INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO
KONSENTRASI TEKNIK ENERGI LISTRIK

OPTIMASI ECONOMIC DISPATCH COMBINED CYCLE POWER PLANT DENGAN PENDEKATAN METODE GENETIC ALGORITHM

SKRIPSI

Disusun Oleh :
ARDIAN HARIYANSYAH
99.12.007

APRIL 2006

LEMBAR PERSETUJUAN

OPTIMASI *ECONOMIC DISPATCH COMBINED CYCLE POWER PLANT* DENGAN PENDEKATAN METODE GENETIC ALGORITHM

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Syarat-Syarat Guna Mencapai Gelar Sarjana Teknik

Disusun Oleh :

ARDIAN HARIYANSYAH
NIM. 99. 12. 007

Mengetahui,	**Menyetujui,**
Ketua Jurusan Teknik Elektro S-1	**Dosen Pembimbing**
Ir. F. Yudi Limpraptono, MT	**Ir. Widodo Pudji M, MT**
NIP. Y. 1039500274	**NIP. 1028700171**

KONSENTRASI TEKNIK ENERGI LISTRIK
JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

ABSTRAKSI

OPTIMASI *ECONOMIC DISPATCH* COMBINED CYCLE POWER PLANT DENGAN METODE *GENETIC ALGORITHM*

Ardian Hariyansyah, 99.12.007, Jurusan Teknik Elektro/Teknik Energi Listrik S-1, Fakultas Teknologi Industri, Institut Tenologi Nasional Malang, 2006, Dosen Pembimbing Ir. Widodo Pudji Muljanto, MT.

Kata Kunci: *Economic Dispatch* (Pembebanan Ekonomis), *Combined Cycle Power Plant*, *Genetic Algorithm*.

Dalam skripsi ini membahas masalah optimasi untuk biaya bahan bakar pada suatu sistem tenaga listrik dengan menggunakan metode *Genetic Algorithm* pada Pembangkit Listrik Tenaga Termal. Pembangkit Listrik Tenaga Termal tersebut adalah *Combined Cycle Power Plant* (CCPP) atau dalam bahasa Indonesia disebut Pusat Listrik Tenaga Gas Uap (PLTGU).

Pada CCPP gas panas yang dibuang hasil pembakaran yang dibuang oleh turbin gas tersebut dimanfaatkan kembali melalui peralatan yang disebut *Heat Recovery Steam Generator* (HRSG) untuk menghasilkan *Cycle* uap-air yang akan digunakan memutar sebuah turbin uap. Pembangkit listrik tenaga termal jenis ini memiliki keuntungan karena efisiensi *Cycle* termalnya menjadi lebih tinggi Hasil dari analisis tersebut nantinya dapat digunakan sebagai salah satu acuan dalam mengefisiensikan biaya bahan bakar yang bisa digunakan oleh PT. PLN. PJB II Unit Pembangkitan Gresik. Sehingga nilai kerugian yang sementara ini diderita oleh PT. PLN. PJB UP Gresik bisa berkurang, yang akhirnya nanti akan bisa menambah keuntungan bagi PT. PLN sebagai perusahaan penyedia energi listrik di Indonesia.

Analisa dilakukan dengan bantuan Program Komputer menggunakan bahasa pemrograman *Borland Delphi* versi 7.0 dicoba pada PT. PLN. PJB II Unit Pembangkitan Gresik yang terdiri dari 3 Blok *Combined Cycle Power Plant*. Dengan menggunakan *Economic Dispatch* dengan metode *Genetic Algorithm* maka akan didapatkan optimasi biaya bahan bakar yang lebih efisien.

PERSEMBAHANKU

Untuk sang pemberi nafasku sehari dan juga penguasa jagad raya Allah SWT,

serta pemimpin jalanku menuju surgadan kebaikan didunia dan diakhirat Muhammad SAW

Kepada Orang Tuaku Yang selalu membimbingku mulai dari dalam Kandangan.

Aku tidak dapat membalasnya aku hanya bisa berdoa Kepada Allah

agar kedua orang tuaku selamat dunia Akhirat

Kepada Kakek dan nenek tertua kasih atas perhatianya dan doanya

Kepada adikku dan Putri terimakasih atas doanya

Kepada keluarga ku Abah Ifan dan Mas Sokhkin, Abah Wiran Mas Pan Pakde Ky n Bude Nam, Abah n Mas Gito

Mba Lis n suami cak Kur, Cak Kasim, Tak Kusnan, Mas Mas, Cak Iyan Mba Nuah tertua kasih semuanya

atas bantuanya dari materi sampai do'a yang terkabulnya atas [illegible]

Kepada sepupuku, epas, echa, [illegible] Abah Nur, om, dan sekeluarga, [illegible] amanah, [illegible]

Zizah, Fatma, Family n yang lupa [illegible] dan Cak Kusnan

Kepada seorang yang selalu kuharap menjadi pendampingku [illegible] Kan aku telah menyebut [illegible] dan setiap doaku

PERSEMBAHAN

KEPADA SANG PENCIPTA ALAM SEMESTA INI DAN RASUL NABI MUHAMMAD SAW. YANG MEMBERI AKU JALAN KEBENARAN

KEPADA KEDUA ORANG TUAKU YANG SELALU MEMBIMBING AKU SAMPAI AKU MENJADI MANUSIA YANG BERBAKTI.

KEPADA KAKEK NENEKU YANG SELALU MEMBERI WEJANGAN DALAM BERKEHIDUPAN DI DUNIA INI.

KEPADA ADIKKU DAN PUTRI YANG SELALU MEMBUAT GERIGITAN

KEPADA KELUARGAKU DIMALANG DAN DI BANJAR: MBAK LIF, DAN MAS SHOLIKHIN, MBA WIN N MAS PUR,MBA IS MAS GITO, PAKDE RI DAN BUDE,CAK KUSNAN DAN CAK KUSNIN, CAK IYAN, MBA INJILAH, MBA MUS ,DAN SEMUA KELUARGA KU YANG KUSAYANG.

KEPADA SEPUPUKU:
GEVAN,OPSA,NELSA,ECHA,RARA,NOVI,ERNA, ETI,ULFA,SOFI,ANJAMUL,ZIZAH,FATMA,FANDI, KUDOAKAN SEMOGA,SUKSES SELALU.

KEPADA SESEORANG YANG SELALU KU INGAT AGAR KELAK KAU MENJADI PENDAMPINGKU

Ardian

99.12 007

Best friend: Rendra , Febri N Jaka, Agus N Rahmad, Roy, Andy pranah,
Dwi Unto, Wahyu, Trilisdiesa, AwanK, AdyBTN, Hernawan, alfert

Best Friend Girl: Azizah, Nani.iin.yuni.feby.Nina.dina.Dini.mudah.indy bely.Widy.tantri.surya.isna.
diana.vinta, Mimin.indah mamake.rosy.ifa.avid, lia.herlin, nelly, uun,
Yuni Nganjuk.vell.pepy, emil.Aiflen.rahma.maya, andien, ndank.novi, n semuanya yang aku lupa

Kos 443B Bend.Sutami
(pyman), lala.susilo.Tompel.andre dani, Gempluk, samsul.andre, mbah edi,
sableng.andikriwul, markopek, arykomandan, Yazid, gofur, dicky.oby, mudin
n david serta teman" yang aku lupa terima kasih atas guyonan n saran sarannya.

Kepada teman Perumahanku
widodo.ratno.dayat.Ojiesx.aglar.ya"n.jono.udin.aili.agus latat.supri.hendri, wayo,
amat dampul.keri gendring.ujang.aili.yudi angot.rilo n semuanya gang aku lupa thank's ya atas bagayaan aya.

Teman Jekal malang raya 14.390 Mhz, Bunda Ayat N om Jemur, Pedro, Oncky, Cayung, Jony, Jikri.nanda.acil.om hardi.mba Jin,

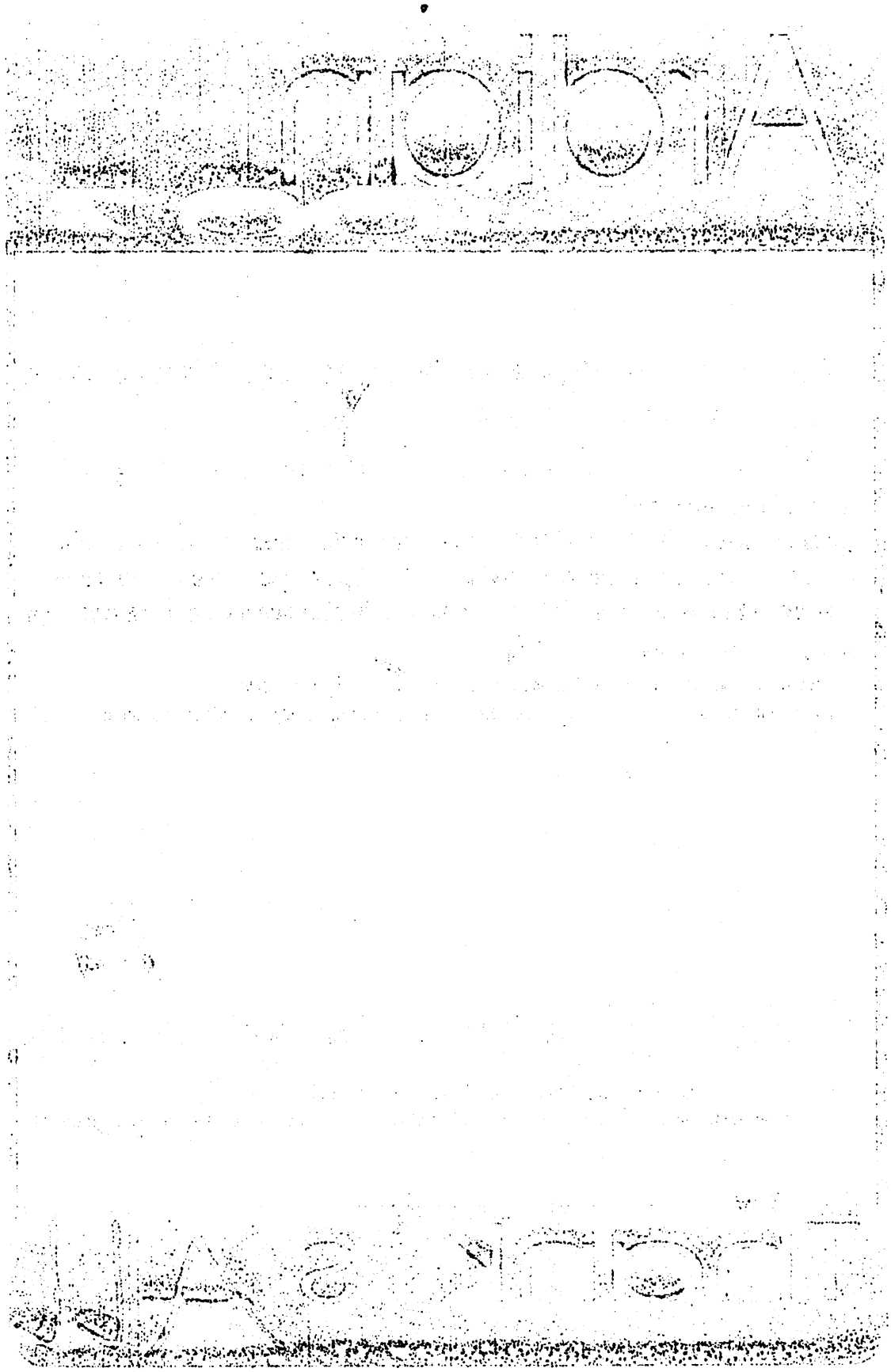
Teman gaulku:Drey.sigit.bowo.aksan.memet.wanda.ojy.ali.hendra.mono.deny.idur.aat.alan.kiky,
teman kampusku:didit.ngeb.farid.rahmad.bowo.gandhi.ekotel.degan.panjul.asrofi.guntur.tanto.wahyu,
peter.agus.pi'i.imam.arman.issa.n semuanya thanks ya atas kerja samanya selama ini

Kopma :Ming, Santi, fahrul, sigit ,dany ,jis, Nita,tepan,gatot,im n crew semoga kopma berjaya trusss

Format: ludy.bastian lpunk.ajie.rino.raman.aan.mba martha.deny.sapi,
iking.dany.gandrong pex.n semua anggota format tahnk's ya. Semoga karya kalian go internasional

http://fly1o/JumpNET

Thank`s ALL



KATA PENGANTAR

Puji Syukur Kehadirat ALLAH SWT atas segala limpahan Rahmat dan Hidayah-Nya yang telah diberikan sehingga penyusunan skripsi ini dapat terselesaikan dengan baik. Karena hanya atas Kehendak-Nya semata penulis mampu dan sanggup bertahan melewati segala macam cobaan dan kendala yang ada selama penyusunan skripsi ini.

Skripsi berjudul "Optimasi *Economic Dispatch Combined Cycle Power Plant* Dengan Pendekatan Metode *Genetic Algorithm*" ini disusun sebagai salah satu syarat untuk mendapatkan gelar Sarjana Teknik di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Nasional Malang.

Selama penyusunan skripsi ini, tidak terlepas dari bantuan, bimbingan dan motivasi dari berbagai pihak. Oleh karena itu atas segala yang pernah diberikan, tak lupa penulis menyampaikan rasa hormat dan terima kasih yang mendalam kepada :

1. Bapak DR. Ir. Abraham Lomi, MSEE. Selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. Mochtar Asroni, MSME, Selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang.
3. Bapak Ir. Yudi Limpraptono, MT , Selaku Ketua Jurusan Teknik Elektro Institut Teknologi Nasional Malang.
4. Ibu Ir. Mimin Mustikawati, Selaku Sekretaris Jurusan Teknik Elektro Institut Teknologi Nasional Malang.

5. Ibu Ida Barita, SH, Selaku Dosen Wali Elektro Energi Listrik 99 Kls 1-2
6. Bapak Ir. Widodo Pudji Muljanto, MT , Selaku Dosen Pembimbing.
7. Rekan-rekan Teknik Elektro ITN Malang yang telah banyak membantu dalam penyelesaian skripsi ini.

Penulis menyadari bahwa masih terdapat berbagai kekurangan dalam penyusunan skripsi ini. Oleh karenanya, segala saran dan kritik yang sifatnya membangun sangat penulis harapkan. Semoga skripsi ini kelak dapat berguna dan bermanfaat bagi yang membacanya.

Malang, Maret 2006

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR GRAFIK

BAB I

PENDAHULUAN

1.1. Latar Belakang

Pembangkitan tenaga listrik merupakan bagian dari permasalahan energi dan lingkungan yang dihadapi oleh Indonesia sebagai negara berkembang. Secara garis besar, suatu sistem tenaga listrik dibagi menjadi 3 bagian yaitu : sisi pembangkit tenaga listrik, jaringan transmisi dan beban. Untuk suatu operasi pada beban tertentu, perhitungan ekonomis harus tetap merupakan suatu prioritas atau nilai yang harus diperhitungkan disamping hal-hal yang lain, sehingga nantinya diperlukan suatu rencana operasi yang optimum dengan tetap memenuhi beberapa persyaratan pengoperasian sistem tenaga listrik yaitu antara lain daya yang dibangkitkan cukup untuk memasok beban. Dalam pembangkitan tenaga listrik dilakukan usaha agar biaya pembangkitannya semurah mungkin. Usaha untuk mengoptimalkan biaya operasi ini , salah satunya dilakukan dengan penerapan *Economic Dispatch.* Didalam operasi sistem tenaga listrik *Economic Dispatch* adalah hal yang sangat perlu diperhatikan untuk mendapatkan biaya bahan bakar yang ekonomis (murah) dalam suatu sistem pembangkit.

Dalam pemenuhan kebutuhan akan energi listrik dari para pelanggan maka diperlukan perencanaan penggunaan pembangkit listrik yang ada secara efisien dan seoptimal mungkin. Namun kendala utama yang dihadapi oleh pembangkit listrik tenaga termal adalah tingginya biaya yang harus

dikeluarkan untuk membeli bahan bakar. Upaya menekan biaya dan pertimbangan akan besarnya kebutuhan energi dalam negeri serta kelestarian sumber daya energi untuk masa mendatang mendorong dilakukannya usaha-usaha konversi energi.

Bertitik tolak dari masalah konservasi energi tersebut, khususnya dalam tahap pengoperasian pembangkit listrik tenaga termal, telah ditempuh cara-cara tertentu antara lain dengan melakukan pengaturan pembebanan yang optimum (ekonomis) pada sebuah pembangkit listrik termal. Sebuah tipe baru Pembangkit Listrik Tenaga Termal mulai dikembangkan dan menjanjikan prospek yang demikian luas. Pembangkit Listrik Tenaga Termal tersebut adalah *Combined Cycle Power Plant* (CCPP) atau dalam bahasa Indonesia disebut Pusat Listrik Tenaga Uap (PLTGU).

1.2. Rumusan Masalah

Pada pembebanan di PLTGU Unit Pembangkitan Gresik pengaturan distribusi beban perlu dipertimbangkan.

Untuk memecahkan permasalahan di atas, akan digunakan metode *Genetic Algorithm*. Maka dari itu muncul beberapa pertanyaan, antara lain:

1. Apakah dengan metode pengaturan pembebanan yang dipakai pada PLTGU Gresik telah tercapai pembebanan ekonomis *(economic dispatch).*
2. Dengan metode *Genetic Algorithm* apa dapat dicapai pembebanan ekonomis *(economic dispatch)* yang lebih baik.

Berdasar pada apa yang ada pada permasalahan di atas maka kami menyusun skripsi dengan judul :

"OPTIMASI *ECONOMIC DISPATCH* *COMBINED CYCLE POWER PLANT* DENGAN PENDEKATAN METODE *GENETIC ALGORITHM* "

1.3. Tujuan Pembahasan

Adapun tujuan dari skripsi ini adalah :

- Menentukan pola pembebanan bagi masing-masing blok yang terdapat di *Combine Cycle Power Plant* (CCPP) PT. PLN. PJB II Unit Pembangkitan Gresik dengan Metode *Genetic Algorithm* agar dapat dioperasikan lebih optimum (ekonomis).
- Membandingkan hasil yang diperoleh dengan pola pembebanan yang telah diterapkan di *Combine Cycle Power Plant* PT. PLN. PJB II Unit Pembangkitan Gresik.

1.4. Batasan Masalah

Agar skripsi ini mencapai sasaran, maka pembahasan masalah dibatasi:

- Menganalisa Optimasi pembebanan hanya pada PLTGU Gresik saja.
- Menggunakan Metode *Genetic Algorithm*
- Menganalisa Unit Pembangkitan Termal berbahan bakar gas.

- Pengamatan dilakukan terhadap CCPP Unit Pembangkitan Gresik yang beroperasi pada pola konfigurasi 3.3.1-3.3.1-3.3.1, yaitu kapasitas maksimum pembangkit.
- Rugi-rugi transmisi diabaikan.

1.5. Metodologi Pembahasan

Metodologi yang digunakan dalam pembahasan dilaksanakan dengan langkah-langkah sebagai berikut :

- Studi kepustakaan mengenai hal-hal yang berhubungan dengan pembahasan masalah.
- Studi lapangan untuk mendapatkan data-data yang dibutuhkan serta mencari data dari objek yang diperlukan sesuai dengan pedoman teori yang diperoleh dari studi kepustakaan.
- Melakukan perhitungan dari data lapangan yang berkaitan dengan optimasi pembebanan dengan metode *Genetic Algorithm* kemudian membandingkannya dengan pencapaian optimasi Pembebanan pada PLTGU Gresik.
- Membuat evaluasi, sehingga dapat disimpulkan apakah dengan metode *Genetic Algorithm* yang diterapkan lebih efesien (ekonomis).

1.6. Kontribusi Pembahasan

Dengan adanya analisa ini diharapkan nantinya dapat memberikan alternatif terbaik dalam pemecahan yang ada pada sistem *Combine Cycle*

Power Plant (CCPP) untuk mendapatkan pembebanan yang optimum (ekonomis) pada masing-masing blok serta harus memperhatikan kendala-kendala pengoperasian pembangkit listrik tenaga temal yang bersangkutan sehingga produksi tenaga listrik dapat dilakukan dengan biaya operasi serendah mungkin.

1.7. Sistematika Penulisan

Penulisan skripsi ini terdiri atas beberapa bab dan sub bab dan tersusun dengan sistematika sebagai berikut :

BAB I PENDAHULUAN

Terdiri atas latar belakang penulisan, rumusan masalah, maksud dan tujuan penulisan skripsi, lingkup bahasan dan sistematika penulisan.

BAB II TEORI DASAR

Berisi antara lain teori dasar tentang sistem tenaga listrik dan karakteristik pembangkitan termal, pembebanan ekonomis pembangkitan termal, economic dispatch menggunakan metode *Genetic Algorithm*, kendala operasi PLTGU.

BAB III PEMBANGKIT LISTRIK TENAGA GAS UAP

Berisi data konsumsi bahan bakar masing-masing blok, konfigurasi PLTGU Unit Pembangkitan Gresik.

BAB IV ANALISA DATA

Berisi tentang hasil perhitungan dengan *Genetic Algorithm* dibandingkan dengan pola pembebanan yang sekarang digunakan di PLTGU Gresik, dengan menggunakan data yang diperoleh selama survey dilapangan sebagai variabelnya.

BAB V KESIMPULAN

Berisi tentang kesimpulan akhir, dan saran-saran.

BAB II

TEORI DASAR

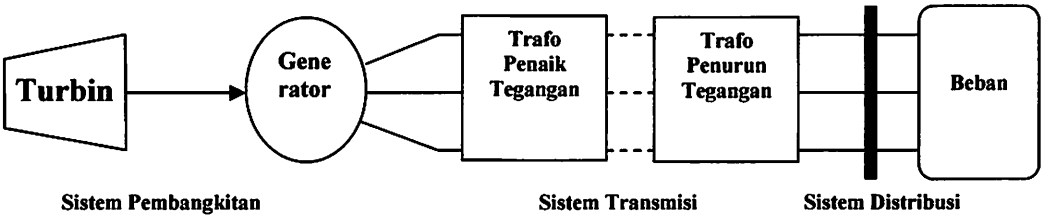
2.1 Sistem Tenaga Listrik[3]

Performa dari sebuah pusat pembangkit tenaga listrik pada prinsip ditentukan oleh apa yang dinamakan lengkung masukan-keluaran (Input-Output).. Lengkung ini memberikan gambaran tentang efisiensi thermis pusat pembangkit tersebut. Selain tergantung pada sifat-sifat pusat tenaga listrik itu sendiri, seperti keadaan air, pendingin kualitas bahan bakar, kecakapan para operator pusat pembangkit dan bentuk lengkung beban.

Berikut ini macam-macam karakteristik pembangkit yang berhubungan dengan penjadwalan operasi pembangkit untuk memperjelas keterangan diatas.

Jaringan setelah keluar dari gardu induk umumnya disebut jaringan distribusi, sedangkan jaringan antara pusat listrik dengan gardu induk disebut jaringan transmisi. Setelah itu disalurkan melalui jaringan distribusi primer maka tenaga listrik kemudian diturunkan tegangannya oleh gardu-gardu distribusi menjadi tegangan 380/220 volt atau 220/127 volt, dan baru kemudian disalurkan ke konsumen.

Dari uraian diatas kiranya dapat dimengerti bahwa besar kecilnya tenaga listrik ditentukan sepenuhnya oleh konsumen, yaitu tergantung bagaimana para konsumen akan menggunakan peralatan listriknya, kemudian PT. PLN (Persero) harus mengimbangi kebutuhan tenaga listrik tersebut dalam arti selalu menyesuaikan daya listrik yang dibangkitkan dari waktu ke waktu.



Gambar 2-1 : Elemen Pokok Sistem Tenaga Listrik

Untuk memenuhi kebutuhan tenaga listrik bagi para konsumen, diperlukan berbagai peralatan listrik. Berbagai peralatan listrik ini dihubungkan satu sama yang lain secara keseluruhan membentuk suatu sistem tenaga listrik. Sehingga yang dimaksud dengan sistem tenaga listrik di sini adalah sekumpulan pusat listrik dan gardu induk yang satu dengan yang lain dihubungkan oleh jaringan transmisi sehingga merupakan sebuah kesatuan interkoneksi. Oleh karena itu daya listrik yang dibangkitkan harus sama dengan tenaga listrik yang dibutuhkan oleh konsumen

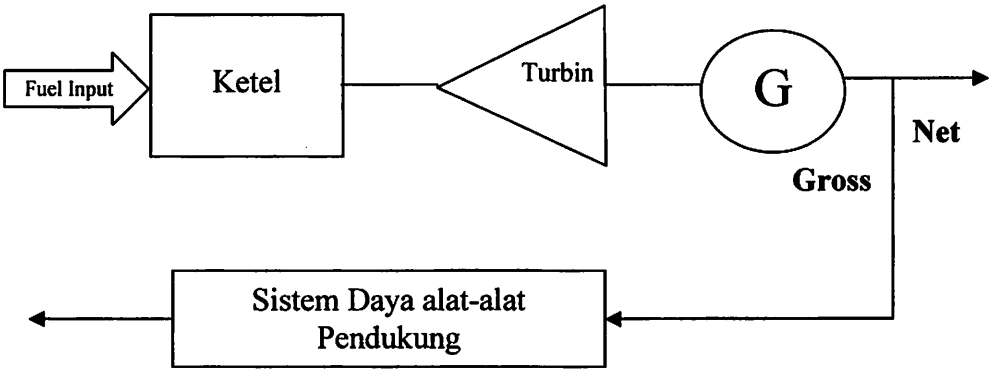
2.2 Karakteristik Pembangkit Listrik Termal

Dasar untuk menyelesaikan masalah pembebanan ekonomis (economic dispatch) dari sebuah pembangkit listrik tenaga termal sangat ditentukan oleh karakteristik masukan-keluaran (input-output characteristics) pusat listrik tersebut.

Tipe dari sebuah blok pembangkit listrik tenaga termal tampak pada gambar 2-2. Blok tersebut terdiri atas sebuah ketel yang menghasilkan uap untuk menggerakkan turbin uap yang aka dikopel dengan sebuah generator listrik. Daya listrik yang dihasilkan tidak seluruhnya disalurkan ke sistem

tetapi sebagian kecil digunakan untuk mengoprasikan peralatan yang terdapat pada pusat listrik tersebut. Sebuah PLTU kira-kira memerlukan 2% sampai 6% dari daya listrik yang dihasilkan oleh generator untuk mengoprasikan berbagai peralatan seperti ketel, pompa, kompresor, dan sebagainya, serta untuk mencatu peralatan kontrol, telemetri, komunikasi, penerangan dan komputer.

Dalam mendefinisikan karakteristik masukan-keluaran sebuah pembangkit listrik tenaga termal, akan dibiarakan tentang gross input dan net output yang dihasilkan pusat listrik tersebut. Gross input pembangkit listrik termal menyatakan jumlah keseluruhan bahan bakar yang diperlukan, sedangkan net output adalah daya nyata (real power) yang dihasilkan generator setelah dikurangi untuk keperluan sendiri.



Gambar 2-2 : Blok Pembangkit listrik tenaga uap

2.2.1 Karakteristik Masukan Keluaran

Masukan sebuah Pembangkit Listrik Termal umumnya dinyatakan sebagai banyaknya energi persatuan waktu dari bahan bakar yang diberikan ke

ketel untuk menghasilkan daya listrik yang merupakan keluaran dari pusat listrik tersebut. Terdapat dua notasi yang umum digunakan :

H dengan satuan [MBTU/hour]

F dengan satuan [$US/hour],

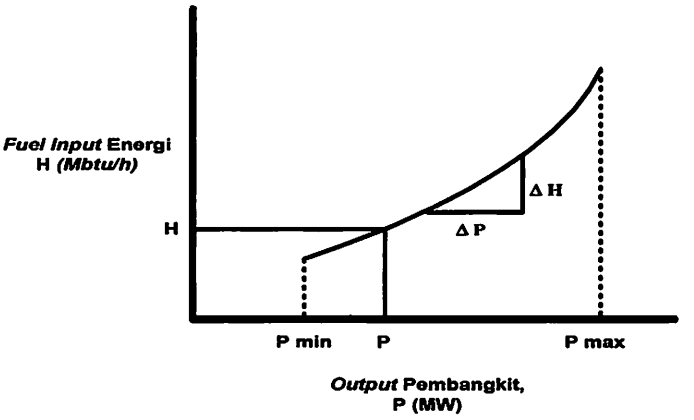
dimana $F = H \times \$US/BTU$, dan *$US/BTU* menyatakan harga bahan bakar persatuan energi yang dikandung oleh bahan bakar tersebut. Sedangkan keluaran dari pembangkit listrik termal adalah daya nyata yang dihasilkan oleh generator dikurangi dengan daya nyata yang dipakai oleh pusat listrik tersebut. Notasi yang digunakan adalah :

P dengan satuan [MW],

jika dinyatakan bahwa masukan pusat listrik merupakan fungsi terhadap keluarannya, maka hubungan tersebut dapat ditulis sebagai berikut :

$H = f(P)$ *[MBTU/H] atau* $F = f(P)$ *[$US/H]*

Pembahasan selanjutya akan berpedoman atas dasar fungsi biaya bahan bakar ($F = f(P)$ *[$US/H]*) pada keadaan tertentu biaya-biaya lain yang merupakan fungsi terhadap keluaran pusat listrik dimasukkan ke dalam perhitungan biaya bahan bakar, misalnya biaya perawata (maintenance cost), upah pekerja (labour cost), dan biaya pengoprasian (operational cost).



Gambar 2-3 : Diagram Karakteristik Masukan-Keluaran Pembangkit Listrik Termal

Data yang diperlukan untuk menggambarkan diagram fungsi karakteristik masukan-keluaran dapat diperoleh dari perhitungan pada saat perencanaan atau tes yang telah dilakukan terhadap unit pembangkit yang bersangkutan.

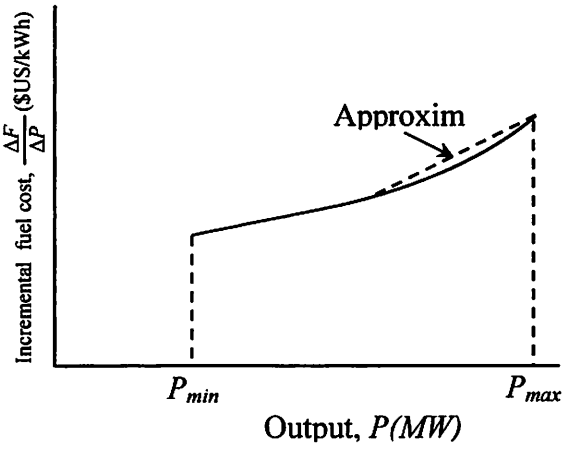
2.2.2 Karakteristik Laju Tambahan Biaya Bahan Bakar

Karakteristik laju tambahan biaya bahan bakar atau *Incremental Fuel Cost Characteristic* adalah turunan pertama dari fungsi biaya bahan bakar F [$US/h] terhadap tingkat pembebanan P[MW] dari pusat listrik yang bersangkutan. Fungsi ini menunjukan besarnya kenaikan atau penurunan biaya bahan bakar untuk setiap satu satuan perubahan beban.

Secara luas fungsi biaya bahan bakar akan digunakan untuk menentukan pembebanan ekonomis dari sebuah pembangkit listrik tenaga termal. Tampak pada gambar 2 - 4 kurva laju tambahan biaya bahan bakar

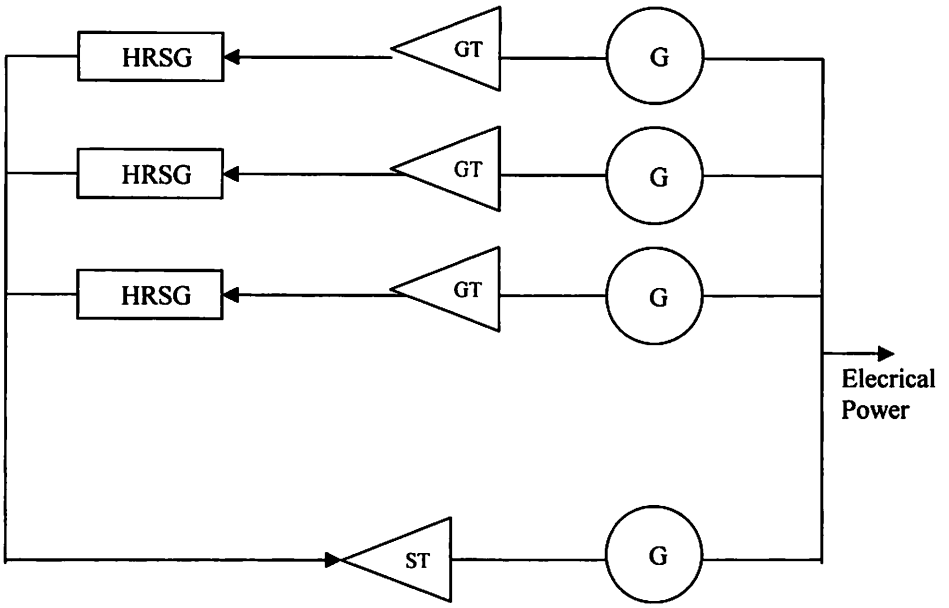
yang telah diidealkan melalui pendekatan linier dari sebuah pembangkit listrik termal.

Gambar 2 -4
Kurva Karakteristik Laju Tambahan Biaya Bahan Bakar



Pembangkit Listrik Tenaga Termal mulai dikembangkan dan menjanjikan prospek yang demikian luas. Pembangkit Listrik Tenaga Termal tersebut adalah Combined Cycle Power Plant (CCPP) atau dalam bahasa Indonesia disebut Pusat Listrik Tenaga Gas Uap (PLTGU).

Pada CCPP gas panas yang dibuang hasil pembakaran yang dibuang oleh turbin gas tersebut dimanfaatkan kembali melalui peralatan yang disebut *Heat Recovery Steam Generator* (HRSG) untuk menghasilkan siklus uap-air yang akan digunakan memutar sebuah turbin uap. Pembangkit listrik tenaga termal jenis ini memiliki keuntungan karena efisiensi siklus termalnya menjadi lebih tinggi. Pada gambar 2-5 tampak diagram segaris dari sebuah blok CCPP yang terdiri atas tiga turbin gas, tiga HRSG dengan satu turbin uap.



Gambar 2-5 : Diagram Segaris CCPP konfigurasi 3.3.1

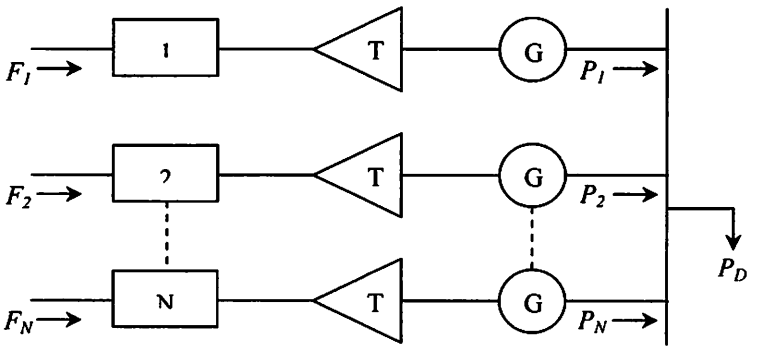
2.2.3 Pembebanan Ekonomis Pembangkit Listrik

Pembebanan ekonomis atau *economic dispatch* adalah pembagian pembebanan pada pembangkit-pembangkit yang ada dalam sistem tenaga listrik, secara optimal dan ekonomis pada harga beban tertentu. Komponen terbesar dari biaya pembangkitan adalah biaya bahan bakar. Oleh sebab itu dengan dilakukannya *economic dispatch* berarti pula didapatkan biaya bahan bakar pembangkitan yang paling murah.

Sistem tenaga listrik dengan mengabaikan rugi-rugi transmisi dapat dilihat pada gambar 2-6. Sistem ini memperlihatkan pembangkit termal yang terdiri atas N buah unit yang dihubungkan pada sebuah bus bar untuk melayani total beban sebesar P_D. Masukan untuk setiap unit ke-i adalah F_i yang menyatakan tingkat biaya bahan bakar dari masing-masing unit, dan daya

keluaran dari masing-masing unit P_i adalah daya listrik yang dibangkitkan oleh tiap-tiap unit.

Biaya total F_T yang ditanggung sistem adalah jumlah biaya dari tiap-tiap unit pembangkit. Dan batasan yang paling penting dari pengoperasian pembangkit termal tersebut adalah daya listrik yang dihasilkan harus sama dengan besarnya beban konsumen.



Gambar 2-6 : N Unit Pembangkit Termal Melayani Beban *PD*

Fungsi obyektif (objective function) F_T sama dengan biaya bahan bakar total yang dibutuhkan untuk melayani beban dari pusat listrik tersebut. Dengan mengabaikan rugi-rugi yang terjadi pada jaringan transmisi, gambaran diatas dapat pula dijabarkan secara matematis separti dibawah ini.

Biaya bahan bakar total pada Pembangkit Termal dengan N Blok adalah :

$$F_T = F_1 + F_2 + ... + F_N$$

$$F_T = \sum_{i=1}^{N} F_i(P_i) \quad \text{.....(2.1)}$$

Dan daya listrik yang dihasilkan oleh setiap Blok untuk melayani beban total adalah:

$$P_R = \sum_{i=1}^{N} P_i \quad \text{.....(2.2)}$$

$$P_R - \sum_{i=1}^{N} P_i = \phi = 0 \quad \text{.....(2.3)}$$

dimana P_R = kebutuhan beban, dan

P_i = jumlah daya yang dihasilkan.

Penyelesaian permasalahan optimasi terkendala seperti ini dapat diselesaikan dengan metode yang menyangkut Fungsi Lagrange:

$$\text{L} = F_t + \lambda\phi \quad \text{.....(2.4)}$$

atau

$$\text{L} = \sum_{i=1}^{N} F_i(P_i) + \lambda\left(P_R - \sum_{i=1}^{N} P_i\right) \quad \text{.....(2.5)}$$

dimana λ = Lagrange Mutiplier.

Bila L diturunkan terhadap λ maka diperoleh :

$$(\partial L / \partial P_i) = \frac{dF_i(P_i)}{dP_i} - \lambda \quad \text{.....(2.6)}$$

Untuk mencari harga optimal fungsi Lagrange terhadap P_i, persyaratan yang harus dipenuhi adalah $(\partial L / \partial P_i) = 0$

atau $\dfrac{dF_i(P_i)}{dP_i} - \lambda = 0$

atau $\dfrac{dF_i(P_i)}{dP_i} = \lambda$ ..... (2.7)

2.2.4 Penyeleseian *Economic Dispatch* Dengan Metode Iterasi Lamda.

Pada metode iterasi lamda (λ), lamda ditentukan dahulu kemudian dihitung harga keluaran untukn tiap-tiap unit pembangkit. Bila jumlah dari daya keluaran tiap-tiap unit pembangkit tidak sama dengan daya beban maka dilakukan perhitungan kembali harga lamda berikutnya untuk mendapatkan lamda berikutnya yang mendekati solusi sebenarnya, setelah beberapa kali iterasi.

2.3 Kendala Operasi PLTGU

PLTGU pada prinsipnya merupakan gabungan antara PLTG dan PLTGU. Pembangunan PLTGU dimaksudkan untuk mengurangi kendala-kendala operasi yang dihadapi PLTG dan PLTU apabila masing-masing beroperasi sendiri-sendiri (independent) namun karena turbin uap pada PLTGU masih dapat dioprasikan meskipun turbin uapnya dimatikan, maka dalam konfigurasi operasi yang demikian kendala-kendala oprasi PLTGU pada dasarnya sama dengan kendala-kendala operasi PLTG.

Hal yang diuntungkan dari pengoperasian PLTGU adalah kapasitas maksimum yang meningkat dan kecepatan melakukan perubahan beban. Untuk tingkat bahan bakar yang sama, tentunya PLTGU akan dapat membangkitkan daya listrik yang lebih besar daripada PLTU dan PLTG.

Ciri-ciri spesifik dari operasi PLTGU adalah:

- PLTU hanya dapat membangkitkan daya listrik bila PLTG beroprasi.

- Kurva masukan-keluaran PLTGU bukan merupakan penjumlahan kurva masukan-keluaran dari PLTG dan PLTU, jadi sifatnya spesifik.

Sedangkan mengenai kecepatan perubahan beban lebih disebabkan adanya blok PLTG, sehingga apabila terjadi kenaikan beban pada Blok PLTG, Blok tersebut dapat merespon perubahan dengan cepat tanpa mengganggu operasional blok PLTU-nya.

Sistem yang demikian ini sangat menguntungkan bagi sebuah pembangkit termal. Selain karena efesiensi total pusat listrik tersebut meningkat , juga pusat listrik tersebut dapat dioprasikan untuk memikul beban dasar, selain juga memikul beban menengah, dan sewaktu-waktu dapat dioprasikan untuk memikul beban puncak yang perubahan bebannya cukup fluktuatif. Tetapi bagaimanapun hal ini tetap tergantung pada biaya bahan bakar yang diperlukan.

2.4 Metode Genetic Algorithm

Algoritma genetika adalah suatu algoritma yang meniru prinsip evolusi alam sebagai metode untuk memecahkan masalah optimasi parameter. Prinsip yang mendasari algoritma genetika pertama kali dikembangkan oleh John Holland pada tahun 1962. Teori algoritma genetik didasari oleh teori evolusi Darwin. Landasan algoritma genetik terinspirasi dari mekanisme seleksi alami, dimana individu yang lebih kuat memiliki kemungkinan untuk menjadi pemenang dan mempunyai kesempatan hidup yang lebih besar di dalam lingkungan yang kompetitif.

Algoritma genetika bekerja dengan populasi string dan melakukan proses pencarian optimal secara paralel. Dengan menggunakan operator genetika algoritma genetik akan melakukan rekombinasi antar individu.

Elemen dasar tertentu yang yang diproses algoritma genetika adalah string (kromosom) dengan panjang tertentu yang tersusun dari rangkaian substring (gen), dan biasanya merupakan kode biner (0,1). Pada substring (gen) dapat diasumsikan suatu nilai biner yang dinamakan **allele.**

Penggambaran sebuah substring kromosom bit biner ditunjukkan oleh gambar berikut ini:

| 1 0 1 0 | → string kromosom

↓

gen

Pada makalah ini algoritma genetika digunakan untuk memecahkan masalah menentukan pola pembebanan bagi masing masing blok yang terdapat di Combined Cycle Power Plant (CCPP). Di PT PLN PJB II Unit Pembangkitan Gresik

2.4.1 Langkah – langkah dalam Genetic Algorithm

Pada dasarnya *Genetic Algorithm* terdiri dari tiga langkah. Terjadi pengulangan sampai diperoleh solusi hasil yang diinginkan (maksimum).

Langkah – langkah tersebut adalah:

1. Pemilihan inisial populasi dari solusi percobaan secara acak. Banyaknya solusi pada populasi yang relevan pada kecepatan optimasi, tapi tidak ada batasan yang digunakan.
2. Masing – masing solusi diulang sampai diperoleh populasi baru. Masing – masing solusi offspring dimutasi berdasarkan distribusi jenis – jenis

mutasi, berkisar antara kecil keadaan ekstrim dengan suatu jenis rangkaian mutasi. Mutasi ini ditentukan oleh parentnya.

3. Masing – masing solusi offspring ditentukan oleh perhitungan fitnessnya. Yang perlu digarisbawahi GA menggunakan *CROSSOVER*

2.4.2 Economic Dispatch dengan Metode Genetic Algorithm

Genetic Algorithm secara konseptual didasarkan pada genetika alami dan mekanisme evolusi yang bekerja pada populasi solusi dan berlawanan dengan teknik penelitian lainnya yang bekerja secara tunggal. Penelitian bukan pada ruang solusi parameter biner, tetapi pada bit string dekodenya, mereka meniru genetika kromosom alami dengan mengaplikasikan operator seperti genetika dalam penelitian untuk memperoleh optimum global.

Aspek paling menarik dari Genetic Algorithm adalah walaupun mereka tidak membutuhkan pengetahuan sebelumnya dan mereka tidak membutuhkan ruang terbatas seperti kehalusan, kecembungan dari fungsi untuk dioptimumkan, mereka menunjukan kinerja yang sangat baik untuk mayoritas problem yang diaplikasikan. Mereka hanya membutuhkan fungsi evaluasi untuk menentukan nilai kualitas (nilai kesesuaian) untuk setiap solusi yang dihasilkan. Ciri menarik lainnya adalah mereka secara inheren paralel (solusi secara individual tidak berhubungan satu sama lain), oleh karena itu implementasi mereka pada mesin paralel secara signifikan mangurangi waktu CPU yang dibutuhkan.

2.4.3. Mekanisme Genetic Algorithm dengan combined cycle Power Plant

A. Pengkodean atau Representasi

Langkah pertama kali yang dilakukan dalam penggunaan GA adalah melakukan pengkodean atau reprentasi terhadap permasalahan yang akan dilakuakan.

Secara umum GA dibentuk oleh serangkaian kromosom yang ditandai dengan *xi (i=1,2...N)*. Setiap elemen dalam kromosom ini adalah variabel string yang disebut gen., berisi nilai-nilai allele. Variabel-variabel ini dapat dinyatakan dalam bentiuk bilangan real (*floating point*).

Selanjutnya beberapa kromosom dibentuk dan berkumpul membentuk populasi. Populasi inilah populasi awal bagi GA untuk awal melakukan pencarian.

B. Fungsi *Fitness* (Fungsi Evaluasi)

Dalam GA, sebuah fungsi *fitness f (x)* harus dirancang untuk masing-masing permasalahan yang akan diselesaikan. Dengan menggunakan kromosom tertentu, fungsi obyektif atau fungsi evaluasi akan mengevaluasi status masing –masing kromosom. Setiap gen *xi (I = 1,2,...N)* dipergunakan untuk menghitung *fk (x) (k = 1,2,...POPSIZE)*

Pada permulaan optimasi, biasanya nilai fitness masing-masing individu masih mempunyai rentang yang lebar. Seiring dengan bertambah besar generasi, beberapa kromosom mendominasi populasi dan mengakibatkan rentang nilai fitness semakin kecil.

C. Seleksi

Masalah yang paling mendasar pada proses ini adalah proses penyeleksiannya. Menurut teori Darwin proses seleksi individu adalah : "individu terbaik akan tetap hidup dan akan menghasilkan keturunan".

D. Crossover (pindah silang)

Crossover adalah operator genetik yang utama dimana ini bekerja dengan mengambil dua individu dan memotong string kromosom pada posisi yang terpilih secara random.

E. *Mutation* (Mutasi)

Operator mutasi digunakan untuk memodifikasi satu atau lebih nilai gen dalam individu. Cara kerjanya dengan membangkitkan sebauh nilai random r_k dimana k = 1,2,...,*NVAR* (panjang kromosom). Probabilitas mutasi (*Pm*) ditentukan dan digunakan untuk mengendalikan frekuensi operator mutasi. Apabila nilai random r_k , P_m maka gen ke-k kromosom tersebut terpilih untuk mengalami mutasi.

F. Konvergensi

Konvergensi merupakan kondisi yang dicapai pada saat populasi kehilangan kondisi keanekaragaman. Algoritma genetik dapat mencapai kondisi konvergen meskipun tidak dapat dipastikan bahwa solusi yang diperoleh merupakan solusi yang optimal. Kondisi kesalahan yang sering

terjadi adalah konvergensi yang terlalu cepat dimana populasi tidak memperoleh solusi yang tidak optimal.

2.5 Algorithma Program

Algoritma GA

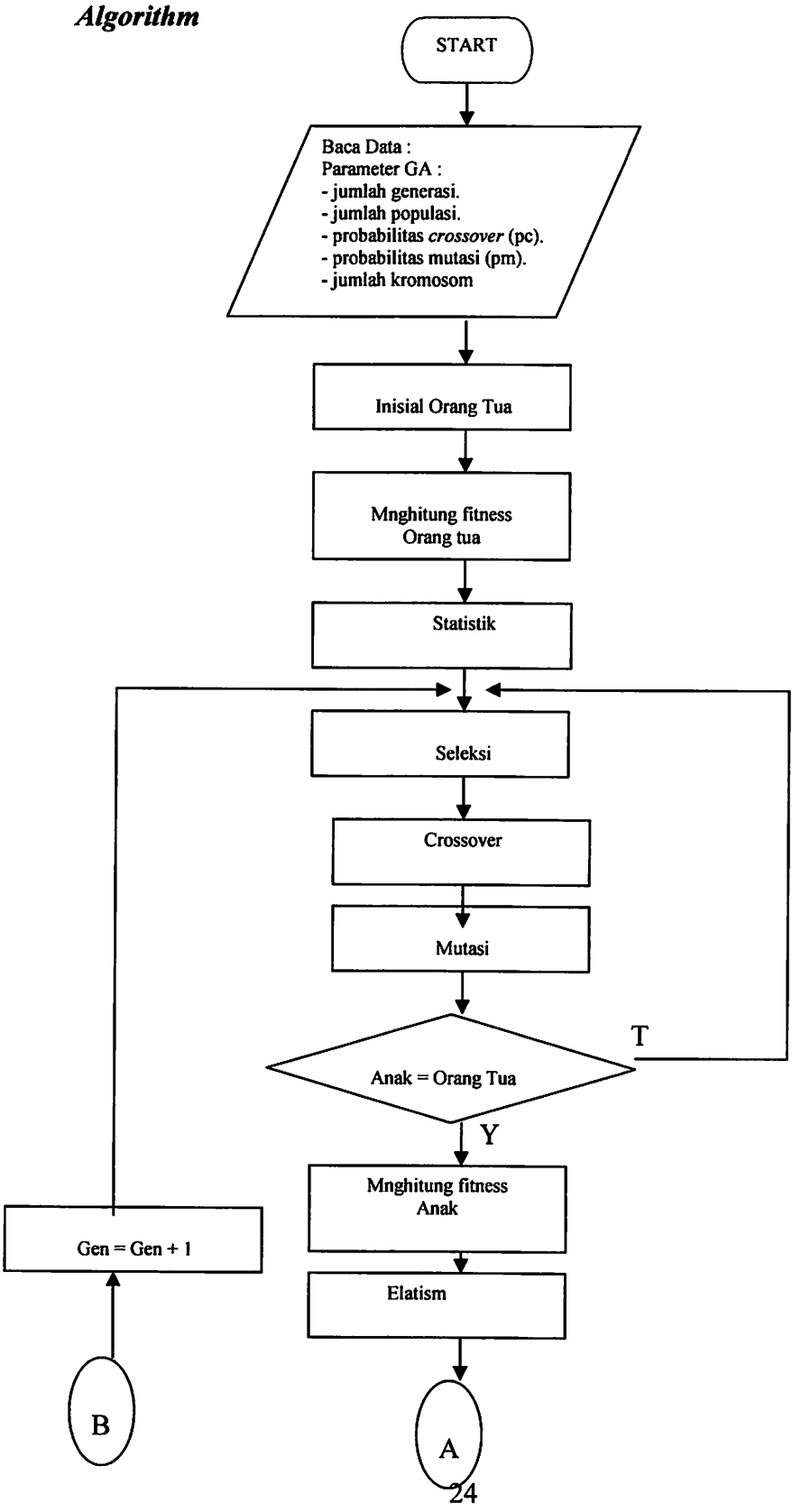
1. Masukkan parameter GA :
   - jumlah generasi.
   - jumlah populasi.
   - probabilitas *crossover* (Pc).
   - probabilitas mutasi (pm).
   - jumlah kromosom
2. Menentukan inisial orang tua.
3. Menghitung fitness orang tua.
4. Melakukan proses statistik.
5. Melakukan proses seleksi.
6. Melakukan proses *crossover.*
7. Melakukan proses mutasi.
8. Jumlah anak, apakah = orang tua.
9. Jika tidak, kembali ke langkah 6.
10. Jika ya, menghitung fitness anak.
11. Melakukan proses *elitism.*
12. Melakukan proses mutasi.
13. Menentukan apakah gen. Tersebut = gen. Maksimal.
14. Jika tidak kembali ke langkah 5 dengan menambah iterasi.

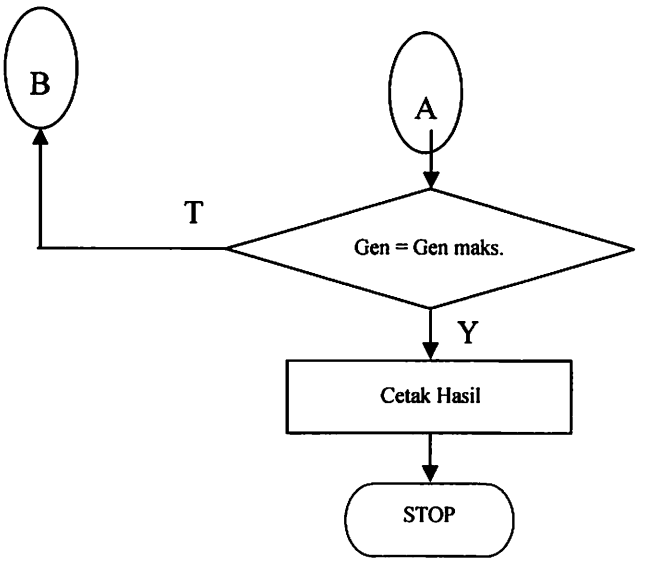
2.5.2 Algoritma Fungsi *Fitness*

1. Memasukkan input kromosom yang dilakukan secara random (acak) menggunakan metode *Genetic Algorithm*
2. Menghitung objective function total fuel cost
3. Selesei

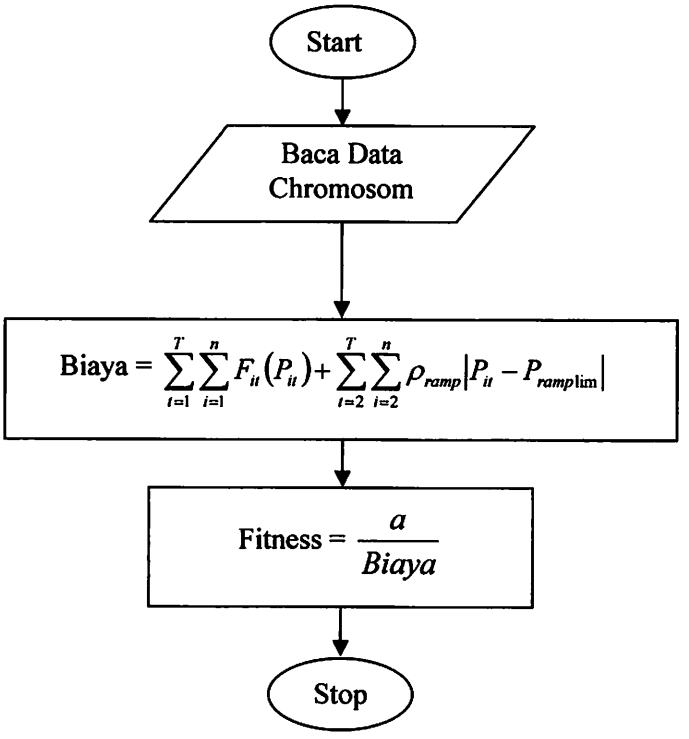
2.6 Flowchart

2.6.1 Flowchart Economic Dispatch menggunakan Metode *Genetic Algorithm*





2.6.2 Flowchart Fungsi *Fitness*



BAB III

Pembangkit Listrik Tenaga Gas Uap

3.1. PLTGU Gresik

Pada pertengahan tahun 1978 untuk pertama kali di kota Gresik dibangun Pusat Listrik Tenaga Gas (PLTG) dengan total kapasitas 40 MW yang terdiri dari 2 mesin pembangkit masing-masing berkapasitas 20 MW, dalam wilayah unit kerja PLN sektor Perak. Awal tahun 1981 di lokasi yang sama dibangun lagi 2 pusat Listrik Tenaga Uap (PLTU) yang berkapasitas masing-masing 100 MW. Berdasarkan SK direksi PLN no. 023 / DIR / 1981 tgl 16 Maret 1981, kedua pusat listrik tersebut di jadikan satu wilayah kerja tersendiri denga nama PLN sektor Gresik. Penambahan mesin pembangkit pun terus dilaksanakan sejalan dengan tuntutan kebutuhan tenaga listrik hingga pada awal tahun 1989 total kapasitas PLN sektor Gresik menjadi ± 702 MW.

Pada pertengahan tahun 1996 berdasarkan SK Dirut PT PLN PJB II no.023.K/Dir/1992, tanggal 4 Pebruari 1992, dibentuk sektor Gresik Baru (SGRB) dengan total kapasitas 1578,78 MW yang terdiri atas 3 blok daur ganda (combined cycle) yang masing-masing blok terdiri atas 3 unit GT 9 (per GT berkapasitas 112,45 MW) dan 1 unit ST yang berkapasitas 188,98 MW.

Pada tahun 1997 terjadi penggabungan antara sektor Gresik baru dengan nama baru yaitu PT PLN PJB II Unit Pembangkitan Gresik.

3.2 Konfigurasi dan Kapasitas Unit Pembangkitan Gresik

PLTGU Unit pembangkitan Gresik terdiri dari 3 blok, masing-masing blok terdiri atas 3 buah Turbin gas dengan 3 Generator Turbo, 3 HRSG dan 1 buah Turbin uap dengan 1 generator turbo. Blok 1 dan blok 2 dapat menggunakan bahan bakar berupa High Speed Diesel (HSD), dan Gas Alam atau Natural Gas (NG), sedangkan Blok 3 hanya berbahan bakar gas.

Terdapat beberapa kemungkinan konfigurasi untuk mengoprasikan Blok PLTGU Unit Pembangkitan Gresik, yaitu:

- Konfigurasi 1 GT, 1 HRSG, dan 1 ST
- Konfigurasi 2 GT, 2 HRSG, dan 1 ST
- Konfigurasi 3 GT, 3 HRSG, dan 1 ST

Tabel 3.1

Tipe Pembangkit di Unit Pembangkitan Gresik

NO	BLOK	FUEL	KAPASITAS	PABRIKAN
	Gas Turbin Power Plant			
I	GT 1 GT 2 GT 3 GT 4 GT 5	HSD/NG HSD/NG HSD HSD/NG HSD/NG	20 MW 20 MW 21 MW 20 MW 21 MW	Alsthom-France Alsthom-France Alsthom-France GE- USA GE- USA
	Steam Turbin Power Plant			
II	ST 1 ST 2 ST 3 ST 4	HSD HSD HSD/NG HSD/NG	100 MW 100 MW 200 MW 200 MW	Toshiba-Japan Toshiba-Japan Toshiba-Japan Toshiba-Japan

III	Combined Cycle Power Plant			
	Blok 1			
	GT 1.1	NG HSD	112,45 MW 100,98 MW	Mitsubishi
	GT 1.2	NG HSD	112,45 MW 100,98 MW	Mitsubishi
	GT 1.3	NG HSD	112,45 MW 100,98 MW	Mitsubishi
	ST 1.0	-	188,92 MW	Mitsubishi
	Blok 2			
	GT 2.1	NG HSD	112,45 MW 100,98 MW	Mitsubishi
	GT 2.2	NG HSD	112,45 MW 100,98 MW	Mitsubishi
	GT 2.3	NG HSD	112,45 MW 100,98 MW	Mitsubishi
	ST 2.0	-	188,92 MW	Mitsubishi
	Blok 3			
	GT 3.1	NG	112,45 MW	Mitsubishi
	GT 3.2	NG	112,45 MW	Mitsubishi
	GT 3.3	NG	112,45 MW	Mitsubishi
	ST 3.0	-	188,92 MW	Mitsubishi
	TOTAL KAPASITAS		2280 MW	

Sumber data operational PLTGU Gresik

Kapasitas terpasang PLTGU Unit Pembangkitan Gresik :

Turbin Gas : 112,45 MW (BBG) dan 100,98 MW (BBM)

Turbin Uap : 188,92 MW (BBG)

Kapasitas total per Blok, Konfigurasi 3.3.1 : 526,26 MW

Kapasitas total PLTGU (BBG) : 1578,78 MW

Kapasitas Minimum :

Kapasitas total per Blok, konfigurasi 3.3.1 : 125 MW

Kapasitas total PLTGU : 375 MW

Kapasitas Maksimum :

Turbin Gas : 115 MW (BBG)

Turbin Uap : 189 MW (BBG), Konfigurasi 3.3.1

Kapasitas total per Blok, konfigurasi 3.3.1 : 538, 2 MW

Kapasitas total PLTGU (BBG) : 1614,6 MW

3.3 Produksi Energi Listrik pada Pusat Listrik Tenaga Gas-Uap Unit Pembangkitan Gresik

Konsep dari sebuah PLTGU adalah pendayagunaan udara panas (500-550 °C) yang telah dihasilkan leh sebuah atau beberapa turbin gas untuk digunakan sebagai sumber energi (kalor) guna memanaskan air di ketel agar dapat dihasilkan uap untuk mengoprasikan turbin uap. Pemanfaatan kembali uap panas yang sebelumnya dibuang ke atmosfir tersebut dapat meningkatkan efisiensi Pembangkitan Listrik Tenaga Termal.

Komponen-komponen utama yang terdapat di Unit Pembangkitan Gresik meliputi:

- Blok pengolahan bahan bakar

- Blok Turbin Gas
- Blok Turbin Uap
- Blok kendali operasi

Berikut ini akan dijelaskan secara umum tentang peralatan-peralatan utama yang terdapat di PLTGU Unit Pembangkitan Gresik.

a) Blok pengolahan bahan bakar PLTGU Unit Pembangkitan Gresik.

Berfungsi untuk memperbaiki mutu gas alam yang akan digunakan sebagai bahan bakar utama PLTGU, karena mutu bahan bakar tersebut mungkin menurun selama proses pengiriman.

b) Blok Turbin Gas PLTGU Unit Pembangkitan Gresik.

Selain berfungsi sebagai peralatan konversi energi listrik dari bahan bakar, juga merupakan sumber energi panas yang akan dimanfaatkan oleh sirkit HRSG.

c) Blok kendali operasi PLTGU Unit Pembangkitan Gresik

Berfungsi sebagai pusat sistem kontrol bagi seluruh Blok PLTGU. Pengendalian operasi PLTGU berlangsung secara otomatis yang dipantau melelaui terminal komputer yang terpasang di Control Centre Room (CCR). Didalam CCR terdapat empat buah Block Control Desk (BCD) yang pada prinsipnya merupakan terminal input-output dari komputer tersebut unutk memantau operasi PLTGU. Tugas utama

komputer adalah untuk menyelenggarakan supervisi dan pengendalian operasi PLTGU. Untuk menyelenggarakan tugas supervisi dan pengendalian tersebut, komputer mengumpulkan data dan informasi dari sistem yang kemudian diolah menurut prosedur tertentu, dimana prosedur ini diatur oleh perangkat lunak (software) komputer. Informasi yang dikumpulkan komputer berasal dari peralatan-peralatan telemetri yang menghasilkan sinyal analog, melalui Direct Digital Controler (DDC) informasi tersebut diterima oleh komputer sebagai sinyal digital untuk kemudian diolah dan ditampilkan sebagai informasi real time yang perlu diketahui operator yang bertugas. Selain informasi real time, komputer juga menyimpan data-data masa lalu (history) yang direkam dalam suatu media penyimpan informasi tertentu.

3.4 Pola Pembebanan Pada PLTGU Unit Pembangkitan Gresik

Pola pembebanan di PLTGU Unit Pembangkitan Gresik dibentuk dengan menggunakan asumsi bahwa blok yang beroperasi dengan konfigurasi sama. Secara fisik hal ini disebabkan oleh peralatan yang digunakan, pabrikan, standart material, kapasitas terpasang, bahkan tahun pembuatan dan tahun pengoperasian masing-masing Blok tidak jauh berbeda. Dengan asumsi diatas maka diadakan test untuk mendapatkan data aliran bahan bakar pada setiap tingkat beban masing-masing blok.

Berikut ini adalah tabel dari pola pembebanan pada konfigurasi 3.3.1 yang mana pada periode tertentu telah diterapkan di PLTGU Unit Pembangkitan Gresik. Tabel ini merupakan acuan bagi operator operasi yang bertugas pada Control Centre Room (CCR) untuk membagi beban per blok untuk memenuhi permintaan P3B.

Tabel 3-2

Tabel Distribusi Beban Blok
Pada Konfigurasi 3.3.1 – 3.3.1 – 3.3.1

Beban Total 375 MW – 725 MW

No	Beban Total	Beban Blok-1	Beban Blok-2	Beban Blok-3	BBG Blok-1	BBG Blok-2	BBG Blok-3	Total Aliran BBG
	(MW)				(KNM^3/H)			
1	**375**	125	125	125	39.21	39.21	39.21	117.63
2	**400**	125	125	150	39.21	39.21	44.78	123.20
3	**425**	125	125	175	39.21	39.21	47.98	126.40
4	**450**	125	125	200	39.21	39.21	52.31	130.73
5	**475**	125	125	225	39.21	39.21	55.62	134.04
6	**500**	125	125	250	39.21	39.21	58.65	137.07
7	**525**	125	125	275	39.21	39.21	62.98	141.40
8	**550**	125	125	300	39.21	39.21	65.54	143.96
9	**575**	125	125	325	39.21	39.21	70.12	148.54
10	**600**	125	125	350	39.21	39.21	74.03	152.45
11	**625**	125	125	375	39.21	39.21	77.96	156.38
12	**650**	125	225	300	39.21	55.62	65.54	10.37
13	**675**	125	250	300	39.21	58.65	65.54	163.40
14	**700**	125	275	300	39.21	62.98	65.54	167.73
15	**725**	125	300	300	39.21	65.54	65.54	170.29

Sumber : Data operasi PLTGU Gresik

Tabel 3-3

Tabel Distribusi Beban Blok
Pada Konfigurasi 3.3.1 – 3.3.1 – 3.3.1

Beban Total 750 MW – 1100 MW

No	Beban Total	Beban Blok-1	Beban Blok-2	Beban Blok-3	BBG Blok-1	BBG Blok-2	BBG Blok-3	Total Aliran BBG
	(MW)				(KNM^3/H)			
1	**750**	125	300	325	39.21	65.54	70.12	174.87
2	**775**	125	300	350	39.21	65.54	74.03	178.78
3	**800**	125	300	375	39.21	65.54	77.96	182.71
4	**825**	225	300	300	55.62	65.54	65.54	186.70
5	**850**	250	300	300	58.65	65.54	65.54	189.73
6	**875**	275	300	300	62.98	65.54	65.54	194.06
7	**900**	300	300	300	65.54	65.54	65.54	196.62
8	**925**	300	300	325	65.54	65.54	70.12	201.20
9	**950**	300	300	350	65.54	65.54	74.03	205.11
10	**975**	300	300	375	65.54	65.54	77.96	209.04
11	**1000**	300	350	350	65.54	74.03	74.03	213.60
12	**1025**	300	350	375	65.54	74.03	77.96	217.53
13	**1050**	300	375	375	65.54	77.96	77.96	221.46
14	**1075**	350	350	375	74.03	74.03	77.96	226.02
15	**1100**	375	375	350	77.96	77.96	74.03	229.95

Sumber : Data operasi PLTGU Gresik

Tabel 3-4

Tabel Distribusi Beban Blok
Pada Konfigurasi 3.3.1 – 3.3.1 – 3.3.1

Beban Total 1125 MW – 1500 MW

No	Beban Total	Beban Blok-1	Beban Blok-2	Beban Blok-3	BBG Blok-1	BBG Blok-2	BBG Blok-3	Total Aliran BBG
	(MW)				(KNM^3/H)			
1	**1125**	375	375	375	77.96	77.96	77.96	233.88
2	**1150**	375	375	400	77.96	77.96	83.04	238.96
3	**1175**	300	375	500	65.54	77.96	83.04	243.56
4	**1200**	350	375	475	74.03	77.96	95.81	247.80
5	**1225**	375	375	475	77.96	77.96	95.81	251.73
6	**1250**	375	375	500	77.96	77.96	100.06	255.98
7	**1275**	375	400	500	77.96	83.04	100.06	261.06
8	**1300**	350	475	475	74.03	95.81	95.81	265.65
9	**1325**	375	475	475	77.96	95.81	95.81	269.58
10	**1350**	375	475	500	77.96	95.81	100.06	273.83
11	**1375**	375	500	500	77.96	100.06	100.06	278.08
12	**1400**	400	500	500	83.04	100.06	100.06	283.16
13	**1425**	475	475	475	95.81	95.81	95.81	287.43
14	**1450**	475	475	500	95.81	95.81	100.06	291.68
15	**1475**	475	500	500	95.81	100.06	100.06	295.93
16	**1500**	500	500	500	100.06	100.06	100.06	300.18

Sumber : Data operasi PLTGU Gresik

BAB IV

ANALISA DATA

4.1. Pendahuluan

PT PLN PJB II Unit Pembangkitan Gresik.dalam penyelenggaraan usaha ketenagalistrikan berdasarkan prinsip industri dan perniagaan yang sehat, dituntut mampu bersaing dan mampu memanfaatkan sebesar-besarnya peluang pasar dalam bidang tenaga listrik. Dalam hal tersebut, PT PLN PJB II Unit Pembangkitan Gresik harus menjaga effisiensi dan keadalan operasional penyediaan tenaga listrik dari pembangkit-pembangkit yang dimilikinya.

Dengan demikian merupakan suatu keharusan bagi seluruh jajaran PT PLN PJB II Unit Pembangkitan Gresik agar selalu berupaya untuk meningkatkan kondisi penyediaan tenaga listrik dari pembangkit agar lebih ekonomis, bermutu dan didukung keandalan yang tinggi.

4.2 Data Pembangkitan

PLTGU Unit pembangkitan Gresik terdiri dari 3 blok, masing-masing blok terdiri atas 3 buah Turbin gas dengan 3 Generator Turbo, 3 HRSG dan 1 buah Turbin uap dengan 1 generator turbo.

Tabel 4.1

Total Aliran BBG per tingkat beban

Pada Blok 1

No	Tingkat Beban	Total Aliran BBG	
		KNM^3/Hr	MBTU/Hr
1	150	46,55	1838,33
2	175	49,37	1939,69
3	200	53,11	2097,39
4	225	56,83	2244,30
5	250	59,67	2356,45
6	275	63,31	2460,71
7	300	65,25	2576,82
8	325	69,70	2752,55
9	350	73,77	2913,28
10	375	78,82	3112,72
11	400	83,59	3301,09
12	425	88,06	3477,62
13	450	92,69	3660,46
14	475	96,05	3793,15
15	500	99,98	3948,35

Sumber : Data test PLTGU Gresik

Catatan :

- 1 KNM^3 = 37,3258 MSCF
- HHV Bahan Bakar Gas = 1058,02 BTU/SCF

Tabel 4.2

Total Aliran BBG per tingkat beban

Pada Blok 2

No	Tingkat Beban	Total Aliran BBG	
		KNM^3/Hr	MBTU/Hr
1	150	43,56	1720,44
2	175	47,42	1872,92
3	200	51,28	2025,44
4	225	55,15	2177,91
5	250	59,01	2330,43
6	275	62,87	2482,91
7	300	66,73	2635,42
8	325	70,59	2787,94
9	350	73,92	2919,21
10	375	78,23	3089,53
11	400	82,73	3267,44
12	425	87,28	3447,13
13	450	91,83	3626,82
14	475	96,38	3806,46
15	500	100,93	3986,15

Sumber : Data test PLTGU Gresik

Catatan :

- 1 KNM^3 = 37,3258 MSCF
- HHV Bahan Bakar Gas = 1058,02 BTU/SCF

Tabel 4.3

Total Aliran BBG per tingkat beban

Pada Blok 3

No	Tingkat Beban	Total Aliran BBG	
		KNM^3/Hr	MBTU/Hr
1	150	41,44	1636,84
2	175	45,27	1787,78
3	200	49,09	1938,67
4	225	52,91	2089,61
5	250	56,73	2240,55
6	275	60,55	2391,48
7	300	64,37	2542,38
8	325	68,20	2693,32
9	350	72,31	2855,98
10	375	75,88	2996,73
11	400	80,22	3168,08
12	425	84,70	3344,96
13	450	89,18	3521,85
14	475	93,66	3698,77
15	500	98,13	3875,65

Sumber : Data test PLTGU Gresik

Catatan :

- 1 KNM^3 = 37,3258 MSCF
- HHV Bahan Bakar Gas = 1058,02 BTU/SCF

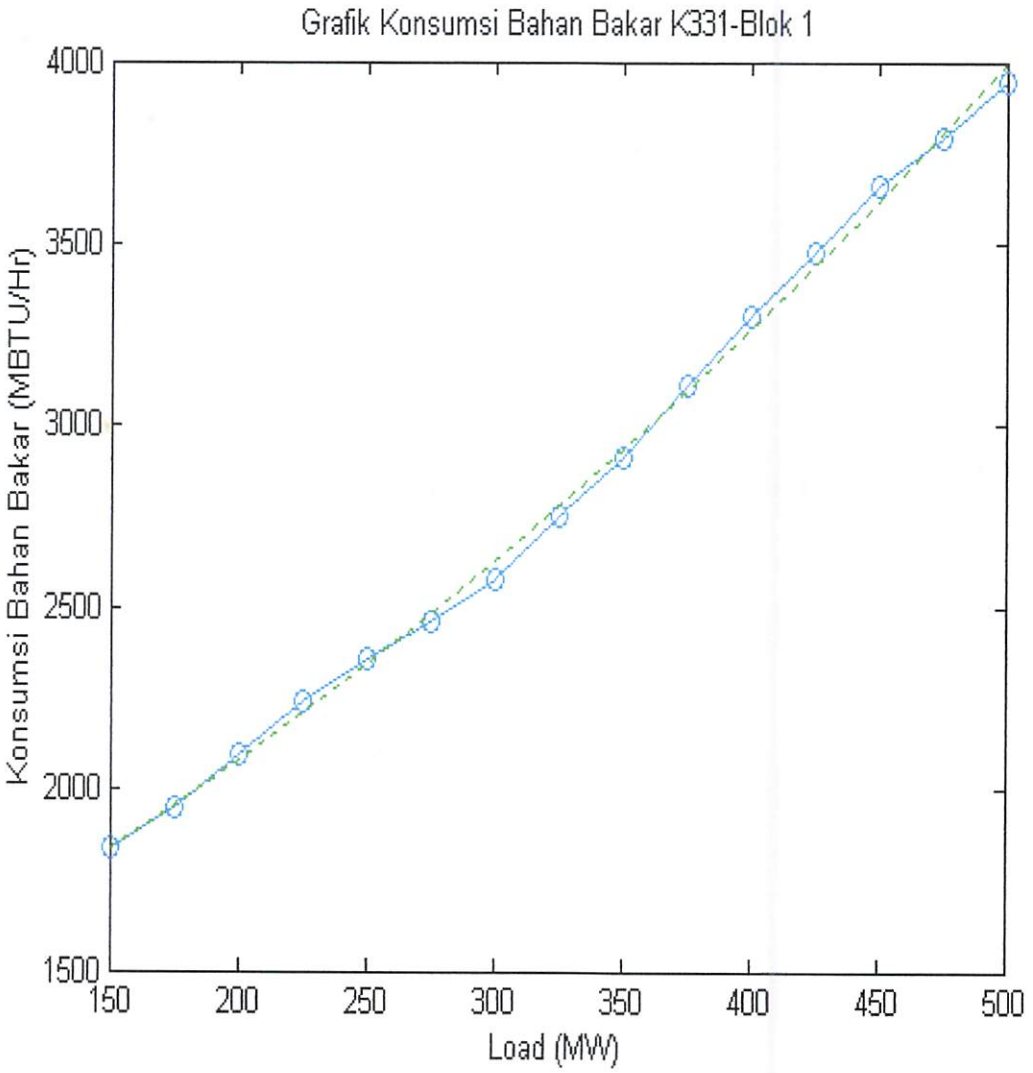
Data hasil perhitungan dilakukan sesuai dengan data dikumpulkan dan diolah. Misalnya nilai HHV bahan bakar gas rata-rata yang digunakan yaitu sebesar 1058,02 (BTU/SCF), adalah sesuai dengan hasil pengukuran nilai HHV rata-rata bahan bakar gas yang dilakukan oleh operator lapangan pada saat data dikumpulkan.

Selanjutnya data aliran bahan bakar gas masing-masing blok yang telah ditabulasikan diatas, diplot pada diagram pancar.

Grafik 4.1 :

Total aliran BBG per tingkat beban

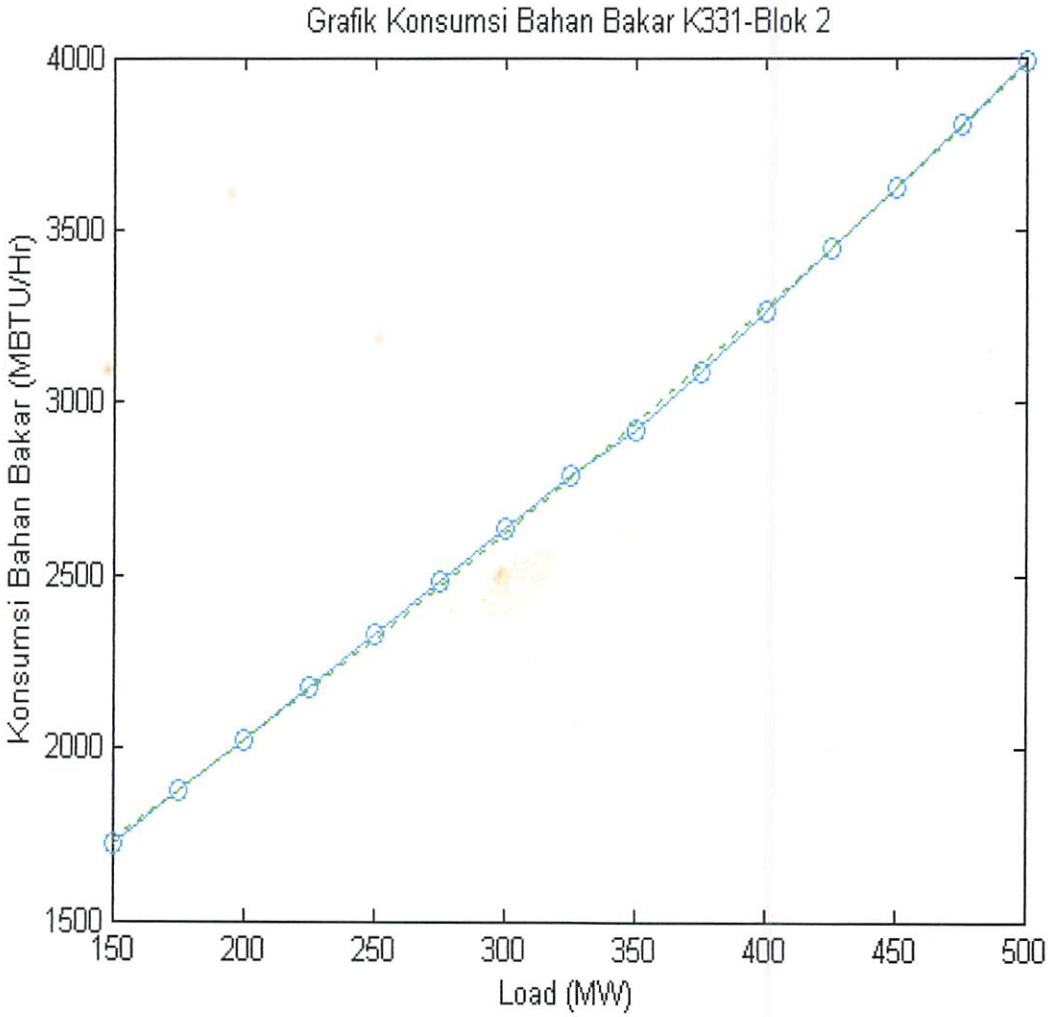
Pada Blok 1



Grafik 4.2 :

Total aliran BBG per tingkat beban

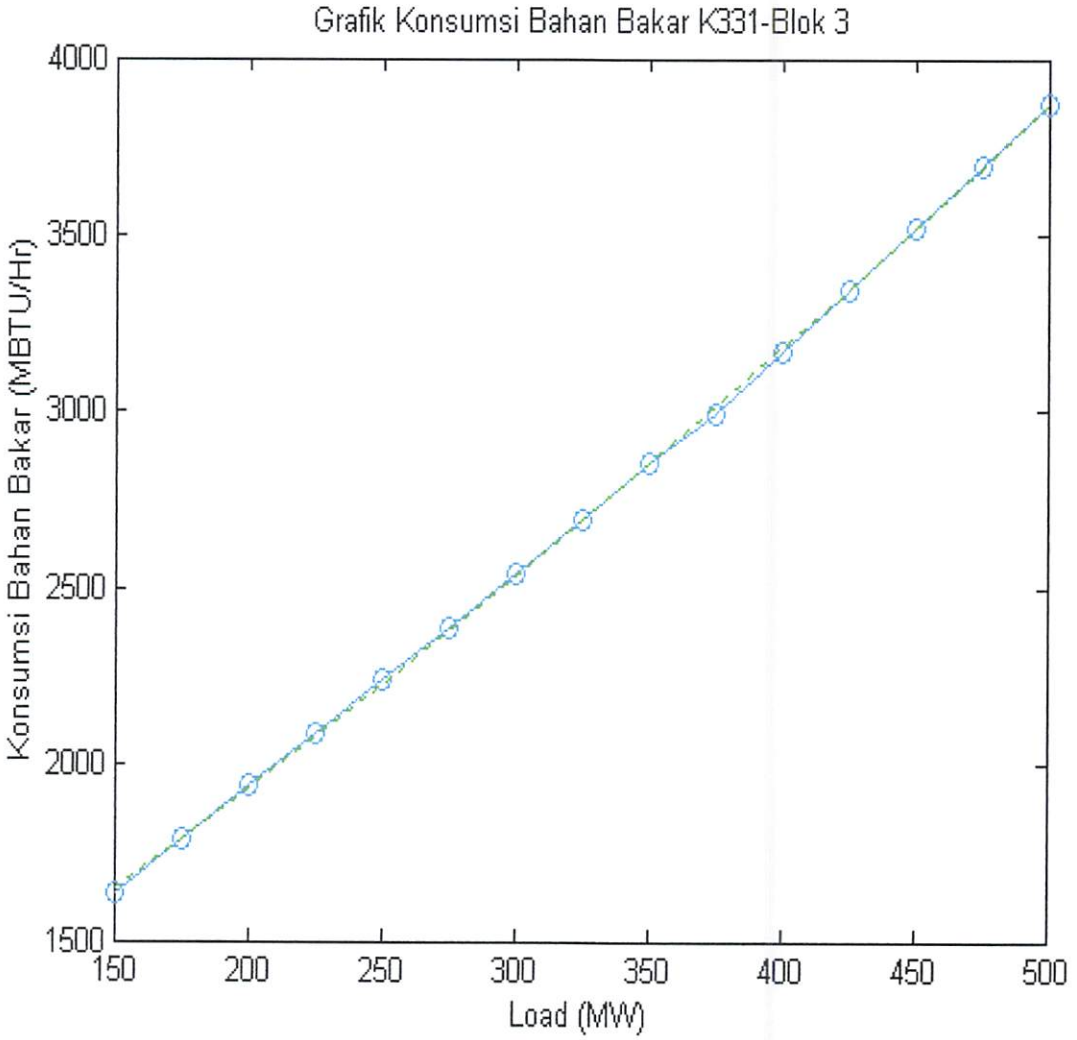
Pada Blok 1



Grafik 4.3 :

Total aliran BBG per tingkat beban

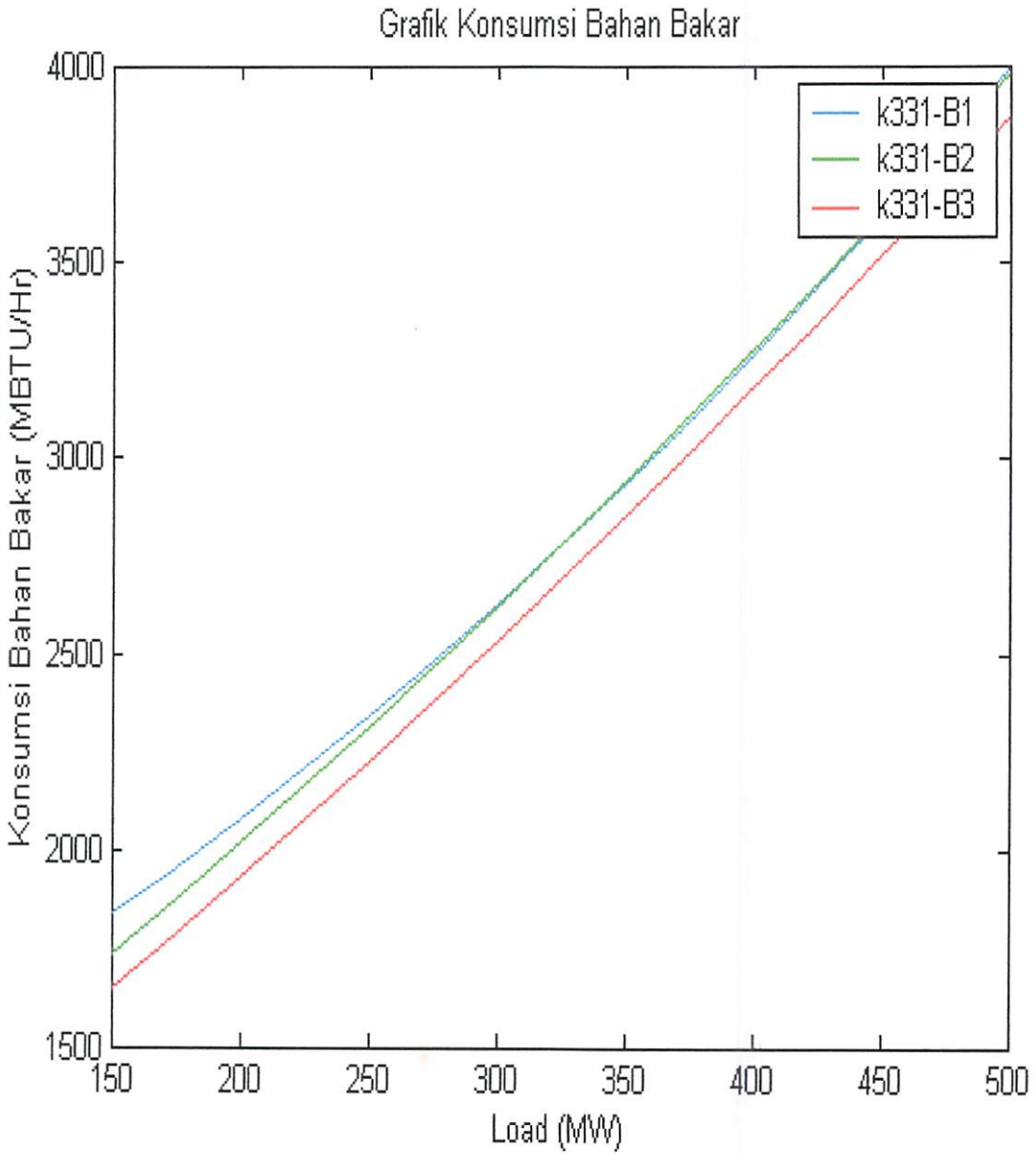
Pada Blok 3



Grafik 4.4 :

Total aliran BBG per tingkat beban

Pada Kombinasi 3.3.1



4.3 Perhitungan Persamaan Karakteristik Masukan-Keluaran(Bahan Bakar) masing-masing Blok Unit Pembangkitan PLTGU Gresik.

Melihat kondisi seperti yang telah diuraikan pada bab sebelumnya, maka langkah pertama yang perlu dilakukan adalah menentukan persamaan karakteristik masukan-keluaran (H = f (P)) dalam (MBTU/H) dan F = f (P) dalam ($ US/H).

Dari kurva aliran BBG sebagai fungsi beban (grafik 4.1, grafik 4.2, grafik 4.3) terlihat bahwa karakteristik masukan-keluaran sebuah Blok Pembangkit Listrik Termal sudah cukup teliti. Sebagai catatan, bahwa harga persatuan energi bahan bakar (natural gas) yang dikonsumsi PLTGU Gresik sebesar 2,53 ($ US/MBTU).

Hasil dari perhitungan sebagaimana terlampir adalah sebagai berikut :

- **Persamaan Karakteristik Masukan-Keluaran (Bahan Bakar) PLTGU Gresik dengan konfigurasi 3.3.1 pada Blok 1.**

$$\mathbf{F_1(P_1)} = 0{,}0112590\ \mathbf{P_1}^2 + 8{,}2427\ \mathbf{P_1} + 3{,}166\,.\,10^3 \left[\frac{\$US}{Hr}\right],$$

Maka persamaan laju tambahan biaya bahan bakar diperoleh dengan menurunkan persamaan biaya bahan bakar terhadap beban $\left(\frac{dF_1(P_1)}{dP_1}\right)$ sehingga menghasilkan :

$$\left(\frac{dF_1(P_1)}{dP_1}\right) = 0{,}022518\ \mathbf{P_1} + 8{,}2427 \left[\frac{\$US}{MWH}\right]$$

- **Persamaan Karakteristik Masukan-Keluaran (Bahan bakar) PLTGU Gresik dengan konfigurasi 3.3.1 pada Blok 2.**

$$\mathbf{F_2(P_2)} = 0{,}0062063\ \mathbf{P_2}^2 + 12{,}1771\ \mathbf{P_2} + 2{,}4282 \,.\, 10^3 \left[\frac{\$US}{Hr}\right],$$

Maka persamaan laju tambahan biaya bahan bakar diperoleh dengan menurunkan persamaan biaya bahan bakar terhadap beban $\left(\frac{dF_1(P_1)}{dP_1}\right)$ sehingga menghasilkan :

$$\left(\frac{dF_1(P_1)}{dP_1}\right) = 0{,}0214126\ \mathbf{P_2} + 12{,}1771 \left[\frac{\$US}{MWH}\right]$$

- **Persamaan Karakteristik Masukan-Keluaran (Bahan bakar) PLTGU Gresik dengan konfigurasi 3.3.1 pada Blok 3.**

$$\mathbf{F_3(P_3)} = 0{,}0053446\ \mathbf{P_3}^2 + 12{,}5668\ \mathbf{P_3} + 2{,}1676 \,.\, 10^3 \left[\frac{\$US}{Hr}\right],$$

Maka persamaan laju tambahan biaya bahan bakar diperoleh dengan menurunkan persamaan biaya bahan bakar terhadap beban $\left(\frac{dF_1(P_1)}{dP_1}\right)$ sehingga menghasilkan :

$$\left(\frac{dF_1(P_1)}{dP_1}\right) = 0{,}0106892\ \mathbf{P_3} + 12{,}5668 \left[\frac{\$US}{MWH}\right]$$

Contoh 1

Untuk beban **150** MW pada blok 1 maka biaya bahan bakarnya akan dapat diketahui dengan cara:

$$\mathbf{F_1(P_1)} = 0{,}0112590\ \mathbf{P_1}^2 + 8{,}2427\ \mathbf{P_1} + 3{,}166 \,.\, 10^3 \left[\frac{\$US}{Hr}\right],$$

$$\mathbf{F_1(150)} = 0{,}0112590\ \mathbf{150}^2 + 8{,}2427\ \mathbf{150} + 3{,}166 \,.\, 10^3 \left[\frac{\$US}{Hr}\right],$$

$$= \mathbf{4.655{,}73\ (\$\ US/Hr)}$$

Begitu juga sama cara perhitungannya pada blok 2,dan 3, disesuaikan dengan persamaan masing-masing blok.

Untuk mencari laju pertambahan bahan bakarnya maka:

$$\left(\frac{dF_1(P_1)}{dP_1}\right) = 0{,}022518\ \mathbf{P_1} + 8{,}2427 \left[\frac{\$US}{MWH}\right]$$

$$\left(\frac{dF_1(P_1)}{dP_1}\right) = 0{,}022518\ \mathbf{150} + 8{,}2427 \left[\frac{\$US}{MWH}\right]$$

$$\left(\frac{dF_1(P_1)}{dP_1}\right) = \mathbf{11{,}26\ (\$\ US/Hr)}$$

Untuk blok 2,dan 3 dicari dengan cara yang sama sesuai dengan persamaannya.

Dari persamaan biaya bahan bakar dan laju tambahan biaya bahan bakar didapatkan nilai masing-masing terhadap tingkat beban, kemudian ditabulasikan dan diplot dalam diagram pancarnya.

Tabel 4.4

Tabel Fungsi Biaya Bahan Bakar Masing-masing Blok
Terhadap Tingkat Beban

No	Beban Blok (MW)	Biaya Bahan Bakar ($ US/Hr)		
		Blok 1	Blok 2	Blok 3
1	150	4.655,73	4.394,41	4.172,86
2	175	4.953,28	4.749,26	4.530,45
3	200	5.264,90	5.111,88	4.894,73
4	225	5.590,59	5.482,25	5.265,68
5	250	5.930,36	5.860,37	5.643,32
6	275	6.284,19	6.246,26	6.207,64
7	300	6.652,11	6.639,90	6.418,64
8	325	7.034,09	7.041,30	6.816,32
9	350	7.430,15	7.450,46	7.220,68
10	375	7.840,29	7.867,37	7.631,72
11	400	8.264,49	8.292,05	8.049,44
12	425	8.702,77	8.724,48	8.473,84
13	450	9.155,13	9.164,67	8.904,92
14	475	9.621,55	9.612,62	9.342,69
15	500	10.102,06	10.068,32	9.787,13

Tabel 4.5

Tabel Fungsi Laju Tambahan Bahan Bakar Masing-masing Blok Terhadap Tingkat Beban

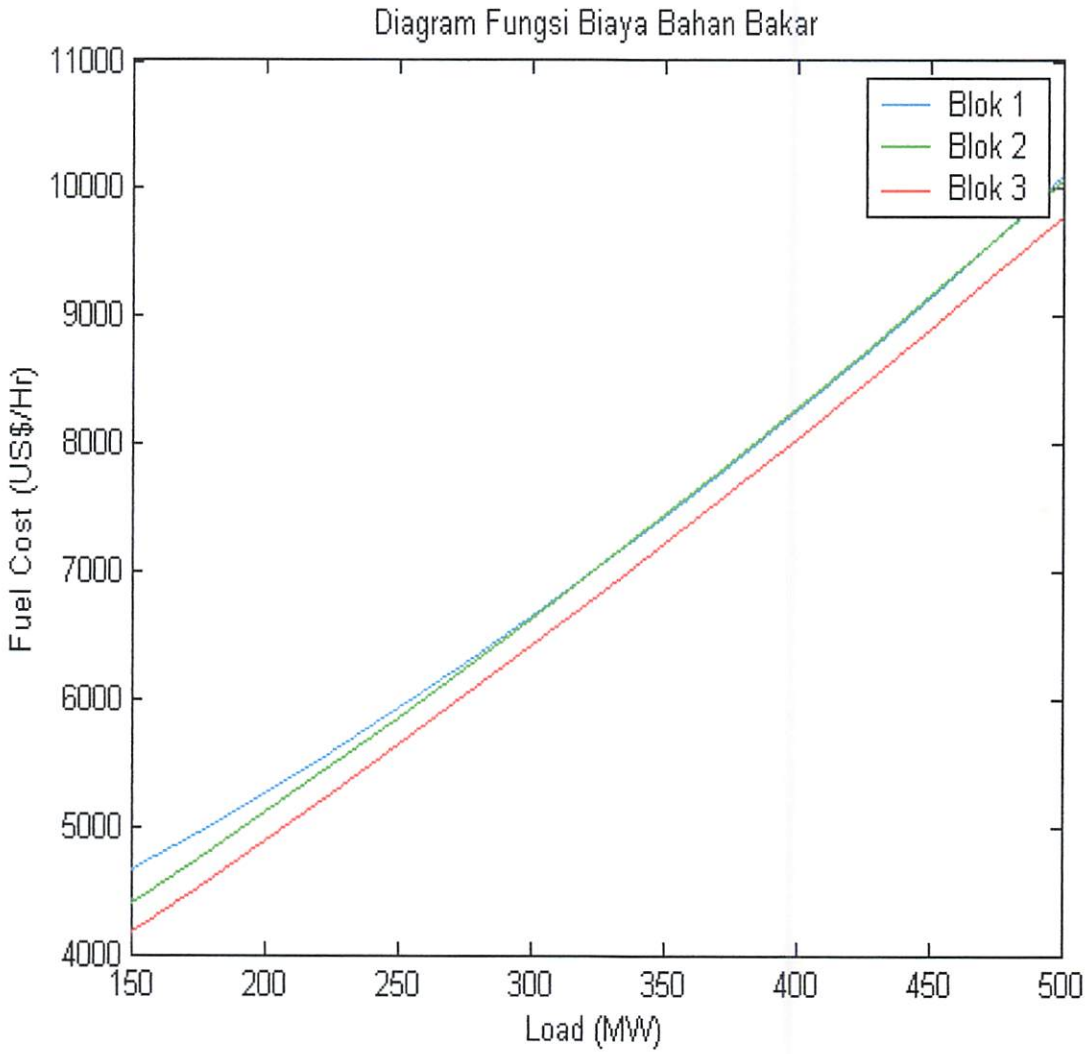
No	Beban Blok (MW)	Laju Tambahan B.Bakar ($ US/Hr)		
		Blok 1	Blok 2	Blok 3
1	150	11,26	14,03	14,17
2	175	12,18	14,34	14,43
3	200	12,74	14,66	14,70
4	225	13,30	14,97	14,97
5	250	13,87	15,28	15,23
6	275	14,43	15,59	15,50
7	300	14,99	15,90	15,77
8	325	15,56	16,21	16,04
9	350	16,12	16,52	16,30
10	375	16,68	16,83	16,57
11	400	17,25	17,14	16,84
12	425	18,37	17,45	17,11
13	450	18,93	17,76	17,37
14	475	18,93	18,03	17,64
15	500	19,50	18,38	17,91

Grafik 4.5 :

Diagram Fungsi Biaya Bahan Bakar

Pada PLTGU Unit Pembangkitan Gresik

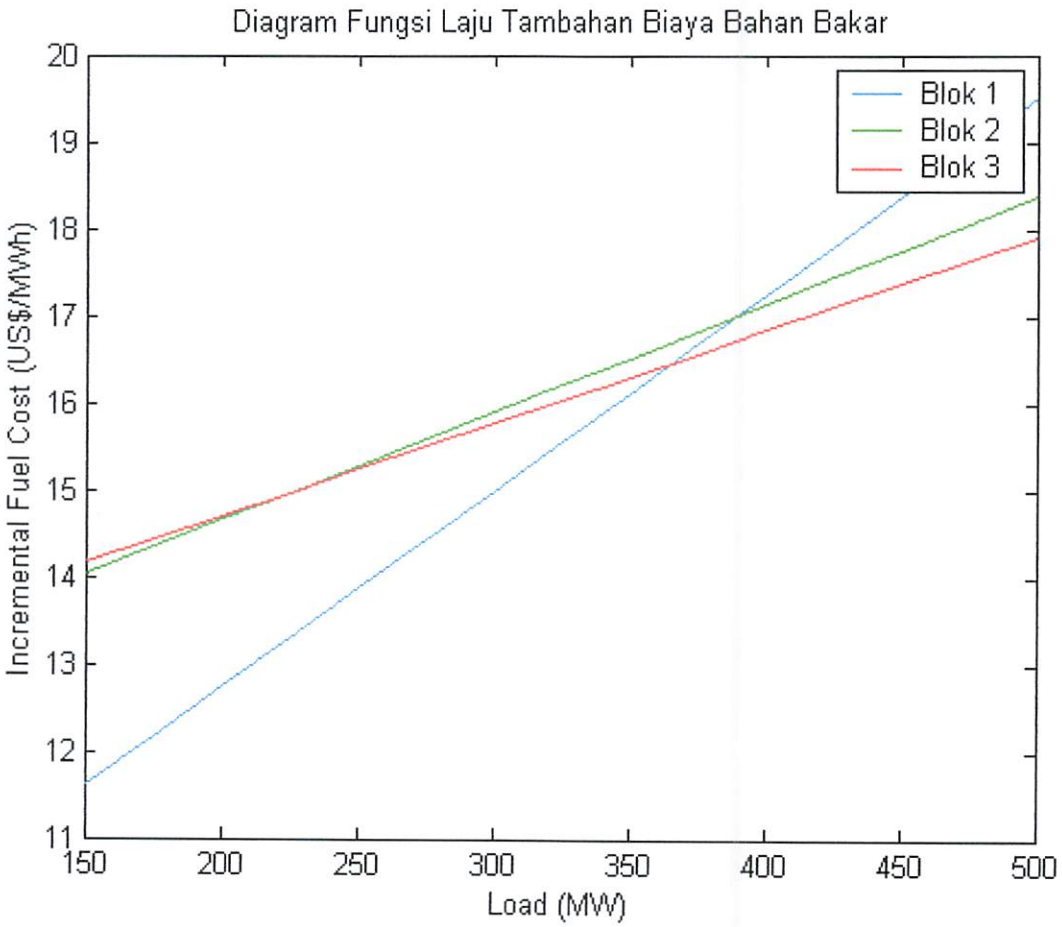
Konfigurasi 3.3.1



Grafik 4.6

Diagram Fungsi Laju Tambahan Biaya Bahan Bakar

PLTGU Unit Pembangkitan Gresik Konfigurasi 3.3.1



Dari grafik dapat diamati bahwa turunan pertama dari fungsi biaya bahan bakar adalah fungsi laju tambahan bahan bakar, untuk blok 1 memiliki kemiringan yang jauh lebih curam dibandingkan dengan blok 2, dan blok 3. Hal ini menunjukkan bahwa untuk kenaikan setiap satu satuan tingkat beban yang sama masing-masing blok, akan menyebabkan biaya untuk memproduksi energi listrik juga akan naik, tetapi kenaikan biaya yang dialami blok1 jauh lebih tinggi daripada blok 2, maupun blok 3.

Tabel 4.6

Tabel Fungsi Biaya Bahan Bakar dan Distribusi Beban Masing-masing Blok Pada PLTGU Gresik

375(MW) - 725(MW)

No	Beban (MW)	Tingkat Beban masing-masing Blok (MW)			Biaya Bahan Bakar ($ US/Hr)			
		Blok 1	Blok 2	Blok 3	Blok 1	Blok 2	Blok 3	Total (Σ)
1	375	125	125	125	4372	4047	3822	12242
2	400	125	125	150	4372	4047	4173	12592
3	425	125	125	175	4,372	4047	4530	12950
4	450	125	125	200	4372	4047	4895	13314
5	475	125	125	225	4372	4047	5265	13689
6	500	125	125	250	4372	4047	5643	14063
7	525	125	125	275	4372	4047	6027	14447
8	550	125	125	300	4372	4047	6418	14838
9	575	125	125	325	4372	4047	6816	15235
10	600	125	125	350	4372	4047	7220	15,640
11	625	125	225	375	4372	5482	7631	17486
12	650	125	225	300	4372	5482	6418	16273
13	675	125	250	300	4372	5861	6418	16651
14	700	125	275	300	4372	6247	6418	17037
15	725	125	300	300	4372	6640	6418	17431

Tabel 4.7

Tabel Fungsi Biaya Bahan Bakar dan Ditribusi Beban Masing-masing Blok Pada PLTGU Gresik

750(MW) – 1125(MW)

No	Beban (MW)	Tingkat Beban masing-masing Blok (MW)			Biaya Bahan Bakar ($ US/Hr)			
		Blok 1	Blok 2	Blok 3	Blok 1	Blok 2	Blok 3	Total (Σ)
16	750	125	300	325	4372	6640	6816	17828
17	775	125	300	350	4372	6640	7220	18233
18	800	125	300	375	4372	6640	7631	18644
19	825	225	300	300	5591	6640	6418	18649
20	850	250	300	300	5930	6640	6418	18989
21	875	275	300	300	6284	6640	6418	19343
22	900	300	300	300	6652	6640	6418	19711
23	925	300	300	325	6652	6640	6816	20108
24	950	300	300	350	6652	6640	7220	20513
25	975	300	300	375	6652	6640	7631	20924
26	1000	300	350	350	6652	7451	7220	21323
27	1025	300	350	375	6652	7451	7631	21734
28	1050	300	375	375	6652	7868	7631	22151
29	1075	350	350	375	7430	7451	7631	22512
30	1100	375	375	350	7840	7868	7220	22928
31	1125	375	375	375	7840	7868	7631	23339

Tabel 4.8

Tabel Fungsi Biaya Bahan Bakar dan Distibusi Beban Masing-masing Blok Pada PLTGU Gresik

1150(MW) – 1500(MW)

No	Beban (MW)	Tingkat Beban masing-masing Blok (MW)			Biaya Bahan Bakar ($ US/Hr)			
		Blok 1	Blok 2	Blok 3	Blok 1	Blok 2	Blok 3	Total (Σ)
32	1150	375	375	400	7840	7868	8049	23757
33	1175	300	375	500	6652	7868	9786	24306
34	1200	350	375	475	7430	7868	9342	24640
35	1225	375	375	475	7840	7868	9342	25050
36	1250	375	375	500	7840	7868	9786	25494
37	1275	375	400	500	7840	8293	9786	25919
38	1300	350	475	475	7430	9613	9342	26385
39	1325	375	475	475	7840	9613	9342	26796
40	1350	375	475	500	7840	9613	9786	27240
41	1375	375	500	500	7840	10069	9786	27696
42	1400	400	500	500	8265	10069	9786	28120
43	1425	475	475	475	9622	9613	9342	28577
44	1450	475	475	500	9622	9613	9786	29021
45	1475	475	500	500	9622	10069	9786	29477
46	1500	500	500	500	10102	10069	9786	29958

4.4 Perhitungan Distribusi Beban Blok.

Sebelum Metode Genetic Algorithm dapat diterapkan untuk menghitung distribusi beban pada masing-masing blok, maka langkah pertama yang harus dilakukan adalah menentukan fungsi karakteristik masukan keluaran (H = f (P) (MBTU/H) dan (F = f (P) ($US/H)) untuk masing-masing blok, dan hal tersebut telah dilakukan pada sub bab (4.3).

Hal lain yang harus diperhatikan adalah nilai batasan yang telah ditentukan. Nilai ini terkait dengan batas maksimum dan minimum kapasitas maksimum dan minimum maisng-maisng blok. Batasan tersebut adalah :

a) Untuk $Pi \geq Pi\,(\max)$, maka $Pi = Pi\,(\max)$

b) Untuk $Pi \leq Pi\,(\min)$, maka $Pi = Pi\,(\min)$

c) Untuk $Pi(\min) \leq Pi \leq Pi(\max)$, maka $Pi = Pi$ (hasil optimasi).

4.4.1. Penggunaan Program Komputer Dengan Metode Genetic Algorithm

Untuk menentukan Optimasi Economic Dispatch pada PLTGU Gresik maka digunakan bantuan komputer. Dengan menggunakan program komputer ini, maka waktu komputasi yang dibutuhkan pada saat menganalisis data akan jauh lebih cepat karena data yang dianalisis banyak dan dibutuhkan ketelitian yang tinggi.

Program komputer ini dengan didukung computer berspecipikasi Pentium IV Intel. 2.8Ghz memori 512 Mb menggunakan bahasa pemrograman

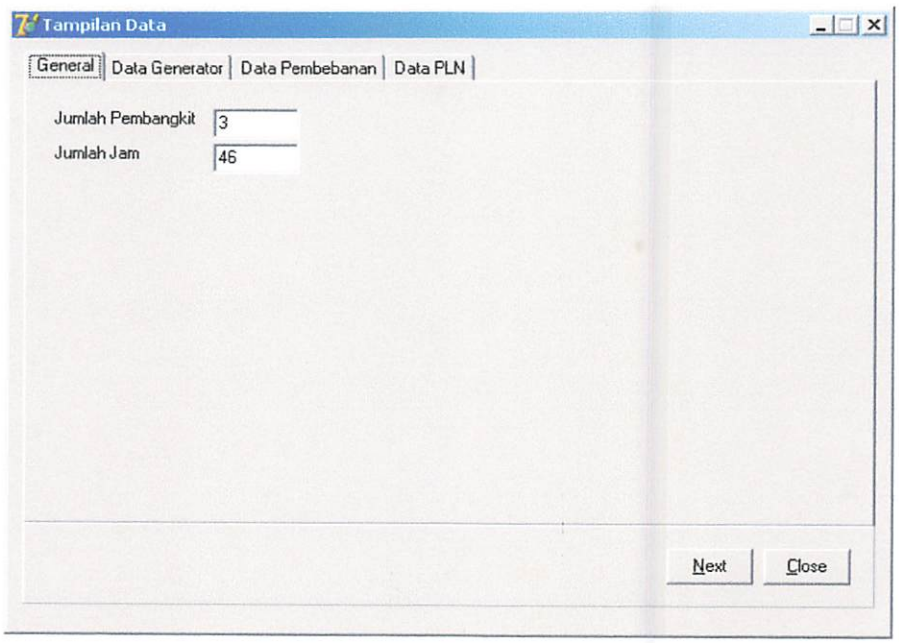
Borland Delphi versi 7.0,merupakan bahasa pemrograman terstruktur yang dapat digunakan untuk berbagai macam aplikasi.

4.4.2. Tampilan program

1. Tampilan utama dari program

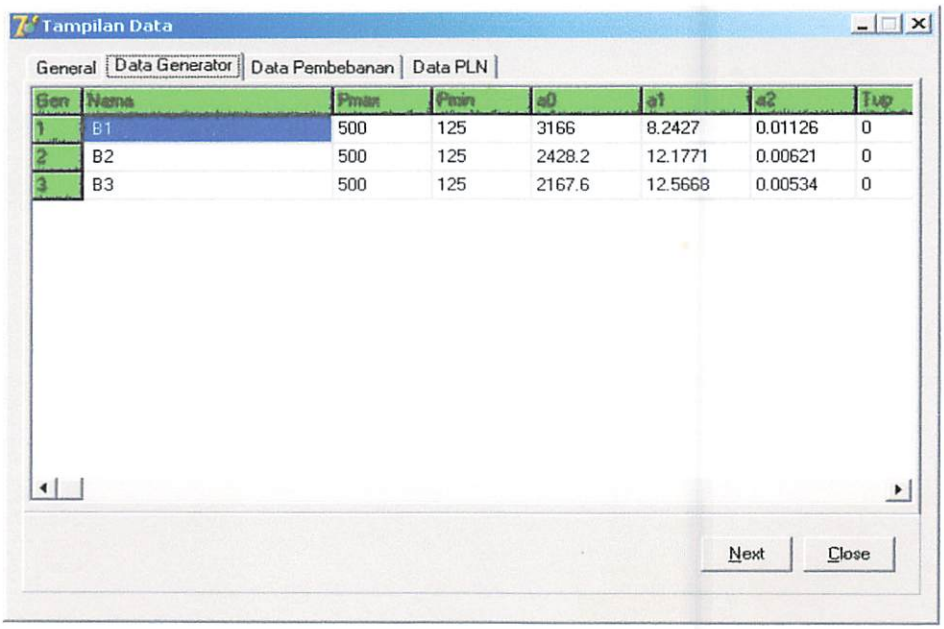
Gambar 4.1. Tampilan Utama Program

2. Kemudian setelah itu tekan tombol buka data untuk membuka file yang tersimpan.



Gambar 4.2. Input Data Genertal

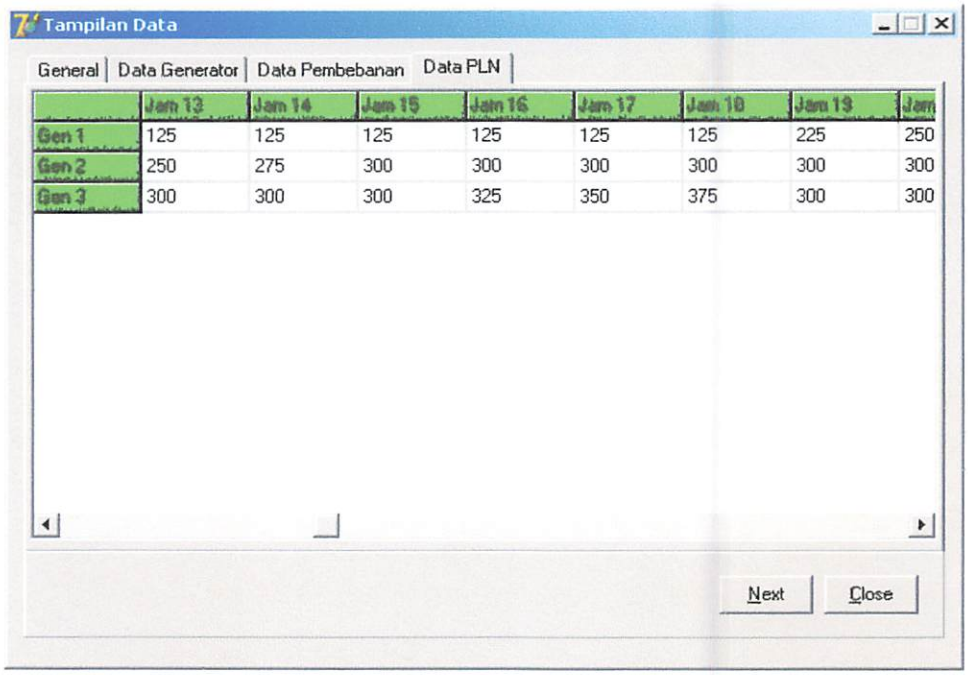
3. Setelah itu masukkan data CCPP Unit Pembangkitan Gresik yang beroperasi pada pola konfigurasi 3.3.1 – 3.3.1 – 3.3.1.



Gen	Nama	Pmax	Pmin	a0	a1	a2	Tup
1	B1	500	125	3166	8.2427	0.01126	0
2	B2	500	125	2428.2	12.1771	0.00621	0
3	B3	500	125	2167.6	12.5668	0.00534	0

Gambar 4.3. Input Data Generator

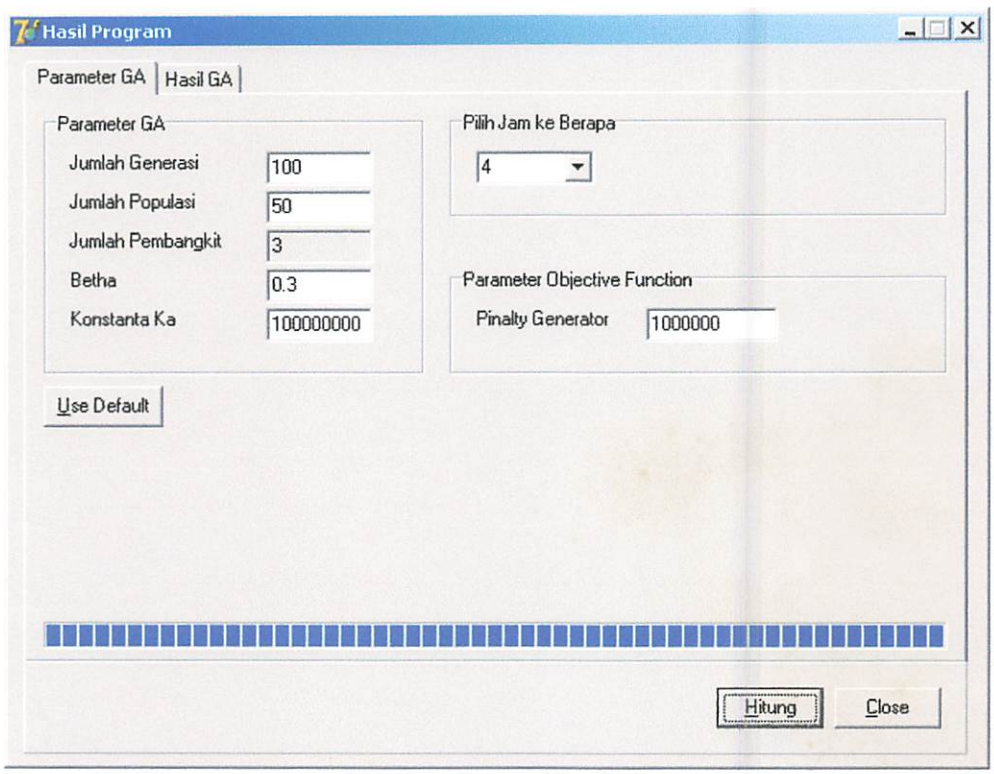
4. Masukkan data tingkatan pembebanan tiap-tiap Blok CCPP Unit Pembangkitan Gresik yang beroperasi pada pola konfigurasi 3.3.1 – 3.3.1 – 3.3.1. mulai kapasitas minumum tiap-tiap blok (125 MW) sampai kapasitas maksimum tiap-tiap blok (500 MW)



	Jam 13	Jam 14	Jam 15	Jam 16	Jam 17	Jam 18	Jam 19	Jam
Gen 1	125	125	125	125	125	125	225	250
Gen 2	250	275	300	300	300	300	300	300
Gen 3	300	300	300	325	350	375	300	300

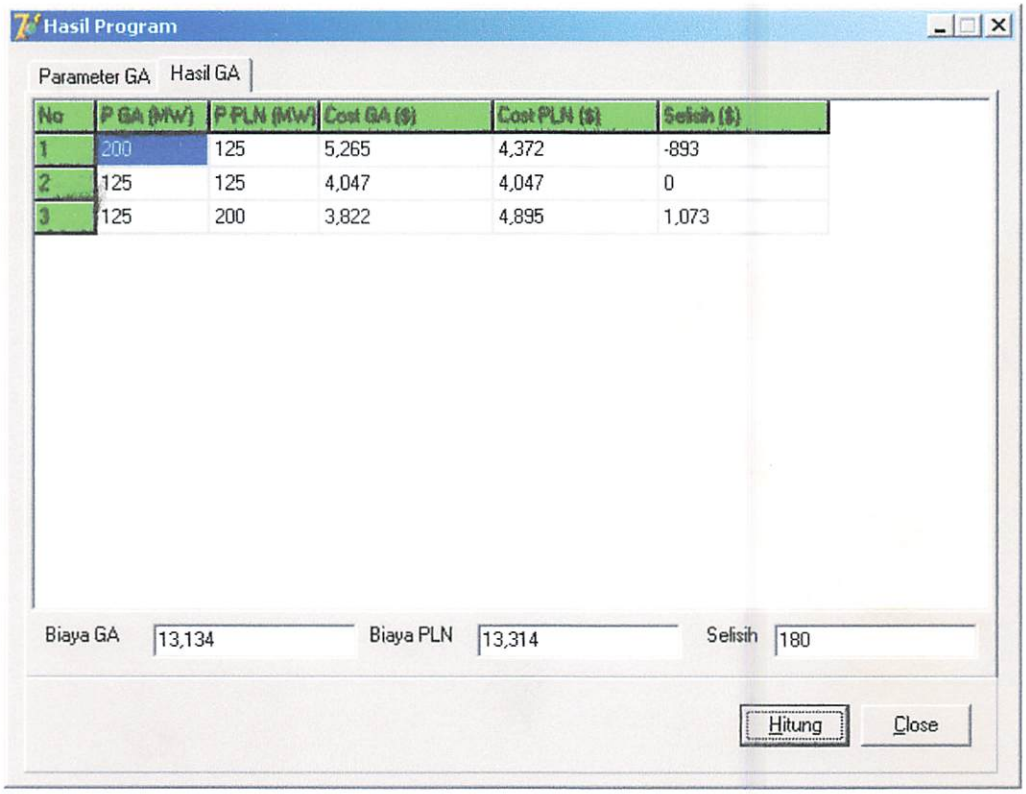
Gambar 4.4. Input Data Tingkatan Pembebanan Pola Gresik

5. Setelah menetapkan parameter tersebut maka komputasi dapat dilakukan, untuk menentukan kebutuhan beban yang disuplai dengan biaya yang minimum perjam.



Gambar 4.5. Parameter *Genetic Algorithm*

6. Tekan tombol Hitung untuk menampilkan hasil komputasi, untuk menentukan hasil optimasi pembebanan ekonomis dengan *Genetic Algorithm,* yaitu *Cost per Jam* untuk melihat hasil perbandingan biaya pola Genetic Algorithm dan pola gresik pada tingkatan beban tertentu serta melihat selisih biayanya dengan kebutuhan beban yang disuplai dengan biaya yang minimum.



No	P GA (MW)	P PLN (MW)	Cost GA ($)	Cost PLN ($)	Selisih ($)
1	200	125	5,265	4,372	-893
2	125	125	4,047	4,047	0
3	125	200	3,822	4,895	1,073

Gambar 4.6.

Hasil Optimasi Pembebanan Ekonomis dengan *Genetic Algorithm*

Tabel 4.12

Tabel Hasil Penghematan Biaya Bahan Bakar Dengan Metode Genetic Algorithm dibanding PLTGU Gresik

Beban 375 (MW) – 725 (MW)

No	Total Beban [MW]	Biaya Bahan Bakar [$US/Hr]		Penghematan Biaya Bahan Bakar [$US/Hr]
		Genetic Algorithm	PLTGU Gresik	
1	375	12242	12242	0
2	400	12525	12592	67
3	425	12823	12950	127
4	450	13134	13314	180
5	475	13460	13685	225
6	500	13799	14063	263
7	525	14146	14447	361
8	550	14496	14838	341
9	575	14849	15235	386
10	600	15205	15640	435
11	625	15564	16051	487
12	650	15925	16273	348
13	675	16290	16651	361
14	700	16657	17037	380
15	725	17027	17431	404

Tabel 4.13

Tabel Hasil Penghematan Biaya Bahan Bakar Dengan Metode Genetic Algorithm dibanding PLTGU Gresik

Beban 750 (MW) – 1125 (MW)

No	Total Beban [MW]	Biaya Bahan Bakar [$US/Hr]		Penghematan Biaya Bahan Bakar [$US/Hr]
		Genetic Algorithm	PLTGU Gresik	
16	750	17400	17828	428
17	775	17776	18233	457
18	800	18155	18644	489
19	825	18536	18649	113
20	850	18921	18987	68
21	875	19308	19343	35
22	900	19698	19711	13
23	925	20091	20108	17
24	950	20487	20513	26
25	975	20886	20924	38
26	1000	21287	21323	36
27	1025	21692	21734	43
28	1050	22099	22151	52
29	1075	22509	22512	3
30	1100	22922	22928	6
31	1125	23398	23339	1

Tabel 4.14.

Tabel Hasil Penghematan Biaya Bahan Bakar Dengan Metode Genetic Algorithm dibanding PLTGU Gresik

Beban 1150 (MW) – 1500 (MW)

No	Total Beban [MW]	Biaya Bahan Bakar [$US/Hr]		Penghematan Biaya Bahan Bakar [$US/Hr]
		Genetic Algorithm	PLTGU Gresik	
32	1150	23757	23757	0
33	1175	24178	24306	128
34	1200	24603	24640	37
35	1225	25030	25050	20
36	1250	25460	25494	34
37	1275	25843	25919	26
38	1300	26329	26385	56
39	1325	26768	26796	28
40	1350	27210	27240	30
41	1375	27654	27696	42
42	1400	28102	28120	18
43	1425	28554	28577	23
44	1450	29011	29021	11
45	1475	29477	29477	0
46	1500	29958	29958	0

Catatan :

Dari table-tabel diatas tampak bahwa pola pembebanan yang selama ini diterapkan oleh Unit Pembangkitan Gresik masih memiliki selisih biaya bahan bakar yang dapat dihemat bila menggunakan dengan Metode Genetic Algorithm

Apabila mengacu pada hasil penghematan biaya bahan bakar pada table (4.6), (4.7) dan (4.8) jumlahnya sangat bervariasi, seperti contoh jika kita melihat bahwa biaya bahan bakar yang dapat dihemat dapat mencapai 457 [$US/Hr] pada tingkat beban 775 [MW] dan 489 [$US/Hr] pada tingkat beban 800 [MW].

BAB V

KESIMPULAN

5.1 Kesimpulan

1. Pola Pembebanan dengan metode Genetic Algorithm dalam skripsi ini menggunakan asumsi karakteristik masukan keluaran masing-masing Blok tidak sama, hal ini berbeda dengan asumsi yang digunakan PLTGU Gresik, dan dapat diambil kesimpulan dengan metode Genetic Algorithm lebih ekonomis.

2. Dengan membandingkan hasil yang diperoleh maka pola pembebanan yang diterapkan di PLTGU Gresik dengan Metode Genetic Algorithm berbeda. Sebagai contoh pada permintaan beban 1425 (MW), biaya bahan bakar apabila menggunakan pola pembebanan Gresik adalah 28577 [$US/Hr]. Metode Genetic Algorithm memerlukan biaya bahan bakar 28554 [$US/Hr], sehingga biaya yang bisa dihemat 23 [$US/Hr].

3. Dalam penggunaan Combined Cycle Power Plant ternyata hanya digunakan pada Pembangkit termal Adalah Pembangkit Listrik Tenaga Gas dan Uap

5.2 Saran

Dalam skripsi ini hanya dilakukan pola pembagian beban pada kombinasi 3.3.1, yaitu pola konfigurasi pada kapasitas maksimum. Hal ini dilakukan akibat data test yang tersedia terbatas pada kombinasi 3.3.1 saja. Perlu dipertimbangkan untuk membandingkan pemakaian bahan bakar pada tingkat beban yang sama dengan

kombinasi yang berbeda. Untuk itu perlu dilakukan pengetesan terhadap pola kombinasi 2.2.1 dan 1.1.1,agar diketahui karakteristik masukan keluaran masing-masing Blok dengan kombinasi tersebut.

LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

BERITA ACARA UJIAN SKRIPSI

Nama : ARDIAN HARIYANSYAH

Nim : 99.12.007

Jurusan : Teknik Elektro

Konsentrasi : Energi Listrik

Judul :

OPTIMASI ECONOMIC DISPATCH COMBINED CYCLE POWER PLANT DENGAN PENDEKATAN METODE GENETIC ALGORITHM

Dipertahankan dihadapan majelis penguji skripsi jenjang strata satu (S-1) pada:

Hari : Sabtu

Tanggal : 18 Maret 2006

Dengan Nilai : B+ (75,8)

Panitia Ujian Skripsi

(Ir. Mocthar Asroni , MSME)
Ketua

(Ir. F. Yudi Limpraptono, MT)
Sekretaris

anggota Penguji

(Ir.Teguh Herbasuki, MT)

(Ir. M. Abdul Hamid,MT)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi jurusan teknik elektro jenjang strata satu (S-1) Telah dilakukan perbaikan skripsi oleh:

Nama : Ardian Hariyansyah

Nim : 99.12.007

Jurusan : Teknik Elektro

Konsentrasi : Energi Listrik

Judul : **OPTIMASI ECONOMIC DISPATCH COMBINED CYCLE POWER PLANT DENGAN PENDEKATAN METODE GENETIC ALGORITHM**

Perbaikan meliputi :

No.	Materi Perbaikan	Paraf
1	Tampilkan Hasil Beban Tiap – Tiap Blok	

Anggota Penguji

(Ir. Teguh Herbasuki,MT) **(Ir. M. Abdul Hamid, MT)**

Dosen Pembimbing

(Ir. Widodo Pudji Muljanto, MT)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK

LEMBAR BIMBINGAN SKRIPSI

1	Nama	:	ARDIAN HARIYANSYAH
2	N.I.M	:	99.12.007
3	Jurusan	:	TEKNIK ELEKTRO
4	Konsentrasi	:	ENERGI LISTRIK
5	Program Pendidikan	:	STRATA SATU (S-1)
6	Judul Skripsi	:	OPTIMASI ECONOMIC DISPATCH COMBINED CYCLE POWER PLANT DENGAN PENDEKATAN METODE GENETIC ALGORITHM
7	Tanggal Mengajuan Skripsi	:	23 APRIL 2005
8	Tanggal Menyelesaian Skripsi	:	31 APRIL 2006
9	Dosen Pembimbing	:	Ir. Widodo Pudji Muljanto,MT
10	Telah Dievaluasi dengan Nilai	:	80 (A)

Mengetahui
Ketua Jurusan T.Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP. Y . 1039500274

Disetujui
Dosen Pembimbing

Ir. Widodo Pudji Muljanto, MT
NIP. 101 880 0160

LEMBAR BIMBINGAN SKRIPSI

1	Nama	:	ARDIAN HARIYANSYAH
2	NIM	:	99.12.007
3	Jurusan	:	TEKNIK ELEKTRO
4	Konsentrasi	:	ENERGI LISTRIK
5	Program Pendidikan	:	STRATA SATU (S-1)
6	Judul Skripsi	:	OPTIMASI EKONOMIS DISPATCH COMBINED CYCLE POWER PLANT DENGAN PENDEKATAN METODE GENETIC ALGORITHM
7	Tanggal Mengajukan Skripsi	:	22 APRIL 2004
8	Tanggal Mengadakan Skripsi	:	21 APRIL 2004
9	Dosen Pembimbing	:	Ir. Widodo Pudji Muljanto,MT
10	Telah Dikoreksi dengan Nilai	:	80 (A)

Mengetahui
Ketua Jurusan T.Elektro S-1

Ir. C. Judi Limpraptono, MT
NIP. Y. 1039500274

Disetujui
Dosen Pembimbing

Ir. Widodo Pudji Muljanto, MT
NIP. 131 880 060

Validasi Program pada IEEE Test Sistem

Dengan simulasi program penyelesaian Economic Dispatch dengan metode Genetic Algorithm dalam skripsi ini dengan judul yang diambil dari IEEE yaitu : data pada jurnal "*Genetic Algorithm Solution of Economic Dispatch with Combine Cycle Power Plants*", untuk jumlah pembangkit sebanyak 3 unit dan periode waktu 1 jam, data-data selengkapnya seperti pada tabel 1 dan 2.

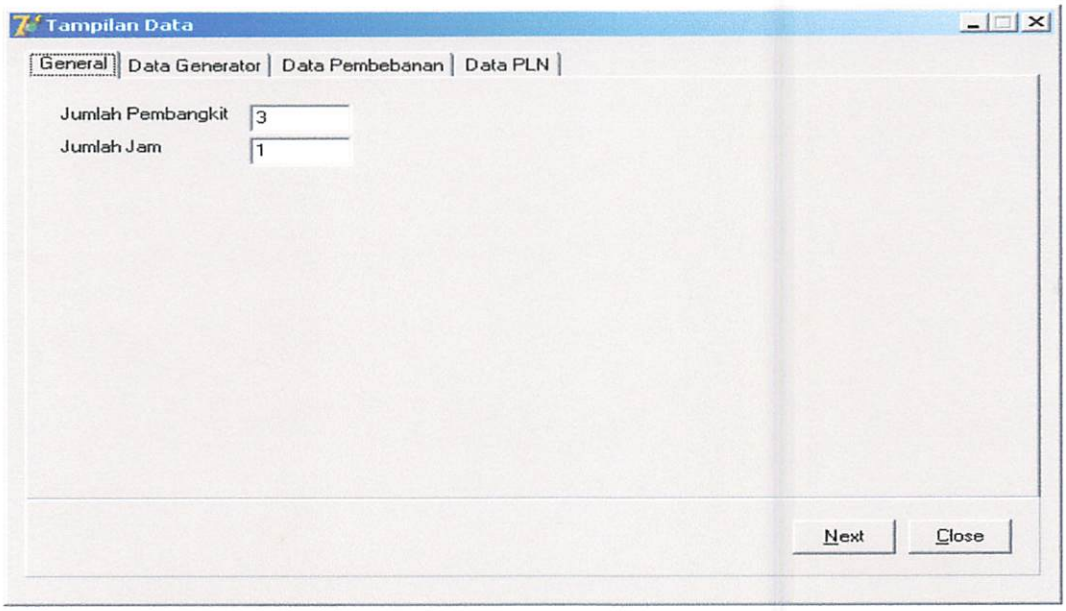
Tabel 1. Data Unit Pembangkit (Data IEEE)

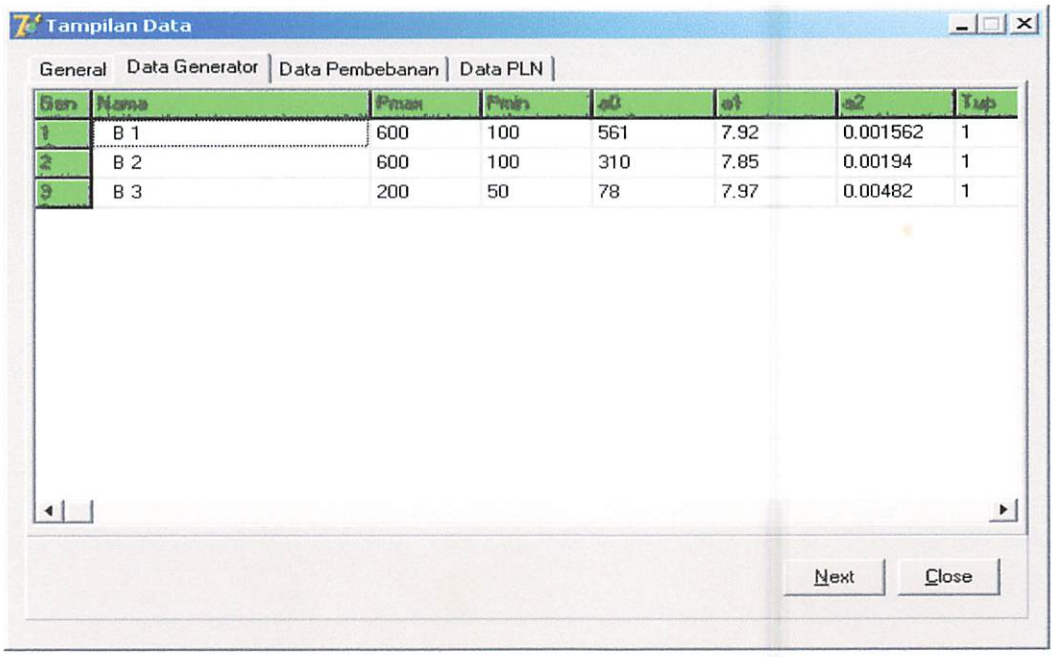
No	Nama Pembangkit	Kapasitas (MW)		Koefisien Biaya Bahan Bakar		
		P maks	P min	a	b	c
1	Unit 1	600	100	0,001562	7,92	561
2	Unit 2	600	100	0,001940	7,85	310
3	Unit 3	200	50	0,004820	7,97	78

Tabel 2. Data Validasi (Data IEEE)

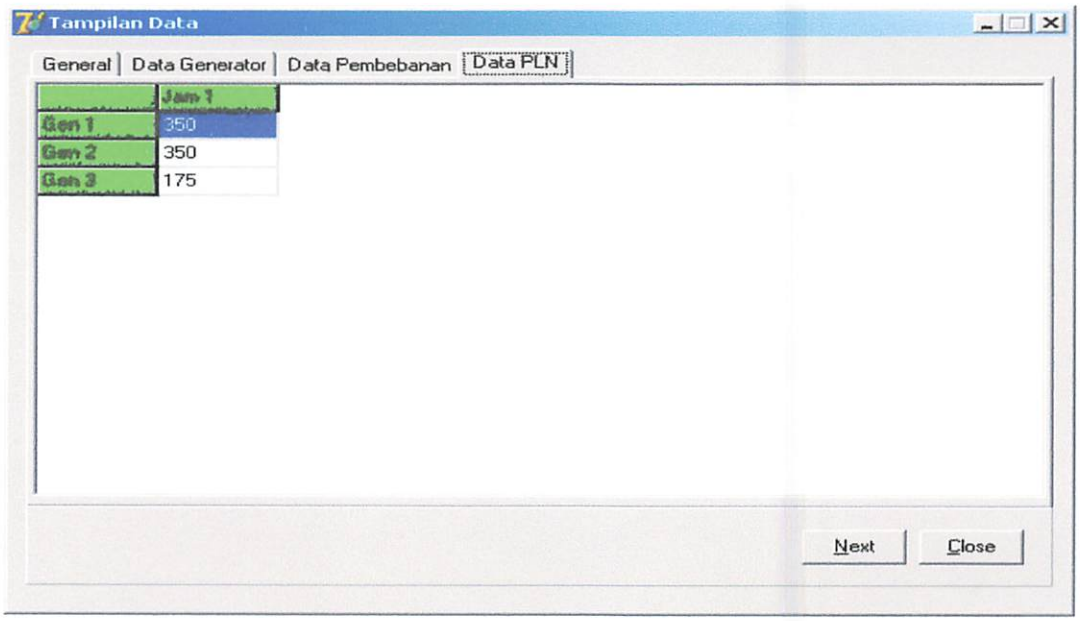
No	Nama Pembangkit	Beban (MW)
1	Unit 1	350
2	Unit 2	375
3	Unit 3	200
	Total	925

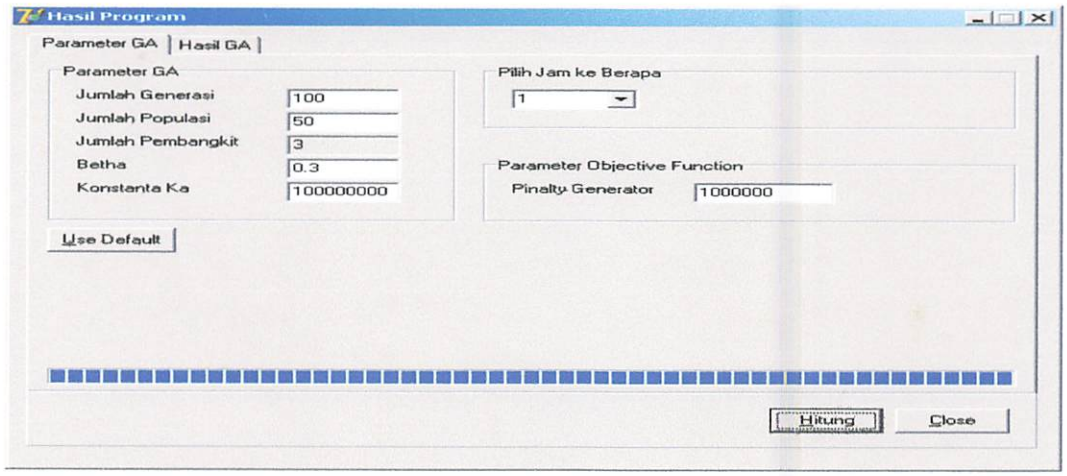
Tampilan Software Metode EP data Jurnal dengan beban system 850 MW

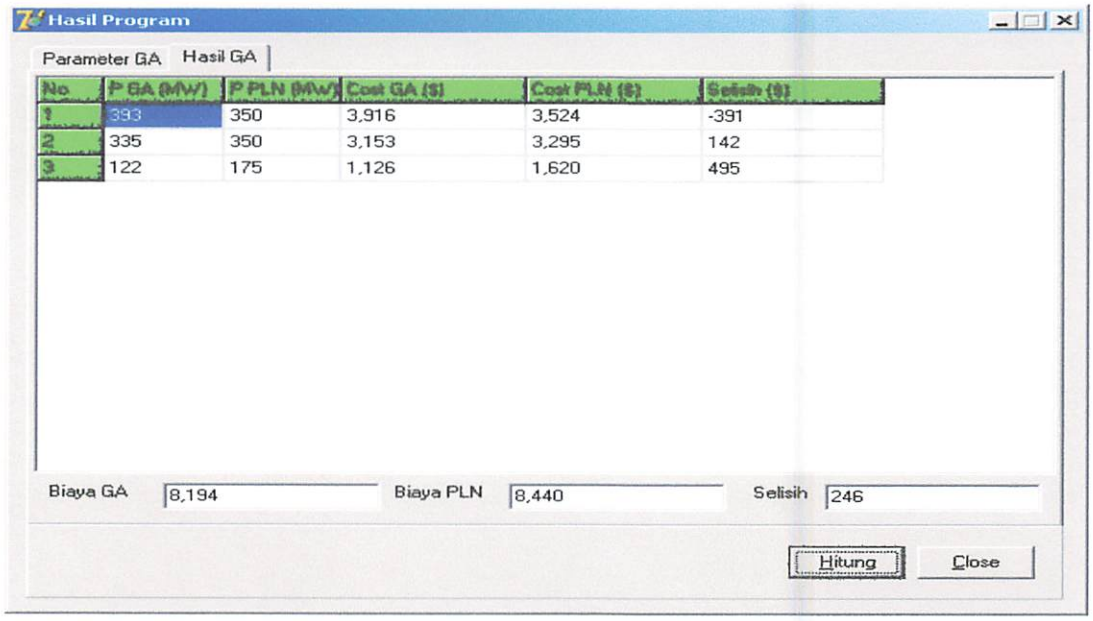




Gen	Nama	Pmax	Pmin	a0	a1	a2	Tup
1	B 1	600	100	561	7.92	0.001562	1
2	B 2	600	100	310	7.85	0.00194	1
3	B 3	200	50	78	7.97	0.00482	1


Tampilan Data
General
Data Generator
Data Pembebanan
Data PLN
Jam 1
Gen 1
350
Gen 2
350
Gen 3
175
Next
Close


Hasil Program
Parameter GA
Hasil GA
Parameter GA
Jumlah Generasi
100
Jumlah Populasi
50
Jumlah Pembangkit
3
Betha
0.3
Konstanta Ka
100000000
Pilih Jam ke Berapa
1
Parameter Objective Function
Pinalty Generator
1000000
Use Default
Hitung
Close



No.	P GA (MW)	P PLN (MW)	Cost GA ($)	Cost PLN ($)	Selisih ($)
1	393	350	3,916	3,524	-391
2	335	350	3,153	3,295	142
3	122	175	1,126	1,620	495

Dengan memakai data IEEE diatas maka diperoleh hasi sebagai berikut dengan parameter generasi = 100, populasi = 50, jumlah pembangkit = 3, komutasi dilakukan jam 1 dengan beban 850 MW.

Setelah dilakukan proses perbandingan data yang ada pada jurnal dan data pada hasil program, maka didapatkan biaya operasional pada data jurnal sebesar $. 8165.37.- dan biaya operasional pada hasil program sebesar $. 8194.-. Maka selisih antara biaya operasional pada jurnal dan biaya operasional hasil program adalah $. 28.63.-(error 0,22).

PJB

DATA PEMBEBANAN DISTRIBUSI BEBAN BLOK
KONFIGURASI 3.3.1PT. PLN PJB II UP. GRESIK

NO	Beban Total	Beban Blok 1	Beban Blok 2	Beban Blok 3	BBG Blok 1	BBG Blok 2	BBG Blok 3	Total Aliran BBG
	MW	MW	MW	MW	KNM^3 / H	KNM^3 / H	KNM^3 / H	KNM^3 / H
1	375	125	125	125	39.21	39.21	39.21	117.63
2	400	125	125	150	39.21	39.21	44.78	123.20
3	425	125	125	175	39.21	39.21	47.98	126.40
4	450	125	125	200	39.21	39.21	52.31	130.73
5	475	125	125	225	39.21	39.21	55.62	134.04
6	500	125	125	250	39.21	39.21	58.65	137.07
7	525	125	125	275	39.21	39.21	62.98	141.40
8	550	125	125	300	39.21	39.21	65.54	143.96
9	575	125	125	325	39.21	39.21	70.12	148.54
10	600	125	125	350	39.21	39.21	74.03	152.45
11	625	125	125	375	39.21	39.21	77.96	156.38
12	650	125	225	300	39.21	55.62	65.54	110.37
13	675	125	250	300	39.21	58.65	65.54	163.40
14	700	125	275	300	39.21	62.98	65.54	167.73
15	725	125	300	300	39.21	65.54	65.54	170.29
16	750	125	300	325	39.21	65.54	70.12	174.87
17	775	125	300	350	39.21	65.54	74.03	178.78
18	800	125	300	375	39.21	65.54	77.96	182.71
19	825	225	300	300	55.62	65.54	65.54	186.70
20	850	250	300	300	58.65	65.54	65.54	189.73
21	875	275	300	300	62.98	65.54	65.54	194.06
22	900	300	300	300	65.54	65.54	65.54	196.62
23	925	300	300	325	65.54	65.54	70.12	201.20

Lampiran 1

DATA PEMBEBANAN DISTRIBUSI BEBAN BLOK
KONFIGURASI 3.3.1-3.3.1-3.3.1
PT. PLN PJB II UP. GRESIK

NO	Beban Total	Beban Blok 1	Beban Blok 2	Beban Blok 3	BBG Blok 1	BBG Blok 2	BBg Blok 3	Total Aliran BBG
	MW	MW	MW	MW	KNM^3 / H	KNM^3 / H	KNM^3 / H	KNM^3 / H
24	950	300	350	350	65.54	65.54	74.03	205.11
25	975	300	375	375	65.54	65.54	77.96	209.04
26	1000	300	350	350	65.54	74.03	74.03	213.60
27	1025	300	350	375	65.54	74.03	77.96	217.53
28	1050	300	375	375	65.54	77.96	77.96	221.46
29	1075	350	350	375	74.03	74.03	77.96	226.02
30	1100	375	375	350	77.96	77.96	74.03	229.95
31	1125	375	375	375	77.96	77.96	77.96	233.88
32	1150	375	375	400	77.96	77.96	83.04	238.96
33	1175	300	375	500	65.54	77.96	83.04	243.56
34	1200	350	375	475	74.03	77.96	95.81	247.80
35	1225	375	375	475	77.96	77.96	95.81	251.73
36	1250	375	375	500	77.96	77.96	100.06	255.98
37	1275	375	400	500	77.96	83.04	100.06	261.06
38	1300	350	475	475	74.03	95.81	95.81	265.65
39	1325	375	475	475	77.96	95.81	95.81	269.58
40	1350	375	475	500	77.96	100.06	100.06	273.83
41	1375	375	500	500	77.96	100.06	100.06	278.08
42	1400	400	500	500	83.04	100.06	100.06	283.16
43	1425	475	475	475	95.81	95.81	95.81	287.43
44	1450	475	475	500	95.81	95.81	100.06	291.68
45	1475	475	500	500	95.81	100.06	100.06	295.93
46	1500	500	500	500	100.06	100.06	100.06	300.18

Lampiran 2

DATA TOTAL ALIRAN BAHAN BAKAR GAS PER TINGKAT BEBAN PADA TIAP BLOK

PJB

No	Tingkat Beban	Total Aliran BBG	
		KNM^3 /Hr	MBTU/Hr
1	150	46,55	1838,33
2	175	49,37	1939,69
3	200	53,11	2097,39
4	225	56,83	2244,30
5	250	59,67	2356,45
6	275	63,31	2460,71
7	300	65,25	2576,82
8	325	69,70	2752,55
9	350	73,77	2913,28
10	375	78,82	3112,72
11	400	83,59	3301,09
12	425	88,06	3477,62
13	450	92,69	3660,46
14	475	96,05	3793,15
15	500	99,98	3948,35

Blok 1

No	Tingkat	Total Aliran BBG	
		KNM^3 /Hr	MBTU/Hr
1	150	43,56	1720,44
2	175	47,42	1872,92
3	200	51,28	2025,44
4	225	55,15	2177,91
5	250	59,01	2330,43
6	275	62,87	2482,91
7	300	66,73	2635,42
8	325	70,59	2787,94
9	350	73,92	2919,21
10	375	78,23	3089,53
11	400	82,73	3267,44
12	425	87,28	3447,13
13	450	91,83	3626,82
14	475	96,38	3806,46
15	500	100,93	3986,15

Blok 2

No	Tingkat	Total Aliran BBG	
		KNM^3 /Hr	MBTU/Hr
1	150	41,44	1636,84
2	175	45,27	1787,78
3	200	49,09	1938,67
4	225	52,91	2089,61
5	250	56,73	2240,55
6	275	60,55	2391,48
7	300	64,37	2542,38
8	325	68,20	2693,32
9	350	72,31	2855,98
10	375	75,88	2996,73
11	400	80,22	3168,08
12	425	84,70	3344,96
13	450	89,18	3521,85
14	475	93,66	3698,77
15	500	98,13	3875,65

Blok 3

*1 KNM^3 = 37,3258 MSCF
*HHV Bahan Bakar Gas = 1058,02 BTU/SCF

Lampiran 3

Lampiran

DATA PEMBANGKIT UP. GRESIK

PJB

NO	BLOK	Fuel	Kapasitas	Pabrikan	Tanggal operasi
I	**Gas Turbin Power Plant**				
	GT 1	HSD/NG	20 MW	Alsthom-France	07-6-1988
	GT 2	HSD/NG	20 MW	Alsthom-France	09-6-1988
	GT 3	HSD	21 MW	Alsthom-France	20-8-1994
	GT 4	HSD/NG	20 MW	GE- USA	02-9-1994
	GT 5	HSD/NG	21 MW	GE- USA	24-2-1995
II	**Steam Turbin Power Plant**				
	ST 1	HSD/RO	100 MW	Toshiba-Japan	31-08-1991
	ST 2	HSD/RO	100 MW	Toshiba-Japan	14-11-1991
	ST 3	HSD/NG	200 MW	Toshiba-Japan	15-03-1998
	ST 4	HSD/NG	200 MW	Toshiba-Japan	01-07-1998
III	**Combined Cycle Power Plant**				
	Blok 1				
	GT 1.1	NG	112,45 MW	Mitsubishi	30-03-2002
		HSD	100,98 MW		24-04-2004
	GT 1.2	NG	112,45 MW	Mitsubishi	01-05-2002
		HSD	100,98 MW		02-05-2004
	GT 1.3	NG	112,45 MW	Mitsubishi	02-06-2002
		HSD	100,98 MW		01-05-2004
	GT 1.0	-	118,91 MW	Mitsubishi	10-04-2003
	Blok 2				
	GT 2.1	NG	112,45 MW	Mitsubishi	20-07-2002
		HSD	100,98 MW		01-05-2004
	GT 2.2	NG	112,45 MW	Mitsubishi	14-08-2002
		HSD	100,98 MW		04-02-2004
	GT 2.3	NG	112,45 MW	Mitsubishi	18-09-2002
		HSD	100,98 MW		03-05-2004
	GT 2.0	-	118,91 MW	Mitsubishi	05-08-2003
	Blok 3				
	GT 3.1	NG	112,45 MW	Mitsubishi	14-01-2003
	GT 3.2	NG	112,45 MW	Mitsubishi	19-01-2003
	GT 3.3	NG	112,45 MW	Mitsubishi	13-01-2003
	GT 3.0	-	118,91 MW	Mitsubishi	30-11-2003
	TOTAL KAPASITAS		**2280 MW**		

Listing Program
DELPHI 7.0

```
unit InputGen;

interface

uses
  Windows, Messages,
SysUtils, Variants.
Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls,
ComCtrls, ExtCtrls,
Grids;

type
  TfrmInput =
class(TForm)
    PageControl1:
TPageControl;
    Panel1: TPanel;
    TabSheet1:
TTabSheet;
    TabSheet2:
TTabSheet;
    TabSheet3:
TTabSheet;
    btnClose: TButton;
    btnNext: TButton;
    SaveDialog1:
TSaveDialog;
    Label1: TLabel;
    Label2: TLabel;
    edtNGen: TEdit;
    edtNjam: TEdit;
    fgGen: TStringGrid;
    fgLoad:
TStringGrid;
    TabSheet4:
TTabSheet;
    fgPLN: TStringGrid;
    procedure
btnCloseClick(Sender:
TObject);
    procedure
FormCreate(Sender:
TObject);
    procedure
edtNGenChange(Sende
r: TObject);
    procedure
edtNjamChange(Sende
r: TObject);
    procedure
btnNextClick(Sender:
TObject);
  private
    { Private
declarations }
  public
    { Public declarations
}
  end;

var
  frmInput: TfrmInput;

implementation

uses uObjFunc, uHasil;

{$R *.dfm}

procedure
TfrmInput.btnCloseCli
ck(Sender: TObject);
begin
  Close;
end;

procedure
TfrmInput.FormCreate(
Sender: TObject);
begin
  fgGen.Cells[0,0]
:='Gen';
  fgGen.Cells[1,0]
:='Nama';
  fgGen.Cells[2,0]
:='Pmax';
  fgGen.Cells[3,0]
:='Pmin';
  fgGen.Cells[4,0]
:='a0';
  fgGen.Cells[5,0]
:='a1';
  fgGen.Cells[6,0]
:='a2';
  fgGen.Cells[7,0]
:='Tup';
  fgGen.Cells[8,0]
:='Tdown';
  fgGen.Cells[9,0]
:='Sh';

fgGen.Cells[10,0]:='Sc'
;

fgGen.Cells[11,0]:='Tc
old';

fgGen.Cells[12,0]:='Ini
tSt';

fgGen.Cells[13,0]:='Ra
mp Rate';
  fgLoad.Cells[0,0]:='';

fgLoad.Cells[1,0]:='Lo
ad';

fgLoad.Cells[2,0]:='Re
s';
end;

procedure
TfrmInput.edtNGenCh
ange(Sender: TObject);
var i:integer;
begin
  if edtNgen.Text=''
then
  begin

fgGen.RowCount:=2;

fgPLN.RowCount:=2;
  end
  else
  begin

fgGen.RowCount:=Str
ToInt(edtNgen.Text)+1
;

fgPLN.RowCount:=Str
```

```
ToInt(edtNgen.Text)+1
;
   for i:=1 to
StrToInt(edtNgen.Text)
do
   begin

fgGen.Cells[0,i]:=IntT
oStr(i);

fgPLN.Cells[0,i]:='Gen
'+IntToStr(i);
   end;
  end;
end;

procedure
TfrmInput.edtNjamCha
nge(Sender: TObject);
var i:integer;
begin
  if edtNjam.Text=''
then
  begin

fgLoad.RowCount:=2;
   fgPLN.ColCount:=2;
  end
  else
  begin

fgLoad.RowCount:=Str
ToInt(edtNjam.Text)+1
;

fgPLN.ColCount:=StrT
oInt(edtNjam.Text)+1;
   for i:=1 to
StrToInt(edtNjam.Text
) do
   begin

fgLoad.Cells[0,i]:=IntT
oStr(i);

fgPLN.Cells[i,0]:='Jam
'+IntToStr(i);
   end;
  end;
end;

procedure
TfrmInput.btnNextClic
k(Sender: TObject);
var input:TextFile;

NamaFile,Nama:string;

Pmin,Pmax,a2,a1,a0,Sh
,Sc,Ramp,Load,Res:do
uble;

i,j,Tup,Tdown,Tcold,In
itSt,Ngen,Njam:integer
;
begin
  if
btnNext.Caption='&Sa
ve' then
  begin
   if
SaveDialog1.Execute
then
   begin

NamaFile:=SaveDialog
1.FileName;

AssignFile(input,Nama
File+'.txt');
    Rewrite(input);

Ngen:=StrToInt(edtNg
en.Text);

Njam:=StrToInt(edtNja
m.Text);

Writeln(input,Ngen);

Writeln(input,Njam);
    for i:=1 to Ngen do
    begin

Nama:=fgGen.Cells[1,i
];

Pmax:=StrToFloat(fgG
en.Cells[2,i]);

Pmin:=StrToFloat(fgG
en.Cells[3,i]);

a0:=StrToFloat(fgGen.
Cells[4,i]);

a1:=StrToFloat(fgGen.
Cells[5,i]);

a2:=StrToFloat(fgGen.
Cells[6,i]);

Tup:=StrToInt(fgGen.
Cells[7,i]);

Tdown:=StrToInt(fgGe
n.Cells[8,i]);

Sh:=StrToFloat(fgGen.
Cells[9,i]);

Sc:=StrToFloat(fgGen.
Cells[10,i]);

Tcold:=StrToInt(fgGen
.Cells[11,i]);

InitSt:=StrToInt(fgGen.
Cells[12,i]);

Ramp:=StrToFloat(fgG
en.Cells[13,i]);

Writeln(input,Pmax:7:0
,' ',Pmin:7:0,' ',
     a0:9:4,' ',a1:9:4,'
',a2:9:5,' ',Tup,'
',Tdown,' ',
     Sh:7:0,' ',Sc:7:0,'
',tcold,' ',InitSt,'
',Ramp:7:0,' ',Nama);
    end;
    for i:=1 to Njam do
    begin

Load:=StrToFloat(fgLo
ad.Cells[1,i]);
```

```
Res:=StrToFloat(fgLoa
d.Cells[2,i]);

Writeln(input,Load,Res
);
    end;

fgPLN.RowCount:=Ng
en+1;

fgPLN.ColCount:=Nja
m+1;
    for i:=1 to Ngen do
    begin
      for j:=1 to Njam
do
      begin

Load:=StrToFloat(fgP
LN.Cells[j,i]);

Write(input,Load:7:2,'
');
      end;
      Writeln(input,'');
    end;
    CloseFile(input);
  end;
 end
 else if
btnNext.Caption='&Ne
xt' then
 begin

frmHasil.edtNparam.T
ext:=IntToStr(gObjFun
c.Ngen);

frmHasil.cmbJam.Item
s.Clear;
  for i:=1 to
gObjFunc.Njam do
  begin

frmHasil.cmbJam.AddI
tem(IntToStr(i),nil);
  end;

frmHasil.cmbJam.Text:
='1';
  frmHasil.Show;
 end;
end;

end.
```

```
unit uGenerator;

interface

type
 TPembangkit=class
 private
  FNama:string;

FPmax,FPmin,Fa2,Fa1,
Fa0,FSh,FSc,FDaya,F
Ramp:double;

FTup,FTdown,FTcold,
FInitSt:integer;
  function
GetAFLC:double;
 public
  constructor
Create;overload;
  constructor
Create(const
rNama:string;
          const
rPmin,rPmax,ra2,ra1,ra
0,rSh,rSc,rRamp:doubl
e;
          const
rTup,rTdown,rTcold,rI
nitSt:integer);overload;
  constructor
Create(const
rPembangkit:TPemban
gkit);overload;
  procedure
Assign(const
rPembangkit:TPemban
gkit);
  function
GetBiaya(const
rDaya:double):double;
  function
GetDaya(const
rLamda:double):double
;
  function
GetLamda(const
rDaya:double):double;
  property
Nama:string read
FNama write FNama;
  property
Pmax:double read
FPmax write FPmax;
  property
Pmin:double read
FPmin write FPmin;
  property a2:double
read Fa2 write Fa2;
  property a1:double
read Fa1 write Fa1;
  property a0:double
read Fa0 write Fa0;
  property Sh:double
read FSh write FSh;
  property Sc:double
read FSc write FSc;
  property
Ramp:double read
FRamp write FRamp;
  property Tup:integer
read FTup write FTup;
  property
Tdown:integer read
FTdown write
FTdown;
  property
Tcold:integer read
FTcold write FTcold;
```

```
  property
InitSt:integer read
FInitSt write FInitSt;
  property
Daya:double read
FDaya write FDaya;
  property
AFLC:double read
GetAFLC;
 end;

 TGenArr=array of
TPembangkit;

implementation

//constructor
constructor
TPembangkit.Create;
begin
 inherited Create;
end;

constructor
TPembangkit.Create(co
nst rNama:string;
        const
rPmin,rPmax,ra2,ra1,ra
0,rSh,rSc,rRamp:doubl
e;
        const
rTup,rTdown,rTcold,rI
nitSt:integer);
begin
 inherited Create;
 FNama:=rNama;
 FPmin:=rPmin;
 FPmax:=rPmax;
 Fa2:=ra2;
 Fa1:=ra1;
 Fa0:=ra0;
 FSh:=rSh;
 FSc:=rSc;
 FRamp:=rRamp;
 FTup:=rTup;
 FTdown:=rTdown;
 FTcold:=rTcold;
 FInitSt:=rInitSt;
end;

constructor
TPembangkit.Create(co
nst
rPembangkit:TPemban
gkit);
begin
 inherited Create;

FNama:=rPembangkit.
Nama;

FPmin:=rPembangkit.P
min;

FPmax:=rPembangkit.
Pmax;
 Fa2:=rPembangkit.a2;
 Fa1:=rPembangkit.a1;
 Fa0:=rPembangkit.a0;

FSh:=rPembangkit.Sh;
 FSc:=rPembangkit.Sc;

FRamp:=rPembangkit.
Ramp;

FTup:=rPembangkit.Tu
p;

FTdown:=rPembangkit
.Tdown;

FTcold:=rPembangkit.
Tcold;

FInitSt:=rPembangkit.I
nitSt;
end;

function
TPembangkit.GetAFL
C:double;
begin

Result:=fa0/fPmax+fa1
+fa2*fPmax;
end;

procedure
TPembangkit.Assign(c
onst
rPembangkit:TPemban
gkit);
begin

FNama:=rPembangkit.
Nama;

FPmin:=rPembangkit.P
min;

FPmax:=rPembangkit.
Pmax;
 Fa2:=rPembangkit.a2;
 Fa1:=rPembangkit.a1;
 Fa0:=rPembangkit.a0;

FSh:=rPembangkit.Sh;
 FSc:=rPembangkit.Sc;

FRamp:=rPembangkit.
Ramp;

FTup:=rPembangkit.Tu
p;

FTdown:=rPembangkit
.Tdown;

FTcold:=rPembangkit.
Tcold;

FInitSt:=rPembangkit.I
nitSt;
end;

//data operation

function
TPembangkit.GetBiaya
(const
rDaya:double):double;
begin
 result:=0;
 if rDaya<>0 then
 begin

result:=Fa2*sqr(rDaya)
+Fa1*rDaya+Fa0;
 end;
```

```
end;

function
TPembangkit.GetDaya(
const
rLamda:double):double
;
begin
 result:=(rLamda-
Fa1)/(2*Fa2);
 if result>FPmax then
result:=FPmax;
 if result<FPmin then
result:=FPmin;
end;

function
TPembangkit.GetLamd
a(const
rDaya:double):double;
begin

result:=2*Fa2*rDaya+
Fa1;
end;

end.


unit uObjFunc;

interface

uses
uUtils,uGenerator,Sys
Utils,uMatrix;

type
 TObjFunc=class
 private

FNgen,FNjam:integer;

FBeban,FRes,FAFLC:
dArr1;
  FSortAFLC:iArr1;
  FPLN:dArr2;
  FGen:TGenArr;
  function
GetBeban:dArr1;
  function
GetRes:dArr1;
  function
GetPLN:dArr2;
  function
GetGen:TGenArr;
  procedure
SetGen(const
rGen:TGenArr);
  procedure
SetBeban(const
rBeban:dArr1);
  procedure
SetRes(const
rRes:dArr1);
  procedure
SetPLN(const
rPLN:dArr2);
  function isON(const
rFlip:double):boolean;
  function
isServe(const
rJam:integer;const
rChrom:bArr1):boolea
n;
  function
isRampRate(const
rJam:integer;const
rPL:dArr2):boolean;
  function
FindAFLC(const
ri:integer):integer;
  procedure
RepairAFLC(var
rChrom:bArr2);
  function
CreateChromBase:bArr
2;
  function
CreateChromONOFF:b
Arr2;
  function
HitungEcoDis(const
rJam:integer;
        const
rChrom1:bArr1):dArr1
;
  function
GetSortAFLC:iArr1;
  function
GetSortChrom(const
rRank:integer):bArr1;
  function
HitungCostGen(const
rPL:dArr2):dArr2;
  function
HitungCostSUC(const
rPL:dArr2):dArr2;
  function
GantiChrom(const
rChrom:bArr2):bArr2;
  function
GetSwap(const
rChrom:bArr2):bArr2;
  function
doCariGreyZone(const
rChrom:bArr2):bArr2;
 public
  constructor
Create;overload;
  constructor
Create(const
rBeban,rRes:dArr1;
        const
rPLN:dArr2;
        const
rGen:TGenArr);overlo
ad;
  function
getRandomChrom(con
st rFlip:double):bArr2;
  function
getConstructSolution(c
onst
rFlip:double):bArr2;
  procedure
setLocalSearch(var
rChrom:bArr2);
  procedure
doHitungChrom(const
rChrom:bArr2;
        var
rCostTotal:double);ove
rload;
  procedure
doHitungChrom(const
rChrom:bArr2;
        var rPL:dArr2;
```

```
        var
rCostPerJam:dArr1;
        var
rCostTotal:double);ove
rload;
  procedure
doHitungPLN(
        var
rCostPerJam:dArr1;
        var
rCostTotal:double);
  procedure
doHitungPL(const
rPL:dArr2;
        var
rCostPerJam:dArr1;
        var
rCostTotal:double);
  procedure
doExecute(var
rChrom:bArr2;
        var rPL:dArr2;
        var
rCostPerJam:dArr1;
        var
rCostTotal:double);
  function
HitungEcoDisGrad(con
st rIterasi,rJam:integer;
        const
rAlpha:double;
        const
rChrom1:bArr1):dArr1
;
  destructor
Destroy;override;
  property
Ngen:integer read
FNgen write FNgen;
  property
Njam:integer read
FNjam write FNjam;
  property
Gen:TGenArr read
GetGen write SetGen;
  property
Beban:dArr1 read
GetBeban write
SetBeban;
  property PLN:dArr2
read GetPLN write
SetPLN;
  property Res:dArr1
read GetRes write
SetRes;
  property
SortAFLC:iArr1 read
GetSortAFLC;
 end;

var
gObjFunc:TObjFunc;

implementation

//constructor
constructor
TObjFunc.Create;
begin
 inherited Create;
 FNgen:=0;
 FNjam:=0;
end;

constructor
TObjFunc.Create(const
rBeban,rRes:dArr1;
        const
rPLN:dArr2;
        const
rGen:TGenArr);
var i,j,Ncek:integer;
begin
 inherited Create;
 FNgen:=high(rGen);

FNjam:=high(rBeban);
 Ncek:=high(rRes);
 if FNjam<>Ncek then
raise
Exception.Create('Dim
ensi matrik tidak
sama!');

SetLength(FGen,FNge
n+1);

SetLength(FBeban,FNj
am+1);

SetLength(FRes,FNjam
+1);

SetLength(FAFLC,FN
gen+1);
 for i:=1 to FNgen do
 begin

FGen[i]:=TPembangkit
.Create(rGen[i]);

FAFLC[i]:=FGen[i].A
FLC;
 end;
 for i:=1 to FNjam do
 begin

FBeban[i]:=rBeban[i];
  FRes[i]:=rRes[i];
 end;

SetLength(FPLN,FNge
n+1,FNjam+1);
 for i:=1 to FNgen do
 begin
  for j:=1 to FNjam do
  begin

FPLN[i,j]:=rPLN[i,j];
  end;
 end;

FSortAFLC:=GetSortA
FLC;
end;

//data accessing
function
TObjFunc.GetBeban:d
Arr1;
var i:integer;
begin

SetLength(result,FNja
m+1);
 for i:=1 to FNjam do
 begin
  result[i]:=FBeban[i];
 end;
```

```
end;

function
TObjFunc.GetRes:dArr
1;
var i:integer;
begin

SetLength(result,FNja
m+1);
  for i:=1 to FNjam do
  begin
    result[i]:=FRes[i];
  end;
end;

function
TObjFunc.GetPLN:dAr
r2;
var i,j:integer;
begin

SetLength(result,FNge
n+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin

result[i,j]:=FPLN[i,j];
    end;
  end;
end;

function
TObjFunc.GetGen:TGe
nArr;
var i:integer;
begin

SetLength(result,FNge
n+1);
  for i:=1 to FNgen do
  begin

result[i]:=TPembangkit
.Create(FGen[i]);
  end;
end;

procedure
TObjFunc.SetGen(cons
t rGen:TGenArr);
var i:integer;
begin
  FNgen:=high(rGen);

SetLength(FGen,FNge
n+1);

SetLength(FAFLC,FN
gen+1);
  for i:=1 to FNgen do
  begin
FGen[i]:=TPembangkit
.Create(rGen[i]);

FAFLC[i]:=FGen[i].A
FLC;
  end;

FSortAFLC:=GetSortA
FLC;
end;

procedure
TObjFunc.SetBeban(co
nst rBeban:dArr1);
var i,Ncek:integer;
begin
  if FNjam>0 then
  begin

Ncek:=high(rBeban);
    if FNjam<>Ncek
then raise
Exception.Create('Dim
ensi matrik tidak
sama!');
  end
  else
  begin

FNjam:=high(rBeban);
  end;

SetLength(FBeban,FNj
am+1);
  for i:=1 to FNjam do
  begin
FBeban[i]:=rBeban[i];
  end;
end;

procedure
TObjFunc.SetRes(cons
t rRes:dArr1);
var i,Ncek:integer;
begin
  if FNjam>0 then
  begin
    Ncek:=high(rRes);
    if FNjam<>Ncek
then raise
Exception.Create('Dim
ensi matrik tidak
sama!');
  end
  else
  begin
    FNjam:=high(rRes);
  end;

SetLength(FRes,FNjam
+1);
  for i:=1 to FNjam do
  begin
    FRes[i]:=rRes[i];
  end;
end;

procedure
TObjFunc.SetPLN(con
st rPLN:dArr2);
var i,j,Ncek:integer;
begin
  if FNgen>0 then
  begin
    Ncek:=high(rPLN);
    if FNgen<>Ncek
then raise
Exception.Create('Dim
ensi matrik tidak
sama!');
  end
  else
  begin
    FNgen:=high(rPLN);
  end;
```

```
 if FNjam>0 then
 begin

Ncek:=high(rPLN[0]);
   if FNjam<>Ncek
then raise
Exception.Create('Dim
ensi matrik tidak
sama!');
 end
 else
 begin

FNjam:=high(rPLN[0])
;
 end;

SetLength(FPLN,FNge
n+1,FNjam+1);
 for i:=1 to FNgen do
 begin
   for j:=1 to FNjam do
   begin

FPLN[i,j]:=rPLN[i,j];
   end;
 end;
end;

//data processing
function
TObjFunc.isON(const
rFlip:double):boolean;
begin
 result:=false;
 if random<=rFlip then
result:=true;
end;

function
TObjFunc.isServe(cons
t rJam:integer;const
rChrom:bArr1):boolea
n;
var i:integer;

load,sBebanMin,sBeba
nMax:double;
begin
 result:=true;
 sBebanMin:=0;
 sBebanMax:=0;
 for i:=1 to FNgen do
 begin
   if rChrom[i]=true
then
   begin

sBebanMin:=sBebanM
in+FGen[i].Pmin;

sBebanMax:=sBebanM
ax+FGen[i].Pmax;
   end;
 end;

load:=FBeban[rJam]+F
Res[rJam];
 if load<sBebanMin
then result:=false;
 if load>sBebanMax
then result:=false;
end;

function
TObjFunc.isRampRate
(const
rJam:integer;const
rPL:dArr2):boolean;
var i:integer;
   delta:double;
begin
 result:=true;
 for i:=1 to FNgen do
 begin
   if rJam>1 then
   begin
     delta:=rPL[i,rJam]-
rPL[i,rJam-1];
     if delta>0 then
     begin
       if
delta>FGen[i].Ramp
then
       begin
         result:=false;
         break;
       end;
     end;
   end;
 end;
end;

function
TObjFunc.HitungEcoD
is(const rJam:integer;
      const
rChrom1:bArr1):dArr1
;
var i,j:integer;
   Status:bArr1;

LoadCek,Pa,Pb,Lmd,L
oadSplit:double;

aBeban,diffa2,diffa1,C
ek,tes:double;
begin

SetLength(Status,FNge
n+1);
 for i:=1 to FNgen do
 begin

Status[i]:=rChrom1[i];
   FGen[i].Daya:=0;
 end;

aBeban:=FBeban[rJam
];
 LoadCek:=aBeban;
 LoadSplit:=aBeban;
 for i:=1 to 15 do
 begin
   Pa:=0;
   Pb:=0;
   for j:=1 to FNgen do
   begin
     if Status[j] then
     begin

diffa2:=FGen[j].a2*2;

diffa1:=FGen[j].a1;
       Pa:=Pa+1/diffa2;

Pb:=Pb+diffa1/diffa2;
     end;
   end;
   if Pa<>0 then
```

```
  begin

Lmd:=(LoadSplit+Pb)/
Pa;
  end
  else
  begin

Lmd:=LoadSplit+Pb;
  end;
  Cek:=0;
  for j:=1 to FNgen do
  begin
    if Status[j],then
    begin

diffa2:=2*FGen[j].a2;

diffa1:=FGen[j].a1;

FGen[j].Daya:=(Lmd-
diffa1)/diffa2;
      if
FGen[j].Daya<FGen[j].
Pmin then
      begin

FGen[j].Daya:=FGen[j]
.Pmin;
      end;
      if
FGen[j].Daya>FGen[j].
Pmax then
      begin

FGen[j].Daya:=FGen[j]
.Pmax;
      end;
    end;

Cek:=Cek+FGen[j].Da
ya;
  end;
  tes:=LoadCek-Cek;
  if (tes<0.0001) and
(tes>-0.0001) then
  begin
    break;
  end
  else if tes>0 then
  begin
    for j:=1 to fNgen
do
    begin
      if Status[j] then
      begin
        if
FGen[j].Daya=FGen[j].
PMax then
        begin
          Status[j]:=false;

LoadSplit:=LoadSplit-
FGen[j].Daya;
          if LoadSplit<0
then
          begin

LoadSplit:=LoadSplit+
fGen[j].Daya;

Status[j]:=true;
          end;
        end;
      end;
    end;
  end
  else if tes<0 then
  begin
    for j:=1 to fNgen
do
    begin
      if Status[j] then
      begin
        if
FGen[j].Daya=FGen[j].
Pmin then
        begin
          Status[j]:=false;

LoadSplit:=LoadSplit-
FGen[j].Daya;
          if LoadSplit<0
then
          begin

LoadSplit:=LoadSplit+
FGen[j].Daya;

Status[j]:=true;
          end;
        end;
      end;
    end;
  end;
end;

SetLength(result,FNge
n+1);
 for i:=1 to FNgen do
 begin
   result[i]:=0;
   if rChrom1[i] then
   begin

result[i]:=FGen[i].Daya
;
   end;
 end;
end;

function
TObjFunc.HitungEcoD
isGrad(const
rIterasi,rJam:integer;
      const
rAlpha:double;
      const
rChrom1:bArr1):dArr1
;

function
CariChrom1(const
rJam:integer):bArr1;
var i:integer;
begin

SetLength(result,FNge
n+1);
 for i:=1 to FNgen do
 begin
   if FPLN[i,rJam]<>0
then
   begin
     result[i]:=true;
   end
   else
   begin
     result[i]:=false;
   end;
```

```
 end;
end;

function InitDaya(const
rJam:integer;
      const
rChrom1:bArr1;
      var
rNgenON:integer):dArr
1;
var i,sa:integer;

sBebanMin,dBeban,tm
p:double;
  Pgen:dArr1;
begin

SetLength(Pgen,FNgen
+1);
 sBebanMin:=0;
 rNgenON:=0;
 for i:=1 to FNgen do
 begin
  sa:=FSortAFLC[i];
  Pgen[sa]:=0;
  if rChrom1[sa]=true
then
  begin
    inc(rNgenON);

Pgen[sa]:=FGen[sa].P
min;

sBebanMin:=sBebanM
in+Pgen[sa];
  end;
 end;

dBeban:=FBeban[rJam
]-sBebanMin;
 for i:=1 to FNgen do
 begin
  sa:=FSortAFLC[i];
  if rChrom1[sa]=true
then
  begin
   if dBeban>0 then
   begin
    tmp:=Pgen[sa];

Pgen[sa]:=Pgen[sa]+d
Beban;
      if
Pgen[sa]>FGen[sa].Pm
ax then
      begin

Pgen[sa]:=FGen[sa].P
max;
      end;
      dBeban:=dBeban-
(Pgen[sa]-tmp);
    end;
   end;
  end;

SetLength(result,rNgen
ON+1);
 sa:=0;
 for i:=1 to FNgen do
 begin
  if rChrom1[i]=true
then
  begin
   inc(sa);
   result[sa]:=Pgen[i];
  end;
 end;
end;
function
CariAveLamda(const
rPgen:dArr1;
      const
rChrom1:bArr1):doubl
e;
var i,sa:integer;
  sLamda:double;
begin
 sLamda:=0;
 sa:=0;
 for i:=1 to FNgen do
 begin
  if rChrom1[i]=true
then
  begin
   inc(sa);

sLamda:=sLamda+FGe
n[i].GetLamda(rPgen[s
a]);
  end;
 end;
 result:=sLamda/sa;
end;

function
CariPgen(const
rChrom:bArr1;
      const
rLamda:double):dArr1;
var i:integer;
begin

SetLength(result,FNge
n+2);
 for i:=1 to FNgen do
 begin
  if rChrom[i]=true
then
  begin

result[i]:=FGen[i].Get
Daya(rLamda);
  end;
 end;
end;

function CaridL(const
rJam:integer;
      const
rChrom1:bArr1;
      const
rPgen:dArr1;
      const
rLamda:double):dArr1;
var i,sa:integer;

minLamda,maxLamda,
sumPgen:double;
begin

SetLength(result,high(r
Pgen)+2);
 sumPgen:=0;
 sa:=0;
 for i:=1 to FNgen do
 begin
```

```
  if rChrom1[i]=true
then
  begin
    inc(sa);

minLamda:=FGen[i].G
etLamda(FGen[i].Pmin
);

maxLamda:=FGen[i].G
etLamda(FGen[i].Pmax
);

sumPgen:=sumPgen+r
Pgen[sa];
    if
(rLamda<minLamda)
or
(rLamda>maxLamda)
then
    //if
(maxLamda<rLamda)
then
    begin
      result[sa]:=0;
    end
    else
    begin

result[sa]:=FGen[i].Get
Lamda(rPgen[sa])-
rLamda;
    end;
  end;
 end;

result[sa+1]:=FBeban[r
Jam]-sumPgen;
end;

function
CariMaxdL(const
rdL:dArr1):double;
var i:integer;
begin
 result:=rdL[1];
 for i:=2 to high(rdL)
do
 begin
  if result<abs(rdL[i])
then
  begin
    result:=abs(rdL[i]);
  end;
 end;
end;

function
MatrixHessian(const
rChrom1:bArr1):dArr2
;
var
i,j,sa,sb,NgenON:integ
er;
begin
 NgenON:=0;
 for i:=1 to FNgen do
 begin
  if rChrom1[i]=true
then
  begin
    inc(NgenON);
  end;
 end;

SetLength(result,Ngen
ON+2,NgenON+2);
 sa:=0;
 for i:=1 to FNgen do
 begin
  if rChrom1[i]=true
then
  begin
    inc(sa);
    sb:=0;
    for j:=1 to FNgen
do
    begin
      if rChrom1[j]=true
then
      begin
        inc(sb);
        if sb=sa then
        begin

result[sa,sb]:=2*FGen[i
].a2;
        end
        else
        begin
          result[sa,sb]:=0;
        end;
      end;
    end;
  end;
 end;
 sa:=0;
 for i:=1 to FNgen do
 begin
  if rChrom1[i]=true
then
  begin
    inc(sa);

result[NgenON+1,sa]:=
-1;

result[sa,NgenON+1]:=
-1;
  end;
 end;

result[NgenON+1,Nge
nON+1]:=0;
end;

function
BalikKeSemula(const
rPgen:dArr1;
      const
rChrom1:bArr1):dArr1
;
var i,sa:integer;
begin

SetLength(result,FNge
n+1);
 sa:=0;
 for i:=1 to FNgen do
 begin
  if rChrom1[i]=true
then
  begin
    inc(sa);

result[i]:=rPgen[sa];
  end;
 end;
end;
```

```
procedure
UpdateLamda(const
rdL:dArr1;
      const
rChrom1:bArr1;
      const
rHessian:dArr2;
      var rPgen:dArr1;
      var
rLamda:double);
var i,sa:integer;
  invHessian:dArr2;
  dX:dArr1;
begin

invHessian:=MatrixInv
ers(rHessian);

invHessian:=MatrixNe
gative(invHessian);

dX:=MatrixMul(invHe
ssian,rdL);
 sa:=0;
 for i:=1 to FNgen do
 begin
  if rChrom1[i]=true
then
  begin
   inc(sa);

rPgen[sa]:=rPgen[sa]+
dX[sa];
   if
rPgen[sa]>FGen[i].Pm
ax then
rPgen[sa]:=FGen[i].Pm
ax;
   if
rPgen[sa]<FGen[i].Pmi
n then
rPgen[sa]:=FGen[i].Pm
in;
  end;
 end;

rLamda:=rLamda+dX[
FNgen+1];
end;
```

```
var i,NgenON:integer;
  status:bArr1;
  Hessian:dArr2;
  Pgen,dL:dArr1;

sPmax,sPmin,Lamda,A
lpha,error,max:double;
begin

SetLength(status,FNge
n+1);
 for i:=1 to FNgen do
 begin

status[i]:=rChrom1[i];
 end;
 sPmax:=0;
 sPmin:=0;
 for i:=1 to FNgen do
 begin
  if status[i]=true then
  begin

sPmin:=sPmin+FGen[i
].Pmin;

sPmax:=sPmax+FGen[
i].Pmax;
  end;
 end;

SetLength(result,FNge
n+1);
 if
sPmin>FBeban[rJam]
then
 begin
  for i:=1 to FNgen do
  begin
   if status[i]=true
then
   begin

result[i]:=FGen[i].Pmin
;
   end;
  end;
 end
```

```
 else if
sPmax<FBeban[rJam]
then
 begin
  for i:=1 to FNgen do
  begin
   if status[i]=true
then
   begin

result[i]:=FGen[i].Pma
x;
   end;
  end;
 end
 else
 begin
  error:=rAlpha;

Pgen:=InitDaya(rJam,r
Chrom1,NgenON);

Lamda:=CariAveLamd
a(Pgen,rChrom1);
  for i:=1 to rIterasi do
  begin
dL:=CaridL(rJam,rChr
om1,Pgen,Lamda);

max:=cariMaxdL(dL);
   if max<error then
break;

Hessian:=MatrixHessia
n(rChrom1);

UpdateLamda(dL,rChr
om1,Hessian,Pgen,Lam
da);
  end;

result:=BalikKeSemula
(Pgen,rChrom1);
 end;
end;

function
TObjFunc.HitungCost
Gen(const
rPL:dArr2):dArr2;
```

```
var i,j:integer;
begin

SetLength(result,FNge
n+1,FNjam+1);
 for i:=1 to FNgen do
 begin
  for j:=1 to FNjam do
  begin

result[i,j]:=FGen[i].Get
Biaya(rPL[i,j]);
  end;
 end;
end;

function
TObjFunc.HitungCostS
UC(const
rPL:dArr2):dArr2;
var i,j,init,tcold:integer;
begin

SetLength(result,FNge
n+1,FNjam+1);
 for i:=1 to FNgen do
 begin
  init:=FGen[i].InitSt;

tcold:=FGen[i].Tdown
+FGen[i].Tcold;
  for j:=1 to FNjam do
  begin
   result[i,j]:=0;
   if rPL[i,j]<>0 then
   begin
    if init>0 then
    begin
     init:=init+1;
    end
    else if init<0 then
    begin
     if
abs(init)<=tcold then
      begin

result[i,j]:=FGen[i].Sh;
      end
      else
      begin
```

```
result[i,j]:=FGen[i].Sc;
      end;
      init:=1;
     end;
    end
    else if rPL[i,j]=0
then
    begin
     if init>0 then
     begin
      init:=-1;
     end
     else if init<0 then
     begin
      init:=init-1;
     end;
    end;
   end;
  end;
end;

procedure
TObjFunc.doHitungCh
rom(const
rChrom:bArr2;
      var
rCostTotal:double);
var i,j:integer;
  PLa:dArr1;

PL,CostGen,CostSUC:
dArr2;
  chrom1:bArr1;
begin

SetLength(PL,FNgen+
1,FNjam+1);

SetLength(chrom1,FNg
en+1);

SetLength(PLa,FNgen
+1);
 for i:=1 to FNjam do
 begin
  for j:=1 to FNgen do
  begin
```

```
chrom1[j]:=rChrom[j,i]
;
  end;

PLa:=HitungEcoDis(i,c
hrom1);
  for j:=1 to FNgen do
  begin
   PL[j,i]:=PLa[j];
  end;
 end;

CostGen:=HitungCost
Gen(PL);

CostSUC:=HitungCost
SUC(PL);
 rCostTotal:=0;
 for i:=1 to FNjam do
 begin
  for j:=1 to FNgen do
  begin

rCostTotal:=rCostTotal
+CostGen[j,i]+CostSU
C[j,i];
  end;
 end;
end;

procedure
TObjFunc.doHitungCh
rom(const
rChrom:bArr2;
      var rPL:dArr2;
      var
rCostPerJam:dArr1;
      var
rCostTotal:double);
var i,j:integer;
  PLa:dArr1;

CostGen,CostSUC:dAr
r2;
  chrom1:bArr1;
begin

SetLength(rPL,FNgen+
1,FNjam+1);
```

```
SetLength(rCostPerJa
m,FNjam+1);

SetLength(chrom1,FNg
en+1);

SetLength(PLa,FNgen
+1);
 for i:=1 to FNjam do
 begin
  for j:=1 to FNgen do
  begin

chrom1[j]:=rChrom[j,i]
;
  end;

PLa:=HitungEcoDis(i,c
hrom1);
  for j:=1 to FNgen do
  begin
   rPL[j,i]:=PLa[j];
  end;
 end;

CostGen:=HitungCost
Gen(rPL);

CostSUC:=HitungCost
SUC(rPL);

SetLength(rCostPerJa
m,FNjam+1);
 rCostTotal:=0;
 for i:=1 to FNjam do
 begin
  rCostPerJam[i]:=0;
  for j:=1 to FNgen do
  begin

rCostPerJam[i]:=rCost
PerJam[i]+CostGen[j,i]
+CostSUC[j,i];
  end;

rCostTotal:=rCostTotal
+rCostPerJam[i];
 end;
end;
```

```
procedure
TObjFunc.doHitungPL
(const rPL:dArr2;
      var
rCostPerJam:dArr1;
      var
rCostTotal:double);
var i,j:integer;

CostGen,CostSUC:dAr
r2;
begin

SetLength(rCostPerJa
m,FNjam+1);

CostGen:=HitungCost
Gen(rPL);

CostSUC:=HitungCost
SUC(rPL);
 rCostTotal:=0;
 for i:=1 to FNjam do
 begin
  rCostPerJam[i]:=0;
  for j:=1 to FNgen do
  begin

rCostPerJam[i]:=rCost
PerJam[i]+CostGen[j,i]
+CostSUC[j,i];
  end;

rCostTotal:=rCostTotal
+rCostPerJam[i];
 end;
end;

procedure
TObjFunc.doHitungPL
N(var
rCostPerJam:dArr1;
      var
rCostTotal:double);
var i,j:integer;

CostGen,CostSUC:dAr
r2;
begin
```

```
SetLength(rCostPerJa
m,FNjam+1);

CostGen:=HitungCost
Gen(FPLN);

CostSUC:=HitungCost
SUC(FPLN);
 rCostTotal:=0;
 for i:=1 to FNjam do
 begin
  rCostPerJam[i]:=0;
  for j:=1 to FNgen do
  begin

rCostPerJam[i]:=rCost
PerJam[i]+CostGen[j,i]
+CostSUC[j,i];
  end;

rCostTotal:=rCostTotal
+rCostPerJam[i];
 end;
end;

//data output
function
TObjFunc.GetSortAFL
C:iArr1;
var i,j,tmp:integer;
  tmpAFLC:double;
begin

SetLength(result,FNge
n+1);
 for i:=1 to FNgen do
 begin
  result[i]:=i;
 end;
 for i:=1 to fNgen-1 do
 begin
  for j:=i to fNgen do
  begin
   if
FAFLC[i]>FAFLC[j]
then
   begin

tmpAFLC:=FAFLC[i];
```

```
FAFLC[i]:=FAFLC[j];

FAFLC[j]:=tmpAFLC;
      tmp:=result[i];
      result[i]:=result[j];
      result[j]:=tmp;
    end;
  end;
 end;
 for i:=1 to fNgen do
 begin

fAFLC[i]:=fGen[i].AF
LC;
 end;
end;

function
TObjFunc.GetSortChro
m(const
rRank:integer):bArr1;
var i:integer;
begin

SetLength(result,FNge
n+1);
 for i:=1 to FNgen do
 begin
  result[i]:=false;
 end;
 for i:=1 to rRank do
 begin

result[FSortAFLC[i]]:=
true;
 end;
end;

function
TObjFunc.FindAFLC(c
onst ri:integer):integer;
var i:integer;
begin
 result:=1;
 for i:=1 to FNgen do
 begin
  if FSortAFLC[i]=ri
then
   begin
    result:=i;
    break;
   end;
  end;
end;

procedure
TObjFunc.RepairAFL
C(var rChrom:bArr2);
var i,j,pos,k:integer;
begin
 for i:=1 to FNjam do
 begin
  for j:=1 to Ngen do
  begin
   if rChrom[j,i]=true
then
    begin

pos:=FindAFLC(j);
      for k:=1 to pos do
      begin
       if
rChrom[FSortAFLC[k]
,i]=false then
        begin

rChrom[FSortAFLC[k]
,i]:=true;

rChrom[j,i]:=false;
          break;
        end;
      end;
    end;
   end;
  end;
end;

function
TObjFunc.CreateChro
mBase:bArr2;
var i,j,k:integer;
   chrom1:bArr1;
   serve:boolean;
begin

SetLength(result,FNge
n+1,FNjam+1);

SetLength(chrom1,FNg
en+1);
 for i:=1 to FNjam do
 begin
  for j:=1 to FNgen do
  begin

chrom1:=GetSortChro
m(j);

serve:=isServe(i,chrom
1);
    if serve=true then
    begin
     for k:=1 to FNgen
do
     begin

result[k,i]:=chrom1[k];
     end;
     break;
    end;
   end;
  end;
end;

function
TObjFunc.CreateChro
mONOFF:bArr2;
var i,j,k:integer;
   chrom1:bArr1;
   serve:boolean;
begin

SetLength(result,FNge
n+1,FNjam+1);

SetLength(chrom1,FNg
en+1);
 for i:=1 to FNjam do
 begin
  for j:=FNgen downto
1 do
   begin

chrom1:=GetSortChro
m(j);
```

```
serve:=isServe(i,chrom
1);
    if serve=true then
    begin
      for k:=1 to FNgen
do
      begin

result[k,i]:=chrom1[k];
      end;
      break;
    end;
  end;
 end;
end;

function
TObjFunc.getRandom
Chrom(const
rFlip:double):bArr2;
var i,j:integer;

chromBase,chromONO
FF:bArr2;
begin

chromBase:=CreateChr
omBase;

chromONOFF:=Create
ChromONOFF;

SetLength(result,Ngen
+1,Njam+1);
 for i:=1 to Ngen do
 begin
  for j:=1 to Njam do
  begin
    if
chromBase[i,j]=true
then
    begin
      result[i,j]:=true;
    end
    else
    begin
      if
chromONOFF[i,j]=true
then
      begin
        if
isON(rFlip)=true then
        begin
          result[i,j]:=true;
        end
        else
        begin

result[i,j]:=false;
        end;
      end;
    end;
  end;
 end;
 RepairAFLC(result);

result:=GetSwap(result
);
end;
function
TObjFunc.getConstruct
Solution(const
rFlip:double):bArr2;
var i,j:integer;

chromBase,chromONO
FF:bArr2;
begin

chromBase:=CreateChr
omBase;

chromONOFF:=Create
ChromONOFF;

SetLength(result,Ngen
+1,Njam+1);
 for i:=1 to Ngen do
 begin
  for j:=1 to Njam do
  begin
    if
chromBase[i,j]=true
then
    begin
      result[i,j]:=true;
    end
    else
    begin
      if
chromONOFF[i,j]=true
then
      begin
        if
isON(rFlip)=true then
        begin
          result[i,j]:=true;
        end
        else
        begin

result[i,j]:=false;
        end;
      end;
    end;
  end;
 end;
 RepairAFLC(result);

result:=GetSwap(result
);
end;

procedure
TObjFunc.setLocalSear
ch(var rChrom:bArr2);
var i,j:integer;

CostBest,CostCek:dou
ble;
  greyChrom:bArr2;
begin

doHitungChrom(rChro
m,CostBest);

greyChrom:=doCariGr
eyZone(rChrom);
 for i:=1 to FNgen do
 begin
  for j:=1 to FNjam do
  begin
    if
greyChrom[i,j]=true
then
    begin
      rChrom[i,j-
1]:=true;
```

```
doHitungChrom(rChro
m,CostCek);
      if
CostBest>CostCek
then
      begin

CostBest:=CostCek;
      end
      else
      begin
        rChrom[i,j-
1]:=false;
      end;
     end;
   end;
  end;
end;

function
TObjFunc.GetSwap(co
nst
rChrom:bArr2):bArr2;
var
i,j,k,init,pos:integer;
begin

SetLength(result,FNge
n+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    init:=FGen[i].InitSt;
    for j:=1 to FNjam do
    begin

result[i,j]:=rChrom[i,j];
      if result[i,j]=true
then
      begin
        if init<0 then
        begin
          if
abs(init)>=FGen[i].Tdo
wn then
          begin
            init:=1;
          end
          else
          begin
            pos:=j+init;
            if pos<1 then
            begin
              pos:=1;
            end;
            for k:=pos to j-1
do
            begin

result[i,k]:=true;
            end;
          end;
        end
        else if init>0 then
        begin
          init:=init+1;
        end;
      end
      else if
result[i,j]=false then
      begin
        if init<0 then
        begin
          init:=init-1;
        end
        else if init>0 then
        begin
          if
init>=FGen[i].Tup then
          begin
            init:=-1;
          end
          else
          begin
            result[i,j]:=true;
            init:=init+1;
          end;
        end;
      end;
    end;
  end;
end;

//destructor
destructor
TObjFunc.Destroy;
var i:integer;
begin
  try
    for i:=1 to FNgen do
    begin
      FGen[i].Free;
    end;
  finally
    inherited Destroy;
  end;
end;

function
TObjFunc.GantiChrom
(const
rChrom:bArr2):bArr2;
var i,j:integer;
begin

SetLength(result,FNge
n+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin

result[i,j]:=rChrom[i,j];
    end;
  end;
end;

function
TObjFunc.doCariGrey
Zone(const
rChrom:bArr2):bArr2;
var i,j,init,tcold:integer;
begin

SetLength(result,FNge
n+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      result[i,j]:=false;
    end;
  end;
  for i:=1 to FNgen do
  begin
    init:=FGen[i].InitSt;

tcold:=FGen[i].Tdown
+FGen[i].Tcold;
    for j:=1 to FNjam do
```

```
  begin
    if rChrom[i,j]=true
then
    begin
      if init<0 then
      begin
        if
abs(init)=(tcold+1)
then
        begin
          result[i,j]:=true;
        end;
        init:=1;
      end
      else if init>0 then
      begin
        init:=init+1;
      end;
    end
    else if
rChrom[i,j]=false then
    begin
      if init<0 then
      begin
        init:=init-1;
      end
      else if init>0 then
      begin
        init:=-1;
      end;
    end;
  end;
 end;
end;

procedure
TObjFunc.doExecute(v
ar rChrom:bArr2;
      var rPL:dArr2;
      var
rCostPerJam:dArr1;
      var
rCostTotal:double);
var i,j:integer;

CostBest,CostCek:dou
ble;

chromBase,greyChrom
:bArr2;

begin

chromBase:=CreateChr
omBase;

rChrom:=GetSwap(chr
omBase);

doHitungChrom(rChro
m,CostBest);

greyChrom:=doCariGr
eyZone(rChrom);
 for i:=1 to FNgen do
 begin
   for j:=1 to FNjam do
   begin
     if
greyChrom[i,j]=true
then
     begin
       rChrom[i,j-
1]:=true;

doHitungChrom(rChro
m,CostCek);
       if
CostBest>CostCek
then
       begin

CostBest:=CostCek;
       end
       else
       begin
         rChrom[i,j-
1]:=false;
       end;
     end;
   end;
 end;

doHitungChrom(rChro
m,rPL,rCostPerJam,rC
ostTotal);
end;

end.
```