459359 76.1756095892138 45 51.6031075926791] -- f(Pmin):
20850.1659
_____________________________________________________________
Iterasi ke-: 4
   Pmin: [70.0961794041117 76.7034335603506 45 50.5448054109817] -- f(Pmin):
20319.6937
.
.
.
Iterasi ke-: 176
   Pmin: [72.3321063020712 45 46.2723263638682 60.7275042978673] -- f(Pmin):
65.5445
_____________________________________________________________
Iterasi ke-: 177
   Pmin: [71.2640201817517 45 46.2764108122332 60.7235716771853] -- f(Pmin):
65.5435
_____________________________________________________________
Iterasi ke-: 178
   Pmin: [71.2640201817517 45 46.2764108122332 60.7235716771853] -- f(Pmin):
65.5435
_____________________________________________________________
Iterasi ke-: 179
   Pmin: [70.7877420031862 45 46.2774641245438 60.7225032404314] -- f(Pmin):
65.5432
_____________________________________________________________
Iterasi ke-: 180
   Pmin: [70.7877420031862 45 46.2774641245438 60.7225032404314] -- f(Pmin):
65.5432
_____________________________________________________________
                         HASIL
-------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 65.5432
-------------------------------------------------------------
                  PEMBAGIAN PEMBEBANAN                      !
-------------------------------------------------------------!
   Pmin : [70.7877420031862 45 46.2774641245438 60.7225032404314]
_____________________________________________________________
F =
   65.5432
P1 =
   45.0000   45.0000   46.2775   60.7225
```

- Beban 238

```
_____________________________________________________________________

                        SKRIPSI
                PERHITUNGAN ECONOMIC DISPATH
          MENGGUNAKAN METODE DYNAMIC GENETIC ALGORITHM
             Rina Anggraeni          (121910201110)
_____________________________________________________________________

Iterasi ke-: 0
   Pmin: [62.9254021289003 64.4944416393017 74.9290456150309 65.0272512550173] --
f(Pmin): 11519.5861
_____________________________________________________________________
Iterasi ke-: 1
   Pmin: [53.9250386977405 48.9716484114059 49.3949577899892 90.8189670262774] --
f(Pmin): 71.2156
_____________________________________________________________________
Iterasi ke-: 2
   Pmin: [55.6100459121372 54.2123022245552 56.0396519761741 82.0247581632062] --
f(Pmin): 69.7856
_____________________________________________________________________
Iterasi ke-: 3
   Pmin: [55.6100459121372 54.2123022245552 56.0396519761741 82.0247581632062] --
f(Pmin): 69.7856
_____________________________________________________________________
Iterasi ke-: 4
   Pmin: [57.1417061022042 52.3061827854811 58.706921070966 79.6339453463067] --
f(Pmin): 69.1599
.
.
.
Iterasi ke-: 176
   Pmin: [85 45 74.8086555672598 73.191338086866] -- f(Pmin): 66.0178
_____________________________________________________________________
Iterasi ke-: 177
   Pmin: [85 45 74.8086555672598 73.191338086866] -- f(Pmin): 66.0178
_____________________________________________________________________
Iterasi ke-: 178
   Pmin: [85 45 74.8086555672598 73.191338086866] -- f(Pmin): 66.0178
_____________________________________________________________________
Iterasi ke-: 179
   Pmin: [85 45 74.808683134935 73.1912156342476] -- f(Pmin): 66.0178
_____________________________________________________________________
Iterasi ke-: 180
   Pmin: [84.8148147175963 45.0000000003053 74.8125410380183 73.1869978458613] --
f(Pmin): 66.0173
_____________________________________________________________________
                        HASIL
---------------------------------------------------------------
   Fungsi Objektif untuk Pmin : 66.0173
---------------------------------------------------------------
                  PEMBAGIAN PEMBEBANAN                        !
---------------------------------------------------------------!
   Pmin : [84.8148147175963 45.0000000003053 74.8125410380183 73.1869978458613]
_____________________________________________________________

F =
   66.0173
P1 =
   45.0005   45.0000   74.8125   73.1870
```