"OPTIMASI KOMITMEN UNIT MENGGUNAKAN METODE *GENETIC ALGORITHM - SIMULATED ANNEALING* (GASA) PADA PEMBANGKITAN JAWA - BALI II (PJB II)"

SKRIPSI

Disusun Oleh :

RACHMANTO JUNAIDI
89.12.097

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

MARET 2007

LEMBAR PERSETUJUAN

OPTIMASI KOMITMEN UNIT MENGGUNAKAN METODE *GENETIC ALGORITHM SIMULATED ANNEALING* (GASA) PADA PEMBANGKITAN JAWA BALI II (PJB II)

SKRIPSI

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

Disusun Oleh :

RACHMANTO JUNAIDI
NIM : 89.12.097

Mengetahui
Ketua Jurusan Teknik Elektro

(Ir. F Yudi Limpraptono, MT)
NIP. Y. 103 95 00274

Diperiksa dan disetujui
Dosen Pembimbing

(Ir. Yusuf Ismail Nakhoda, MT)
NIP.Y. 101 88 00189

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

KATA PENGANTAR

Dengan memanjatkan puji syukur keahdirat Allah SWT, atas limpahan Rahmat dan Hidayah-Nya, maka penulisan Skripsi yang merupakan salah satu syarat dalam menyelesaikan studi program strata satu (S-1) Jurusan Teknik Elektro Konsentrasi Energi Listrik, Fakultas Teknologi Industri, Institut Teknologi Nasional Malang ini dapat terselesaikan.

Sebelum dan selama penulisan Skripsi, penulis telah banyak mendapatkan bantuan dan bimbingan dari berbagai pihak. Untuk itu pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada :

1. Bapak Dr. Ir. Abraham Lomi, MSEE, selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. F Yudi Limpraptono, MT, selaku Ketua Jurusan Teknik Elektro Konsentrasi Teknik Energi Listrik, Institut Teknologi Nasional Malang.
3. Bapak Ir.Yusuf Ismail Nakhoda, MT, selaku Dosen Pembimbing penulisan Skripsi ini.
4. Bapak-bapak dan ibu-ibu dosen Jurusan Teknik Elektro yang telah mendidik memberikan ilmunya kepada kami.
5. Bapak dan ibuku, serta keluargaku, yang sangat berarti dalam kehidupan penulis, dimana doa dan keridhaannya senantiasa penulis harapkan.
6. Teman-teman di Jurusan Teknik Elektro Institut Teknologi Nasional Malang yang telah banyak membantu dalam penyelesaian Skripsi ini.

7. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah membantu dalam penyelesaian Skripsi ini.

Penulis menyadari sepenuhnya akan segala kekurangan yang ada dalam penulisan Skripsi ini, maka dengan segala kerendahan hati penulis mengharapkan kritik dan saran demi penyempurnaan Skripsi ini.

Malang, Maret 2007

Penulis

ABSTRAKSI

OPTIMASI KOMITMEN UNIT MENGGUNAKAN METODE *GENETIC ALGORITHM SIMULATED ANNEALING* (GASA) PADA PEMBANGKITAN JAWA BALI II (PJB II)

(Rachmanto Junaidi, 8912097, Teknik Elektro Energi Listrik S-1, Dosen Pembimbing : Ir. Yusuf Ismail Nakhoda, MT)

Kata kunci : Komitmen Unit, *Genetic Algorithm* (GA), *Genetic Algorithm Simulated Annealing* (GASA)

Adanya persoalan operasi pembangkit tenaga listrik yaitu, bagaimana daya listrik yang dibangkitkan harus sama dengan beban yang dibutuhkan. Oleh karena itu pada suatu operasi pada beban tertentu, perhitungan ekonomis harus tetap merupakan suatu prioritas atau nilai yang harus diperhitungkan disamping hal-hal lain sehingga nantinya diperlukan suatu rencana operasi pembangkitan yang optimum dengan tetap memenuhi beberapa persyaratan pengoperasian sistem tenaga. Pada komitmen unit diasumsikan ada seperangkat N unit pembangkit yang tersedia dan adanya ramalan permintaan beban yang akan dikonsumsi, dengan demikian persoalannya adalah menentukan sub perangkat mana yang seharusnya dioperasikan agar memperoleh biaya operasi seekonomis mungkin.

Penulisan ini membahas penggunaan metode *Genetic Algorithm Simulated Annealing* (GASA) untuk menyelesaikan masalah optimasi pada Komitmen Unit, yaitu penjadwalan yang paling ekonomis pada pengoperasian unit pembangkit untuk jangka waktu tertentu dan permintaan beban tertentu pula. Metode GASA merupakan gabungan dari 2 buah metode, yaitu metode *Genetic Algorithm* (GA) dan *Simulated Annealing* (SA). GA adalah metode optimasi serbaguna berdasar pada prinsip yang diilhami dari disiplin ilmu biologi, berfungsi untuk membangkitkan populasi inisial sekaligus menghasilkan solusi turunan untuk membentuk populasi yang baru menggunakan operator-operator GA. Sedangkan SA adalah metode optimasi yang diadaptasi dari disiplin ilmu metalurgi yang berfungsi untuk menentukan diterima atau tidaknya suatu solusi sekaligus mencegah terjadinya konvergensi prematur.

Untuk simulasi aplikasi program dilakukan menggunakan data unit pembangkit termis yang terdiri dari 38 unit, yang diperoleh dari PT. PJB, yaitu dengan melakukan perhitungan terhadap permintaan beban pada hari Rabu, Sabtu dan Minggu, tanggal 10, 13 dan 14 Maret 2004, yang dianggap dapat mewakili kondisi beban yang berbeda-beda. Dari hasil simulasi didapat prosentase rata-rata penghematan biaya pembangkitan sebesar 4,52 % dan waktu proses perhitungan rata-rata sebesar 12,27 detik.

DAFTAR ISI

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman


INSTITUT TEKNOLOGI NASIONAL
ITN
MALANG

BAB I
PENDAHULUAN

1.1. Latar Belakang

Energi listrik sampai saat ini dianggap paling efisien dan praktis dibandingkan energi bentuk lain, hal ini disebabkan karena energi listrik mudah dalam pembangkitan dan pendistribusiannya dalam skala besar.

Untuk meningkatkan kualitas pelayanan tenaga listrik bagi konsumen, maka dibuatlah sistim interkoneksi. Dalam sistim interkoneksi, beberapa unit pembangkit yang berada dalam suatu wilayah tertentu dihubungkan secara paralel untuk menyuplai kebutuhan energi listrik bagi masyarakat di sekitar daerah tersebut.

Sistim interkoneksi ini sangat menguntungkan bila ditinjau dari segi ekonomis, karena dapat menghemat kapasitas pembangkitan dari masing-masing unit pembangkit. Selain itu karena setiap unit menyumbangkan dayanya ke sistem, maka akan lebih menjamin kehandalan suplai energi listrik sesuai dengan permintaan beban yang berubah-ubah. Sebagai contoh, pada saat terjadi kegagalan (*disconnections*) dari salah satu unit pembangkit, daya dapat disuplai dari pembangkit yang lain dengan cepat, sehingga tidak sampai terjadi pemadaman total.

Selanjutnya di samping struktur beban yang berubah-ubah, tenaga listrik juga merupakan suatu komoditi yang tidak bisa disimpan, maka hal ini menyebabkan perlunya suatu perhitungan yang cepat dan tepat mengenai jenis dan kapasitas pembangkit yang harus beroperasi dengan biaya yang optimal dengan tetap mempertimbangkan kualitas tenaga listrik yang akan disalurkan

1.2. Rumusan Masalah

1. Bagaimana cara menekan biaya bahan bakar dari unit-unit pembangkit agar didapat hasil yang seekonomis mungkin.
2. Bagaimana menentukan pembangkit yang harus beroperasi dan pembangkit yang dinonaktifkan untuk permintaan beban tertentu.
3. Bagaimana menentukan pembagian beban untuk masing-masing pembangkit yang beroperasi pada permintaan beban tertentu.
4. Bagaimana semua permasalahan di atas dapat diatasi dalam waktu yang relatif cepat.

1.3. Tujuan

Dengan menggunakan metode *Genetic Algorithm Simulated Annealing* (GASA) diharapkan permasalahan-permasalahan di atas dapat diatasi, berupa penjadwalan operasi unit pembangkit yang paling optimal, yaitu kombinasi unit pembangkit yang terbaik untuk melayani permintaan beban dalam jangkauan 24 jam ke depan, agar didapat biaya operasional yang ekonomis dapat ditentukan dalam waktu yang relatif cepat.

1.4. Batasan Masalah

Apabila ditinjau lebih jauh, permasalahan yang ada dalam pembahasan ini cukup luas dan kompleks. Oleh karena itu agar pembahasan lebih terarah dan mempunyai tujuan, maka perlu dilakukan pembatasan masalah sebagai berikut:

1. Unit pembangkit yang dihitung hanya pembangkit termis yang termasuk ke dalam Pembangkit Listrik Jawa - Bali II (PJB II) saja.

2. Tiap perhitungan dilakukan untuk periode satu hari (24 jam) dengan tingkat akurasi penjadwalan per jam, berdasarkan data permintaan beban pada tanggal 10, 13 dan 14 Maret 2004.
3. Tidak membahas rugi-rugi transmisi.
4. Cadangan berputar hanya sebagai kendala atau batasan.
5. Tidak membahas *combined cycle* pada Pusat Listrik Tenaga Gas Uap (PLTGU). Untuk ST (*Steam Turbin*) pada *combined cycle*, parameter diambil dari PLTGU CC 1.1.1. yang beroperasi.

1.5. Metode Penelitian

Dalam penulisan Skripsi ini dilakukan langkah-langkah sebagai berikut:

1. Tinjauan Pustaka (Studi Literatur)

   Mempelajari referensi yang berhubungan dengan materi yang akan dibahas.

2. Pengumpulan Data

   Berdasarkan data yang diperoleh di PT. PJB yang meliputi:

   a. Jumlah dan jenis pembangkit termis yang termasuk ke dalam wilayah PT. PJB.

   b. Daya maksimum dan minimum tiap pembangkit.

   c. Karakteristik fungsi biaya bahan bakar tiap pembangkit.

3. Analisis

   Dari data yang diperoleh dan ditunjang dengan literatur yang ada, maka dapat diketahui jenis dan kapasitas pembangkit yang harus beroperasi sesuai dengan jadwal untuk memenuhi permintaan beban dangan daya yang seekonomis mungkin dengan cepat.

4. Kesimpulan

Setelah diadakan penganalisaan data, akan dapat diambil kesimpulan.

BAB II

TINJAUAN PUSTAKA

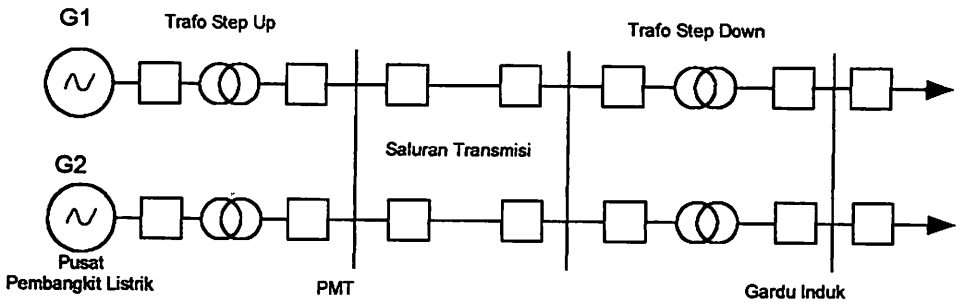
2.1. Sistem Tenaga Listrik[1]

Karena berbagai persoalan teknis, tenaga listrik hanya dapat dibangkitkan pada lokasi tertentu. Mengingat pemakai tenaga listrik atau pelanggan tenaga listrik tersebar diberbagai tempat, maka penyaluran tenaga listik dari tempat pembangkit sampai ketempat pelanggan memerlukan berbagai penanganan teknis.

Tenaga listrik dibangkitkan dipusat-pusat listrik seperti PLTA, PLTD, PLTU, PLTG, dan PLTGU, kemudian disalurkan melalui saluran transmisi setelah tegangannya dinaikkan terlebih dahulu oleh transformator penaik tegangan yang terdapat dipusat listrik.

Setelah tenaga listrik disalurkan melalui transmisi, maka sampailah tenaga listrik tersebut di Gardu Induk (GI) yang untuk kemudian tegangannya diturunkan oleh transformator penurun tegangan menjadi tegangan menengah atau tegangan distribusi primer. Tegangan distribusi primer yang dipakai PT.PLN (Pesero) adalah 20 kV, 12kV, 6kV.

Setelah disalurkan melalui jaringan distribusi primer maka tenaga listrik kemudian diturunkan tegangannya oleh gardu induk distribusi menjadi tegangan 380/220 Volt atau 220/127 Volt, dan baru kemudian disalurkan ke konsumen.



Gambar 2.1 Diagram Satu Garis Sistem Tenaga Listrik

Untuk keperluan penyediaan tenaga listrik bagi para konsumen atau beban, diperlukan berbagai peralatan listrik. Berbagai peralatan listrik ini dihubungkan satu sama lain mempunyai inter relasi dan secara keseluruhan membentuk suatu sistem tenaga listrik.

Yang dimaksud sistem tenaga listrik disini adalah sekumpulan Pusat Listrik dan Gardu Induk (Pusat Beban) yang satu sama lain dihubungkan oleh Jaringan Transmisi sehingga merupakan sebuah kesatuan interkoneksi. Biaya operasi dari Sistem Tenaga Listrik pada umumnya merupakan bagian biaya yang terbesar dari biaya operasi suatu Perusahaan Listrik. Secara garis besar biaya operasi dari suatu sistem Tenaga Listrik terdiri dari :

1. Biaya pembelian tenaga listrik.
2. Biaya pegawai.
3. Biaya bahan bakar.
4. Biaya lain-lain

Dari keempat biaya tersebut diatas, biaya bahan bakar pada umumnya adalah biaya yang terbesar. Untuk PLN biaya bahan bakar adalah kira-kira 60% dari biaya operasi secara keseluruhan.

Mengingat hal-hal tersebut diatas maka operasi Sistem Tenaga Listrik perlu dikelola atas dasar pemikiran manajemen operasi yang baik terutama karena melibatkan biaya operasi yang terbesar dan juga karena langsung menyangkut citra PLN kepada masyarakat. Manajemen Operasi Sistem Tenaga Listrik haruslah memikirkan bagaimana menyediakan tenaga listrik yang seekonomis mungkin dengan tetap memperhatikan mutu dan keandalan.

Karena daya listrik yang dibangkitkan harus selalu sama dengan daya listrik yang dibutuhkan oleh konsumen maka Manajemen Operasi Sistem Tenaga Listrik harus memperhatikan hal-hal sebagai berikut :

1. Perkiraan beban (*load forecast*).
2. Syarat-syarat pemeliharaan peralatan.
3. Keandalan yang diinginkan.
4. Alokasi beban dan produksi pembangkit yang ekonomis.

Keempat hal tersebut diatas seringkali masih harus dikaji terhadap bebarapa kendala seperti :

1. Aliran beban dalam jaringan.
2. Daya hubung singkat peralatan.
3. Penyediaan suku cadang dan dana.
4. Stabilitas Sistem Tenaga Listrik.

Dengan memperhatikan kendala-kendala ini maka seringkali harus dilakukan pengaturan kembali terhadap rencana pemeliharaan dan alokasi beban. Makin besar suatu sistem tenaga listrik makin banyak unsur yang harus dikoordinasikan serta yang harus diamati, sehingga diperlukan perencanaan, pelaksanaan, pengendalian serta analisa operasi sistem yang cermat.

2.2. Sistem Operasi Tenaga Listrik[1]

Seperti telah diketahui bahwa dalam masalah pengaturan beban pada suatu operasi sistem tenaga listrik harus selalu dicapai suatu keadaan operasi yang bisa diandalkan dan cukup ekonomis.

Ada beberapa kerja yang harus dilaksanakan untuk menjamin keandalan sistem operasi antara lain, pengaturan frekuensi dan tegangan sistem untuk berada pada harga normalnya karena adanya perubahan beban pada sistem. Dan seperti yang diketahui dan berulang kali disebutkan bahwa tenaga listrik tidak dapat disimpan sehingga dalam operasinya harus selalu dicapai keseimbangan antara penyediaan dengan pemenuhan kebutuhan daya serta perlu juga diingat bahwa sistem selalu berubah setiap saat. Maka sudah tentu jauh-jauh sebelumnya sudah harus diketahui atau diramalkan keadaan tersebut dengan tepat yaitu keadaan beban pada hari itu dari waktu ke waktu sampai selama 24 jam. Keadaan beban ini digambarkan sebagai kebutuhan daya sebagai fungsi dari waktu yang biasa disebut dengan lengkung beban harian . Lengkung beban harian ini adalah merupakan sesuatu yang sangat penting disamping karakteristik-karakteristik lainnya sehingga dalam operasi hariannya harus berdasarkan lengkung beban harian yang telah dibuat karena dengan lengkung beban harian ini dapat ditentukan perencanaan operasi pembangkit-pembangkit yang ada, baik itu unit pembangkit thermal maupun hidro. Tentu saja kebutuhan beban dalam suatu harinya tidak merata akan tetapi dari jam ke jam berbeda sesuai dengan kebutuhan konsumen. Berdasarkan lengkung beban yang telah ada maka dapat ditentukan berapa unit pembangkit yang harus bekerja dan siap bekerja pada hari itu.

Sebagai dasar pertimbangan yang sifatnya umum, untuk menentukan biaya produksi tenaga listrik yang dibutuhkan adalah dengan memperhatikan bahwa dalam

keadaan beban minimum maka tenaga listrik yang dibutuhkan diberikan oleh unit pembangkit yang bekerja paling efisien pada keadaan tersebut. Pembangkit ini akan terus beroperasi atau dibebani sampai pada batas efesiensi maksimumnya. Dan apabila ternyata beban masih terus bertambah sedangkan unit pembangkit ini telah mencapai maksimumnya maka selanjutnya beban ditanggung oleh unit pembangkit yang lain yang belum mencapai efesiensi maksimumnya. Dengan dasar operasi yang demikian maka dapat dicapai keadaan operasi yang cukup ekonomis.

Akan tetapi dengan semakin berkembangnya sistem itu sendiri maka diperlukan suatu perencanaan pembangkitan yang optimum dengan biaya operasi yang ekonomis dan harus memperhitungkan rugi-rugi yang terjadi pada saluran transmisi. Mengingat bahwa beban sistem adalah selalu berubah-ubah dari waktu ke waktu maka perlu untuk membuat secara grafis perubahan beban terhadap waktu.

Oleh karena biaya operasi untuk memproduksi daya listrik, suatu pembangkit hidro (PLTA) sangat kecil jika dibandingkan dengan pembangkit thermal (PLTU, PLTG, PLTGU, PLTD) maka pembahasan selanjutnya untuk mendapatkan biaya operasi yang ekonomis sebagian besar ditekan pada unit pembangkit thermal saja karena disini akan membutuhkan biaya operasi yang cukup tinggi sehingga usaha penghematan biaya bahan bakar akan sangat berarti. Dengan kata lain dengan mengkoordinasikan operasi pembangkit-pembangkit yang tersedia dengan tepat dan sesuai dengan beban maka akan didapat suatu keadaan operasi yang ekonomis.

Pembahasan mengenai operasi ekonomis adalah merupakan salah satu cara bagaimana menekan biaya produksi dari sistem tenaga listrik. Dalam hal ini maka metode yang dipakai adalah dengan memanfaatkan karakteristik dari menganalisa operasi dari sistem tersebut. Disamping karakteristik dari unit-unit pembangkit perlu

juga diketahui karakteristik beban, karena karakteristik bebanlah maka dapat dianalisa pengaturan yang paling ekonomis dari setiap unit pembangkit. Adapun karakteristik yang perlu diketahui dari setiap unit pembangkit adalah :

1. Karakteristik *input* bahan bakar sebagai fungsi dari *output* daya.
2. Nilai panas sebagi fungsi *output* daya.
3. Kenaikan jumlah bahan bakar yang dibutuhkan jika terdapat perubahan beban.

Ketiga karakteristik tersebut merupakan pedoman menganalisa penjadwalan selanjutnya. Kemudian yang juga perlu diperhitungkan adalah variabel-variabel yang terdapat pada saluran transmisi, karena variabel-variabel ini juga sangat menentukan ekonomis tidaknya penjadwalan pembangkit yang kita tentukan.

Maka untuk mencapai suatu operasi yang ekonomis pada suatu sistem tenaga listrik adalah dengan melakukan penjadwalan pada sistem pembangkit yang ada pada suatu sistem tenaga listrik yang ditinjau tersebut dengan memanfaatkan karakteristik dari setiap masing-masing unit pembangkit yang ada pada dasarnya bertujuan untuk menekan biaya produksi listrik agar harga dari listrik yang dihasilkan dapat ditekan serendah mungkin sehingga dapat memuaskan pemakai listrik.

2.3. Karakteristik Unit Pembangkit[1]

2.3.1. Karakteristik *Input-Output*[1]

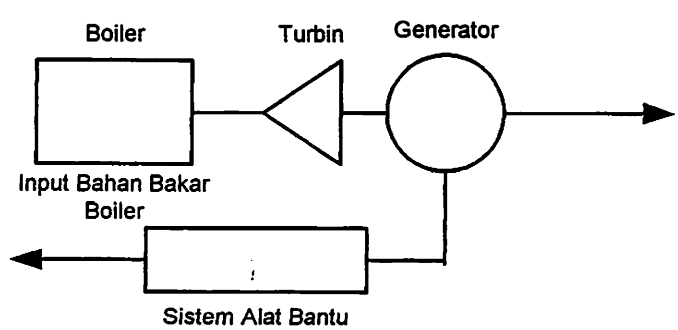
Hal yang paling mendasar dalam operasi pembangkitan yang ekonomis adalah dengan membuat karakteristik *input-output* dari unit pembangkit thermal. Karena ini diperoleh dari desain perencanaan atau melalui test pembangkit. Adapun definisi dari karakteristik *input-output* dari pembangkit itu sendiri adalah formula yang menyatakan hubungan antara *input* pembangkit sebagai fungsi dari *output*

pembangkit. Sedangkan ciri dari unit boiler-turbin-generator dapat digambarkan dalam gambar 2.2, dimana unit ini memuat sebuah boiler yang menghasilkan uap untuk menjalankan turbin yang dikopel dengan rotor dari generator.

Pada pembangkit termal input diberikan dalam satuan panas Btu/jam atau Kalori/jam dari bahan bakar yang diberikan boiler untuk menghasilkan *output* pembangkit. Sedangkan notasi yang digunakan adalah H (*MBtu/h*) atau dalam satuan yang lain H (*MKal/h*). Adapaun dalam skripsi ini, perhitungan dilakukan adalah dalam satuan *MKal/jam*. Selain itu *input* dari pembangkit dapat pula dinyatakan dalam nilai uang yang menyatakan besarnya biaya yang diperlukan untuk bahan bakar. Notasi yang digunakan adalah F (*Rp/h*). Hubungan antara H dan F dapat dinyatakan dalam rumus sebagai berikut ini :

$$F = H \times \frac{Rupiah}{MBtu} \quad \text{...}(2.1)$$

Dimana $\frac{Rupiah}{MBtu}$ adalah nilai uang yang diperlukan per satuan panas dari bahan bakar.



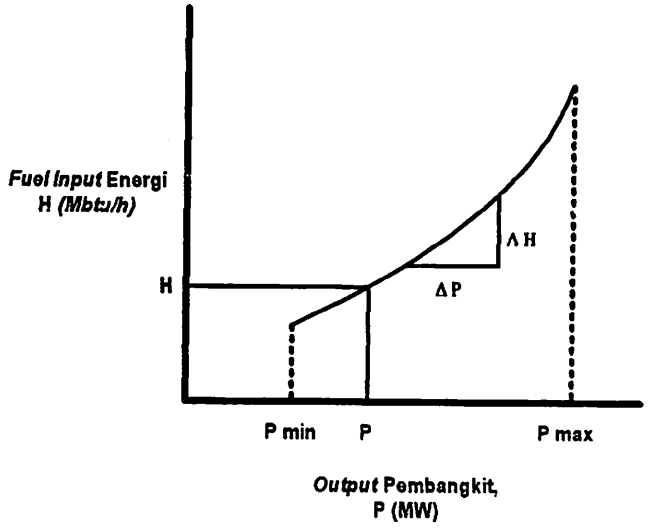
Gambar 2.2
Unit Boiler-Turbin-Generator[1]

Seperti digambarkan dalam gambar 2.2, maka *output* dari pembangkit tidak hanya dihubungkan dengan sistem saja akan tetapi juga untuk sistem peralatan bantu pembangkit didefinisikan sebagai daya yang dikeluarkan oleh generator karakteristik *input-output*, daya *output* adalah berupa daya netral dari pembangkit, notasi yang digunakan adalah P (MW).

Persamaan karakteristik *input-output* pembangkit dapat dilihat pada persamaan (2.2) dan (2.3) dibawah ini, sedangkan kurva dari karakteristik *input-output* pembangkit dapat dilihat pada gambar 2.3.

$H = f(P)$, atau ..... (2.2)

$F = f(P)$ .....(2.3)

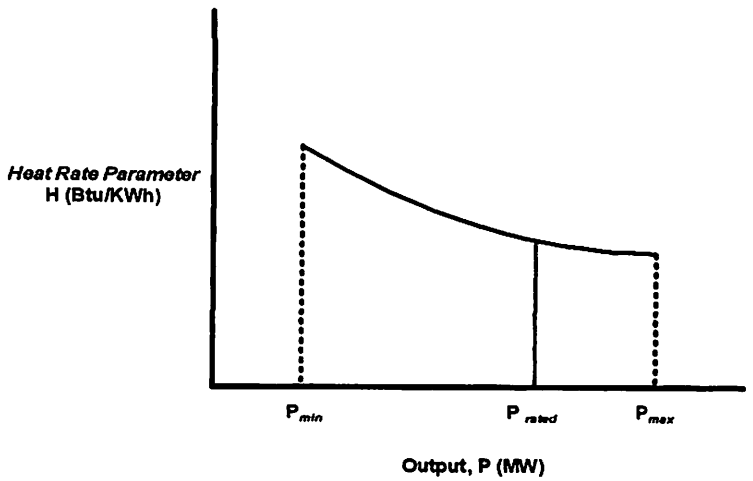


Gambar 2.3
Kurva Karakteristik *Input-Output* Pembangkit Thermal[2]

2.3.2. Karakteristik *Heat-Rate*[2]

Karakteristik *heat-rate* merupakan karakteristik yang menunjukkan efesiensi dari sebuah mesin. Karakteristik *heat-rate* sebuah unit pembangkit menunjukkan

input kalor yang diberikan untuk menghasilkan energi sebesar 1 kiloWatt jam pada MegaWatt *output* dati suatu unit. Kurva dari karakteristik *heat-rate* ini dapat dilihat pada gambar 2.4 di bawah ini.



Gambar 2.4
Kurva Karakteristik *Heat-Rate* Unit Pembangkit[2]

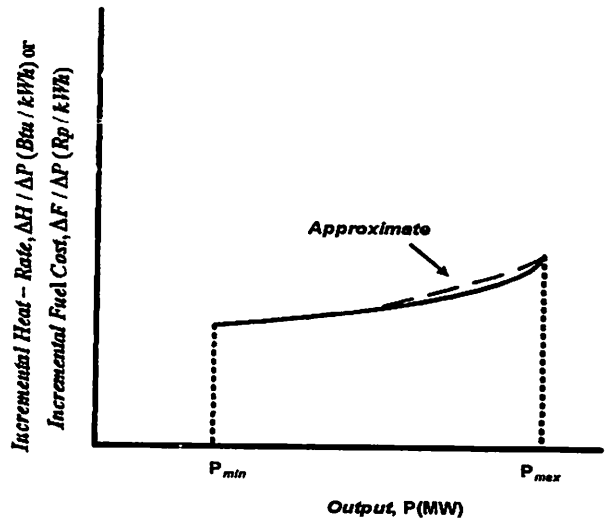
2.3.3. Karakteristik *Incremental Heat-Rate* dan *Incremental Fuel Cost*[2]

Perwujudan yang lain dari karakteristik pembangkit adalah karakteristik *Incremental Heat-Rate* atau perubahan tingkat laju panas dan karakteristik *Incremental Fuel Cost* atau perubahan tingkat laju bahan bakar. Karakteristik ini menyatakan hubungan daya keluaran pembangkit sebagai fungsi *Incremental Heat-Rate* atau *Incremental Fuel Cost*. Karakteristik *Incremental Heat-Rate* ini menunjukkan besarnya perubahan *input* energi bila ada perubahan *output* pada unit pembangkit.

Kurva dari karakteristik *Incremental Heat-Rate* atau *Incremental Fuel Cost* dapat dilihat pada gambar 2.5. Sedangkan persamaan *Incremental Heat-Rate* dan persamaan *Incremental Fuel Cost* dapat dilihat pada persamaan (2.4) hingga persamaan (2.7).

$$Incremental\ Heat\text{-}Rate = \frac{\Delta H}{\Delta P}\left(\frac{MBtu}{kWh}\right) \quad \text{.....(2.4)}$$

$$Incremental\ Fuel\ Cost = \frac{\Delta F}{\Delta P}\left(\frac{Rupiah}{kWh}\right) \quad \text{.....(2.5)}$$



Gambar 2.5
Kurva Karakteristik *Incremental Heat-Rate/Fuel Cost*[2]

Bila harga Δ sangat kecil maka dapat dinyatakan dengan persamaan berikut ini :

$$Incremental\ Heat\text{-}Rate = \frac{dH}{dP}\left(\frac{MBtu}{kWh}\right) \quad \text{.....(2.6)}$$

$$Incremental\ Fuel\ Cost = \frac{dF}{dP}\left(\frac{Rupiah}{kWh}\right) \quad \text{.....(2.7)}$$

2.4. Komitmen Unit[2]

Secara umum beban listrik dalam suatu sistem tenaga listrik selama 24 jam per hari selalu berubah berdasarkan kebutuhan masyarakat yang dapat direkam dalam interval 1 jam misalnya. Dalam kasus sistem tenaga listrik, total beban pada sebuah sistem secara umum akan lebih besar pada selang waktu tengah hari dan sore

hari, ketika beban industri dan beban rumah tangga tinggi. Begitu sebaliknya antar tengah malam hingga dini hari beban pada sistem akan lebih rendah, dikarenakan pada saat itu semua aktifitas mulai berhenti dan penduduk mulai tidur.

Untuk mengatasi permasalahan tersebut agar mendapatkan suatu keseimbangan antara beban yang dibutuhkan konsumen dengan daya keluaran dari perusahaan listrik (yang dalam hal ini sebagai produsen) diperlukan suatu penjadwalan operasi unit pembangkit, karena tidak mungkin untuk mencukupi beban puncak kurun waktu siang hari unit yang dioperasikan hanya satu dua unit pembangkit saja, untuk itu diperlukan unit-unit lain yang ikut beroperasi yang biasa disebut sebagai sistem hubungan interkoneksi. Yang pada prinsipnya adalah menggabungkan secara paralel beberapa pusat pembangkit yang berada di beberapa lokasi melalui suatu jaringan transmisi bertegangan tinggi untuk menyuplai beban gabungan (*infinite bus*).

Dengan sistem hubungan interkoneksi yang diharapkan akan menghasilkan suatu keseimbangan antara kebutuhan beban dengan daya yang dibangkitkan oleh unit-unit pembangkit, muncul sebuah masalah bagaimana mengoperasikan unit-unit pembangkitan secara optimal. Artinya bagaimana mendapatkan suatu operasi pembangkitan yang optimal agar dapat mencapai biaya operasi (biaya bahan bakar khususnya) seminimal mungkin, karena biaya bahan bakar merupakan faktor terpenting dalam biaya operasi.

Metode yang paling tepat dalam menyelesaikan permasalahan tersebut yang mempunyai tujuan untuk mendapatkan jadwal unit pembangkit yang beroperasi selama periode waktu tertentu agar optimal dengan biaya operasi yang minimum adalah Komitmen Unit. Pada Komitmen Unit diasumsikan ada seperangkat N unit

pembangkit yang tersedia dan adanya ramalan permintaan beban yang akan dikonsumsi, dengan demikian persoalannya adalah menentukan sub perangkat mana yang seharusnya dioperasikan agar memperoleh biaya operasi seekonomis mungkin.

Secara garis besar, metode yang digunakan untuk menyelesaikan masalah Komitmen Unit dapat dibagi menjadi 3 kategori [5], yaitu :

- Metode Optimasi Klasik (*Classical Optimization Method*), seperti *Dynamic Programming, Integer Programming, Branch and Bound* dan *Lagrangian Relaxation.*
- Metode Heuristic (*Heuristic Method*), seperti *Priority List.*
- Metode Kecerdasan Buatan (*Artificial Intelligence*), seperti *Neural Networks, Expert System, Genetic Algorithms, Tabu Search,* dan *Simulated Annealing.*

2.5. Fungsi Obyektif

Fungsi obyektif merupakan fungsi terpenting dalam komitmen unit, karena merupakan obyek atau tujuan dari proses perhitungan, yaitu untuk mengusahakan biaya operasional serendah-rendahnya.

a. Biaya Bahan Bakar

Bahan bakar yang diperlukan oleh unit pembangkit untuk beroperasi berbanding lurus dengan waktu operasi. Semakin lama waktu operasi, bahan bakar yang digunakan semakin banyak dan biaya yang diperlukan juga semakin besar. Demikian pula sebaliknya

b. Biaya *Start-Up*

Biaya *start-up* adalah biaya yang diperlukan pembangkit untuk *start* dari keadaan tidak beroperasi sampai pembangkit beroperasi (terhubung pada sistem tenaga listrik). Ada dua macam *start* yaitu :

1. Biaya *start* pada kondisi panas (*Hot Start*).

   Keadaan dimana unit pembangkit baru saja dimatikan dan relatif masih mempunyai temperatur mendekati temperatur operasi.

2. Biaya *start* pada kondisi dingin (*Cold Start*)

   Keadaan dimana unit pembangkit dioperasikan dari keadaan berhenti.

   Ada dua macam biaya *start* dingin, pertama biaya saat *cooling* yaitu biaya untuk memanaskan boiler unit pembangkit termal yang setelah behenti dibiarkan menjadi dingin kemudian dipanaskan kembali hingga mencapai temperatur operasi pada waktunya untuk penjadwalan penyalaan. Kedua adalah *banking* yaitu biaya untuk memanaskan boiler dari suatu temperatur tertentu yang dipertahankan dengan memanaskan boiler, sampai temperatur operasi. Kedua biaya tersebut dapat dibandingkan, sehingga pendekatan terbaik (*cooling* atau *banking*) dapat dipilih.

2.6. Kendala-kendala Unit Termis

Dalam pengoperasian unit pembangkit untuk memenuhi kebutuhan beban terdapat berbagai kendala yang merupakan syarat pembatas (*constraint*). Kendala tersebut antara lain :

a. Kendala Sistem (*System Constraints*)

Yang termasuk ke dalam kendala sistem adalah:

- Permintaan Beban (*Load Demand*)

  Besarnya daya listrik yang harus disuplai oleh keseluruhan unit pembangkit pada saat tertentu dan untuk jangka waktu tertentu.

- Cadangan Putaran (*Spinning Reserve*)

  Merupakan cadangan daya yang harus diperhitungkan dari unit-unit pembangkit yang beroperasi, dimana apabila ada salah satu unit mengalami kegagalan operasi atau keluar dari sistem, harus ada cukup cadangan daya untuk mencukupi berkurangnya suplai daya dalam periode waktu tertentu. Umumnya cadangan daya diperhitungkan untuk mampu mengganti apabila unit yang terbesar mengalami kegagalan operasi.

b. Kendala Unit (*Unit Constraints*)

Yang termasuk ke dalam kendala unit adalah :

- Batas-batas Kemampuan Pembangkit (*Generation Limit*)

  Batas maksimum dan minimum pembangkit adalah *output* daya maksimum dan minimum yang dapat disuplai oleh suatu unit pembangkit untuk memenuhi permintaan beban tertentu.

- Waktu minimum pembangkit dalam keadaan *off* atau *on* (*minimum up / down time*).

  a). *Minimum Up Time* (MUT)

  Minimum Up Time (MUT) adalah waktu minimum sebuah generator dapat dimatikan (*OFF*) setelah generator tersebut nyala (*ON*), artinya

bahwa sekali unit dinyalakan, tidak boleh langsung dimatikan harus ada tenggang waktu bekerjanya.

b). *Minimum Down Time* (MDT)

Minimum Down Time (MDT) adalah waktu minimum sebuah generator untuk dapat dinyalakan (*on*) kembali setelah generator tersebut mati(*off*). Artinya bahwa sekali unit dimatikan, ada waktu minimum sebelum unit dapat dihidupkan kembali.

- *Crew Constraint*

Jika sebuah pembangkitan terdiri dari dua unit atau lebih, unit-unit tersebut tidak dapat dinyalakan pada waktu yang bersamaan apabila operator yang ada tidak mencukupi.

BAB III
KOMITMEN UNIT MENGGUNAKAN METODE *GENETIC ALGORITHM SIMULATED ANNEALING* (GASA)

3.1. Fungsi Obyektif (*Objective Function*)

Komponen utama dari biaya operasional pada pembangkit termis adalah biaya produksi daya listrik dari unit-unit yang terkait sesuai dengan persamaan :

$$F_{it}(P_{it}) = A_i P^2_{it} + B_i P_{it} + C_i \quad ; \quad \$ / HR \qquad \ldots\ldots(3.1)$$

dimana:

$F_{it}(P_{it})$ = Biaya produksi dari unit i pada waktu t ($/HR)

P_{it} = Daya keluaran dari unit i pada waktu t (MW)

A_i, B_i, C_i = Parameter fungsi biaya dari unit i

Biaya *Start-up* tergantung dari waktu "OFF" unit yang dapat bervariasi dari nilai maksimum yaitu ketika unit pembangkit dihidupkan mulai dari keadaan dingin (*Cold State*) hingga nilai yang lebih kecil yaitu ketika unit pembangkit dihidupkan dari keadaan hangat (*Warm State*). Dalam hal ini biaya *Start-up* untuk unit i pada waktu t dirumuskan dalam bentuk persamaan :

$$S(t) = \begin{cases} S_h & if -x(t) \le t_{cold\ start} \\ S_c & otherwise \end{cases} \qquad \ldots\ldots(3.2)$$

dimana:

S(t) = Biaya *start up*

Sh = Biaya *hot start up*

Sc = Biaya *cold start up*

$t_{cold\ start}$ = Batas waktu bagi *boiler* untuk melakukan *hot start.*

maka fungsi obyektif keseluruhan dari Komitmen Unit adalah :

$$F_T = \sum_{t=1}^{T}\sum_{i=1}^{N}(U_{it}F_{it}(P_{it})+V_{it}S_{it}) \quad ; \quad \$ \ldots\ldots (3.3)$$

dimana:

F_T = Biaya operasional keseluruhan selama jangka waktu tertentu ($)

U_{it} = Status unit i pada jam t (1 jika unit dalam keadaan ON dan 0 jika unit daiam keadaan OFF)

V_{it} = Status *Start-up / Shut-down* dari unit i pada jam t (1 jika unit di start pada jam t dan 0 jika tidak)

3.2. Kendala-kendala (*Constraints*)

Dalam menyelesaikan masalah Komitmen Unit, beberapa kendala mesti dipenuhi. Kendala yang perlu diperhitungkan dalam hal ini dapat dibagi menjadi dua kelompok utama yaitu:

1. Kendala Sistem (*System Constraints*)

- Kendala Permintaan Beban (*Load Demand Constraints*)

  Daya yang dibangkitkan dari seluruh unit pembangkit yang termasuk dalam Komitmen Unit harus memenuhi persamaan :

$$\sum_{i=1}^{N}U_{it}P_{it}=PD_t \quad ; 1 \leq t \leq T \ldots\ldots (3.4)$$

  dimana :

  PD_t = Permintaan beban sistem pada jam t (MW)

- Kendala Cadangan Putaran (*Spinning Reserve Constraints*)

Spinning Reserve adalah jumlah total dari daya yang dibangkitkan oleh pembangkit yang beroperasi dikurangi dengan permintaan beban saat itu.

$$\sum_{i=1}^{N} U_{it} P_{max\,i} \geq (PD_t + R_t) \quad ; \quad 1 \leq t \leq T \qquad \text{.....(3.5)}$$

dimana :

$P_{max\,i}$ = Batas maksimum pembangkitan oleh unit i (MW)

R_t = Cadangan sistem pada jam t (MW)

2. Kendala Unit (*Unit Constraints*)

- Batas-batas Pembangkitan (*Generation Limit*)

$$U_{it} P_{min\,i} \leq P_{it} \leq P_{max\,i} U_{it} \quad ; \; 1 \leq t \leq T, 1 \leq i \leq N \qquad \text{.....(3.6)}$$

dimana:

$P_{min\,i}$ = Batas minimum pembangkitan oleh unit i (MW)

- Waktu minimum pembangkit dalam keadaan *off* atau *on* (*minimum up down time*)

$$T_{off\,i} \geq T_{down}\,I \qquad \text{.....(3.7)}$$

$$T_{on\,i} \geq T_{up\,i} \quad ; \; 1 \leq i \leq N \qquad \text{..... (3.8)}$$

dimana:

$T_{off\,i}$ = Selang waktu dimana unit i dalam keadaan OFF

$T_{down\,i}$ = *Down time* minimum dari unit i

$T_{on\,i}$ = Selang waktu dimana unit i dalam keadaan ON

$T_{up\,i}$ = *Up time* minimum dari unit i

3.3. Metode *Genetic Algorithm Simulated Annealing* (GASA)

Teknik kecerdasan buatan menjadi suatu cara yang digunakan secara luas dalam menyelesaikan masalah optimasi, dalam hal ini metode GASA yang merupakan penggabungan 2 buah metode yang terdiri atas *Genetic Algorithm* (GA) dan *Simulated Annealing* (SA) yang digunakan untuk meyelesaikan masalah komitmen unit.

Dalam penerapan metode *Genetic Algorithm Simulated Annealing* (GASA) pada masalah *commitment unit,* digunakan suatu bentuk perhitungan untuk memudahkan dalam menentukan suatu jadwal pembangkit dalam horizon perencanaan, kemudian tiap-tiap unit dapat dikombinasi untuk mencapai biaya operasional yang paling ekonomis namun tetap memenuhi batasan sistem.

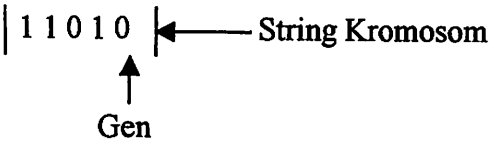
3.3.1. Genetic Algorithm (GA)[3]

Genetic Algorithm (GA) merupakan *general-purpose* yaitu teknik pencarian yang didasarkan atas prinsip penalaran dari asas keturunan dan pengamatan mekanisme pada evolusi dalam sistem alami populasi kehidupan pada makluk hidup.

Genetic Algorithm (GA) adalah suatu algoritma untuk memecahkan masalah optimasi parameter. Prinsip yang mendasari GA pertama kali dikembangkan oleh John Holland pada tahun 1962. Teori GA didasari oleh teori evolusi Darwin. Landasan GA 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.

GA bekerja dengan populasi string, dan melakukan proses pencarian nilai optimal secara paralel. Dengan menggunakan operator genetika. GA akan melakukan rekombinasi antar individu. Elemen dasar yang diproses GA 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 string kromosom bit biner ditunjukkan oleh gambar 3.1.



Gambar 3.1
String Kromosom Bit Biner

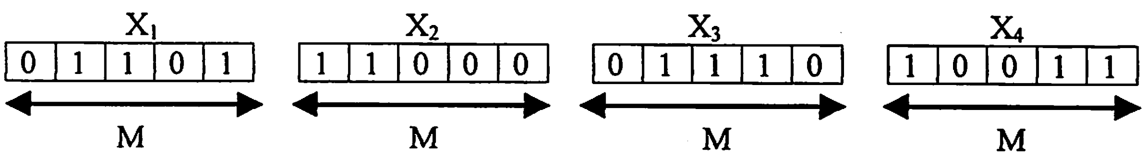
GA merupakan prosedur iteratif, bekerja dengan suatu kumpulan string sebagai kandidat solusi dengan jumlah konstan. Populasi ini kemudian berkembang dari generasi ke generasi melalui operator genetika. Setiap langkah iterasi disebut generasi, individu dalam populasi saat itu akan dievaluasi dan diseleksi untuk menentukan populasi pada generasi selanjutnya.

Selama proses evolusi genetika, kromosom yang lebih sehat memiliki kecenderungan menghasilkan *off spring* (keturunan) yang sehat pula, dan kromosom yang sehat diharapkan menghasilkan keturunan yang lebih banyak serta mungkin dapat bertahan pada generasi selanjutnya. Pada evolusi genetika terdapat beberapa konsep dasar evolusi genetika, antara lain :

- "*Fitness*/kesesuaian/kecocokan" adalah elemen yang mengatur kelebihan suatu individu yang akan mempengaruhi generasi berikutnya.
- "Operator genetika/pengatur genetika" yang akan mengatur genetika keturunan.

3.3.1.1. Definisi-definisi Pada *Genetic Algorithm*

Individu



Gambar 3.2.

Susunan Kromosom Dalam GA

Gen : Sederetan nilai biner (0,1) pada suatu kromosom.

Pada gambar di atas jumlah Gen = M = 5 buah.

Kromosom : Susunan dari deretan gen biner.

Pada gambar di atas kromosom = $X_1/X_2/ \ldots X_n$.

Individu : Susunan dari deretan kromosom.

Pada gambar di atas individu = $(X_1, X_2, X_3, \ldots X_n)$.

Jadi pada individu di atas akan mempunyai : 5 X n gen biner.

3.3.1.2. Fungsi *Fitness*

Fungsi *fitness* meliputi kemampuan untuk membandingkan solusi dari satu generasi ke generasi yang lain. Nilai fungsi *fitness* [F(x)] berbanding lurus dengan kuadrat string kromosom (X^2) dan dirumuskan sebagai berikut :

$$F(x) = x^2 \qquad (3.9)$$

Persamaan di atas dapat mengurangi efek genetika yang menyimpang dan akan menghasilkan kromosom yang sehat. Penggambaran nilai *fitness* ditunjukkan oleh tabel di bawah ini.

Tabel 3.1
String Dan Nilai *Fitness*

No.	String	*Fitness*	% dari jumlah
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Jumlah		1170	100.0

Nilai fungsi *fitness* tidak dapat langsung dihubungkan dengan nilai tujuannya melainkan harus dirangking terlebih dahulu nilai tujuannya. Dengan cara ini, dapat ditentukan kromosom-kromosom mana yang layak digunakan dalam proses selanjutnya sehingga konvergensi awal dapat dihindari dan akan mempercepat penelitian ketika populasi mendekati konvergen.

3.3.1.3. Operator Genetika

GA merupakan metode yang memperlakukan kromosom atau populasi (string yang akan menampilkan calon solusi dari suatu masalah) ke populasi yang baru dengan menggunakan seleksi dan operator. Operator genetika yang digunakan adalah reproduksi, *crossover*, dan mutasi yang masing-masing menggunakan proses probabilitas dalam pemilihan dan pengoperasian.

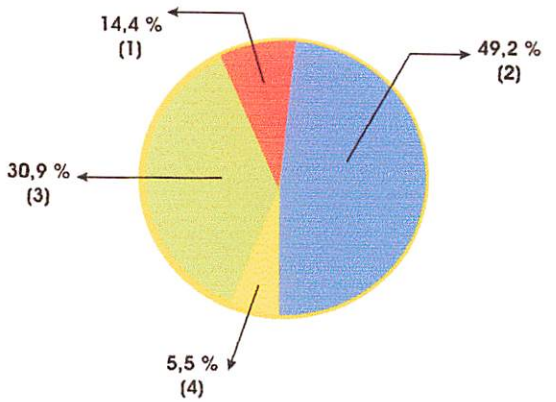
3.3.1.4. Reproduksi

Reproduksi merupakan suatu proses dimana struktur string kromosom disalin dari generasi ke generasi yang lain sesuai dengan nilai *fitness*. Tiruan string yang sesuai dengan nilai *fitness* atau lebih tinggi memiliki kemungkinan yang lebih besar atas penambahan keturunan pada generasi berikutnya. Operator ini merupakan salah satu

jenis seleksi alami buatan, yang penyelesaiannya Menggunakan "*Roulette Wheel*/roda roulette".

Prosedur penyeleksian dengan Menggunakan *Roulette Wheel* :

- Jumlah populasi yang sehat/*fitness* disebut total *fitness* (F_{sum})
- Menghasilkan suatu jumlah yang acak/random (n) antara 0 dan total *fitness* (F_{sum})
- Melakukan proses pengurutan (*ranking*) terhadap hasil pengacakan.
- Melakukan *mapping fitness* dan mencari nilai *fitness*
- Mencari *range* dari masing-masing kromosom.
- Menentukan *parents* sebagai penghasil keturunan.



Gambar 3.3
Roulette Wheel Selection

Berdasarkan contoh gambar di atas, garis tengah dari *roulette wheel* adalah F_{sum} dari ke empat kromosom. Kromosom nomor 2 adalah yang terlihat dan berada pada interval yang terbesar dan sebaliknya kromosom nomor 4 adalah yang terjelek dan ada pada interval yang terkecil dalam *roulette wheel*, dari penyelesaian di atas string yang

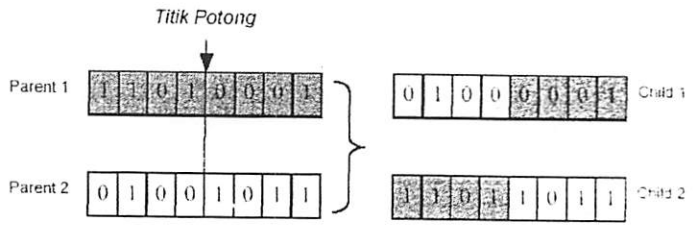
paling sehat dimasukkan ke dalam suatu "*Mapping Pool*/tempat persemaian". String kromosom ini sifatnya sementara, karena akan dilakukan operasi genetika selanjutnya.

3.3.1.5. *Crossover* (Pindah Silang)

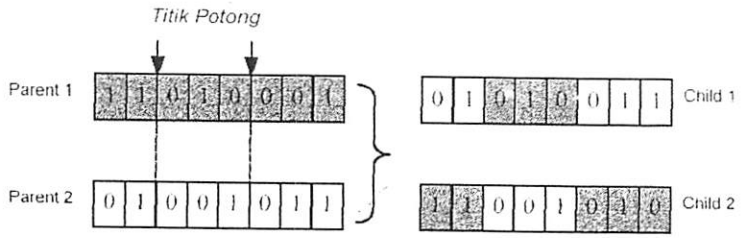
Crossover adalah operator genetika yang utama. Operator ini bekerja dengan mengambil 2 individu dan memotong string kromosom mereka pada posisi yang terpiliih secara random. Sebagai contoh adalah jika kita mengambil induk yang dipresentasikan dengan 5 dimensi vektor (a_1,b_1,c_1,d_1,e_1) dan (a_2,b_2,c_2,d_2,e_2) kemudian dilakukan *crossing* pada posisi ketiga kromosom-kromosomnya sehingga didapat keturunan (a_1,b_1,c_2,d_2,e_2) dan (a_2,b_2,c_1,d_1,e_1). Gambar 3.4 merupakan ilustrasi dari operasi *crossover*.

Crossover menghasilkan titik baru dalam ruang pencarian yang siap untuk diuji. Operasi ini tidak selalu dilakukan pada semua individu yang ada. Individu dipilih secara random untuk dilakukan secara *crossing* dengan P_c antara 0,6 sampai dengan 1,0. Jika *Crossover* tidak dilakukan, maka nilai dari induk akan diturunkan kepada keturunan.

- *Crossover* Satu Titik



- *Crossover* Dua Titik

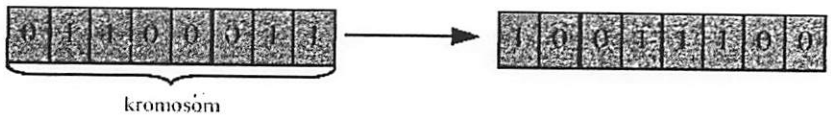


Gambar 3.4
Ilustrasi Operasi *Crossover* dalam Algoritma Genetika

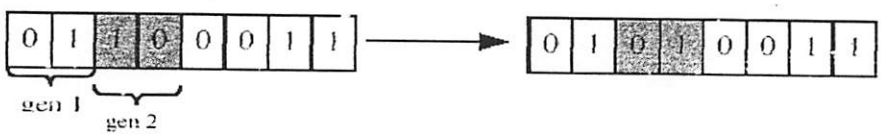
3.3.1.6. Mutasi

Operator mutasi digunakan untuk melakukan modifikasi satu atau lebih nilai gen dalam individu yang sama. Mutasi memastikan bahwa probabilitas untuk pencarian pada daerah tertentu dalm persoalan tidak akan pernah nol dan mencegah kehilangan total materi genetika setelah pemilihan dan penghapusan. Mutasi ini bukanlah operator genetika yang utama, yang dilakukan secara random pada gen dengan kemungkinan yang kecil (Pm sekitar 0,001). Ilustrasi mengenai cara kerja operator ini digambarkan pada gambar 3.5

- Semua Gen Dalam Kromosom Berubah



- Semua Bit Dalam Satu Gen Akan Berubah (misal: gen 2 yang mengalami mutasi)



- Hanya Satu Bit Yang Berubah

Gambar 3.5
Ilustrasi Operasi Mutasi Dalam *Genetic Algorithm*

Pada lazimnya, GA mempunyai tiga tahap pencarian yang berbeda, meliputi :

1. Menciptakan suatu populasi awal (*initial population*)

Populasi awal sangat berpengaruh terhadap kinerja algoritma genetik. Jika populasi awal tidak bervariasi, kemungkinan untuk mencapai kondisi jawaban yang konvergen tetapi tidak optimal, cukup besar.

Cara yang digunakan untuk menghasilkan populasi awal adalah dengan cara menciptakan bilangan acak yang unik sehingga dalam satu kromosom tidak terdapat bilangan acak yang sama. Pada saat populasi awal diciptakan, nilai *fitness* dari masing-masing individu langsung dapat dihitung.

2. Mengevaluasi fungsi *fitness*

Dengan fungsi *fitness*, dapat diketahui seberapa baik solusi yang diperoleh satu individu. Fungsi *fitness* harus sesuai dengan dengan permasalahan yang hendak diselesaikan, degan fungsi *fitness* dapat dibedakan antara kromosom yang berkualitas baik dan kromosom yang berkualitas buruk. Kromosom yang berkualitas baik mempunyai kemungkinan yang lebih besar untuk dipilih sebagai induk.

3. Memproduksi suatu populasi baru

Reproduksi merupakan suatu prosedur untuk menghasilkan individu baru yang sama dengan induknya. Pada operasi tersebut, semua karakter individu induk disalin tanpa ada perubahan apa pun sehingga individu baru itu akan memiliki nilai *fitness* yang sama persis dengan induknya.

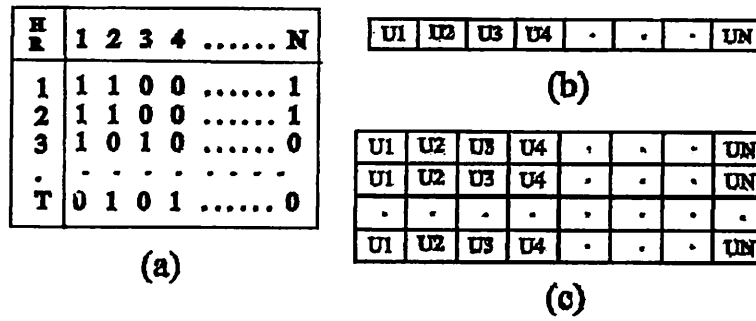
3.3.1.7. Implementasi GA Pada Algoritma GASA

Secara lengkap implementasi komponen GA dalam metode GASA diuraikan sebagai berikut:

1. Solusi Pengkodean

Pemecahan masalah dalam *unit commitment* diwakili oleh matrik biner (U) pada dimensi TxN (Gambar 3.6.a) Pengkodean disini merupakan campuran antara nomor biner dan desimal. Masing-masing vektor kolom dalam solusi matrik (Yang menunjukkan penjadwalan operasi pada satu unit) dengan panjang T dikonversikan sistem perbandingan angka desimal. Matrik solusi kemudian dikonversikan menjadi

vektor satu baris (*Cromossome*) angka desimal N (U_1, U_2, U_3.....U_N), masing-masing menunjukkan penjadwalan satu unit, secara khas U_1, U_2, U_3.....U_N adalah bilangan bulat yang berkisar antara 0 dan 2^N-1, maka sebuah populasi ukuran NPOP disimpan dalam matrik NPOP x N (gambar 3.6.b)



Gambar 3.6 a Matrik U biner
b Ekuivalen vektor desimal (satu *chromosom*)
c. Populasi pada NPOP *chromosom*

2. Fungsi *Fitness*

Fungsi *fitness* meliputi kemampuan untuk membandingkan solusi dari satu generasi ke generasi yang lain. Disini fungsi *fitness* diambil sebagai timbal balik pada total biaya operasi, sejak kita membangkitkan generasi dijalankan.

3. *Crossover*

Untuk mendapatkan perhitungan kecepatan penuh, pengoperasian *crossover* dikerjakan antara dua cromossom dalam batas nilai desimal. Dua *parent* diseleksi dan dua posisi dalam dua kromosom diseleksi secara acak. Nomor desimal dirubah antara dua parent untuk mendapatkan dua anak. Dua anak kemudian dikodekan melalui persamaan binner mereka dan dicek seberapa batas pelanggarannya. Jika batas tidak terpenuhi maka suatu mekanisme perbaikan diberlakukan untuk mengembalikan jalan untuk memproduksi anak.

4. Mutasi

Pengoperasian mutasi dilakukan secara acak untuk menyeleksi dan memilih beberapa *cromossom* dengan suatu kemungkinan . Kromosom yang dipilih kemudan dikodekan melalui pemecahan persamaan biner. Suatu nomor unit dan periode waktu secara acak diseleksi, kemudian aturan diberlakukan dalam status pembalikan pada unit ini, guna menjaga kemungkinan layaknya batasan unit berhubungan dengan waktu minimum *up-time dan minimum down- time.* Melakukan pengecekan dalam perubahan periode, dan jika perlu mengoreksi untuk pembuatan batasan cadangan kemudian.

5. Generasi

Aturan ini dibuat untuk mencapai batas minimum *up-time/down-time* dalam menyelesaikan komitmen unit, sedangkan batasan baliknya dicek dan dikoreksi jika perlu menggunakan mekanisme perbaikan.

6. Mekanisme perbaikan.

Dalam kaitanya dengan *crossover* dan pengoperasian mutasi, batas kebaikan boleh untuk dilanggar. Mekanisme perbaikan untuk mengembalikan kemungkinan pembatasan itu, diterapkan dan diuraikan sebagai berikut:

- Mengambil secara acak satu pada unit *OFF* dan satu pada jam pelanggaran
- Menerapkan aturan kemungkinan generasi untuk menghubungkan unit yang dipilih dari *OFF* sampai *ON* dengan menjaga kemungkinan waktu penurunan batasan.
- Melakukan pengecekan untuk batasan cadangan pada jam ini. Jika mencukupi, menuju ke jam yang lain. Cara lainnya, mengulangi proses pada waktu jam yang sama untuk unit yang lain.

3.3.2. Algoritma *Simulated Annealing*[5]

Simulated Annealing, dikenalkan oleh Kirkpatrick, Gela dan Vecchi pada tahun 1982;1983 dan Cerny di tahun 1985. *Annealing* secara fisika, mengacu pada proses dalam memanaskan suatu benda padat (logam) dengan temperatur yang tinggi kemudian didinginkan pelan-pelan dengan menurunkan temperatur di sekelilingnya. Pada masing-masing langkah, temperatur dijaga konstan dalam periode waktu yang cukup agar logam mencapai keseimbangan termal. Pada keadaan seimbang logam mempunyai banyak konfigurasi, masing-masing dengan rotasi elektron yang berbeda dan tingkat energi tertentu.

Dengan membuat suatu analogi antara proses annealing dan masalah optimasi, sejumlah besar permasalahan optimasi kombinasi dapat dipecahkan mengikuti prosedur peralihan yang sama dari status keseimbangan yang satu ke yang lain untuk mencapai energi yang minimum pada sistem. Analogi ini dapat dinyatakan sebagai berikut:

a. Solusi dalam masalah optimasi kombinasi adalah ekuivalen dengan status (konfigurasi) pada sistem fisika.
b. Solusi biaya adalah ekuivalen dengan status dari energi.
c. Suatu parameter control, Cp, diperkenalkan untuk memainkan peran temperatur pada proses annealing.

Dalam menerapkan Algoritma *Simulated Anealing*, untuk memecahkan masalah optimasi kombinasi, pikiran dasarnya adalah memilih suatu solusi yang mungkin secara acak kemudian mencari yang berdekatan atau tetangga dari solusi ini. Suatu langkah ke tetangga ini dilakukan jika mempunyai nilai objektif yang lebih baik (rendah) atau dalam kasus tetangga dengan nilai fungsi objektif lebih tinggi, jika $\exp(-\Delta E/Cp) \geq U(0,1)$, dimana ΔE adalah peningkatan di dalam nilai fungsi objektif. Pengaruh

mengurangi Cp adalah kemungkinan diterimanya suatu peningkatan dalam nilai fungsi objektif menjadi berkurang sepanjang pencarian.

Bagian yang paling utama dalam *Simulated Annealing* adalah mempunyai suatu aturan yang bagus untuk bermacam temuan dan meningkatkan pendekatan sehingga sejumlah besar ruang solusi dapat diselidiki. Bagian penting yang lain bagaimana cara memilih nilai inisial Cp untuk memilih dan bagaimana Cp perlu dikurangi selama pencarian.

3.3.2.1. Implementasi SA Dalam Algoritma GASA

Tahap-tahap dalam algoritma SA diterapkan sebagai berikut:

Suatu Algoritma Simulated Aneling Secara Umum dapat dijelaskan sebagai berikut:

Langkah (0) : Awali dengan hitungan iterasi k=0, temperatur $Cp=Cp_0$, harus cukup tinggi agar probabilitas sembarang solusi yang diterima mendekati 1.

Langkah (1) : Tetapkan suatu kemungkinan solusi awal = solusi terkini, X_i dengan nilai fungsi objectif terkait E_i.

Langkah (2) : Jika kondisi keseimbangan terpenuhi pergi ke langkah (5), jika tidak laksanakan langkah (3) dan (4).

Langkah (3) : Buatlah suatu solusi percobaan X_j sebagai tetangga X_i. Umpamakan E_j nilai fungsi objektif dari solusi percobaan.

Langkah (4) : Uji penerimaan: jika $E_j \geq E_i$ solusi percobaan diterima, tetapkan $X_i = X_j$ dan pergi ke langkah (2). Jika tidak, apabila $\exp [(E_i-E_j)/Cp] \geq U(0,1)$ tetapkan $X_i = X_j$ dan pergi ke langkah (2).

Jika tidak pergi ke langkah (2).

Langkah (5) : Jika kriteria penghentian terpenuhi maka berhentilah, jika tidak turunkan temperatur Cp^k dan pergi ke langkah (2).

3.4. Uji Validasi Program

Program yang digunakan telah mengalami uji validasi berdasarkan jurnal "*A Genetic Algorithm Solution To The Unit Commitment Problem*", IEEE Trans. On Power Systems, Vol.11, No.1, Feb. 1996.

Agar uji validasi bisa dilakukan, program telah disesuaikan sehingga memiliki kemampuan untuk menampilkan hasil perhitungan menggunakan metode GA, disamping metode GASA. Dengan kemampuan ini dimungkinkan untuk menampilkan hasil perhitungan dari kedua metode tersebut untuk dibandingkan hasilnya dengan biaya operasional PT. PJB.

Berdasarkan uji validasi yang telah dilakukan, program yang digunakan memiliki relatif *error* sebesar 0,0000053.

Adapun data yang terdapat dalam jurnal adalah sebagai berikut :

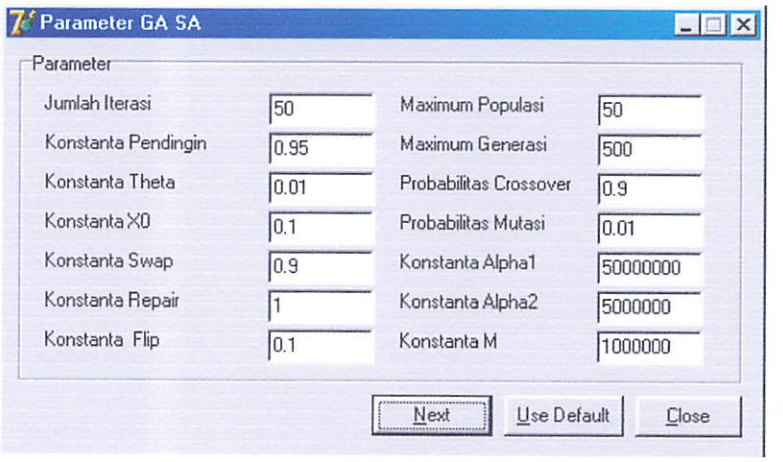
Tabel 3.2.
Data Pembangkit Untuk Uji Validasi

No	Nama	Pmax MW	Pmin MW	a0	a1	a2	Tup Hour	Tdown Hour	Sh	Sc	Tcold Start	Initial State
1	Unit 1	455	150	1000	16.19	0.00048	8	8	4500	9000	5	8
2	Unit 2	455	150	970	17.27	0.00031	8	8	5000	10000	5	8
3	Unit 3	130	20	700	16.6	0.002	5	5	550	1100	4	-5
4	Unit 4	130	20	680	16.5	0.00211	5	5	560	1120	4	-5
5	Unit 5	162	25	450	19.7	0.00398	6	6	900	1800	4	-6
6	Unit 6	80	20	370	22.26	0.00712	3	3	170	340	2	-3
7	Unit 7	85	25	480	27.74	0.00079	3	3	260	520	2	-3
8	Unit 8	55	10	660	25.92	0.00413	1	1	30	60	0	-1
9	Unit 9	55	10	665	27.27	0.00222	1	1	30	60	0	-1
10	Unit 10	55	10	670	27.79	0.00173	1	1	30	60	0	-1

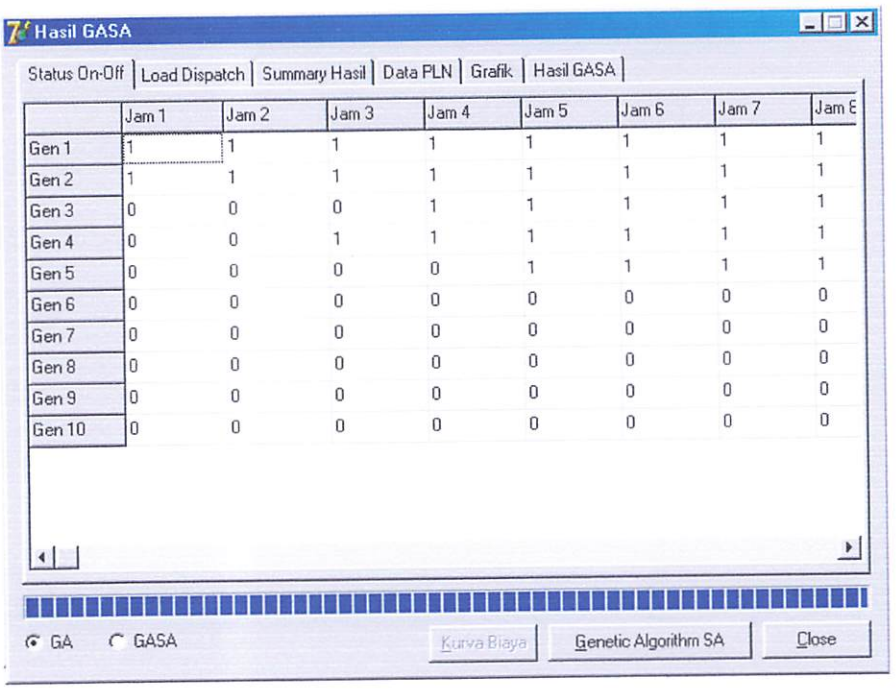
Tabel 3.3.
Data Pembebanan Untuk Uji Validasi

Hour	Jam 1	Jam 2	Jam 3	Jam 4	Jam 5	Jam 6	Jam 7	Jam 8	Jam 9	Jam 10	Jam 11	Jam 12
Load MW	700	750	850	950	1000	1100	1150	1200	1300	1400	1450	1500
Reserve MW	70	75	85	95	100	110	115	120	130	140	145	150
Hour	Jam 13	Jam 14	Jam 15	Jam 16	Jam 17	Jam 18	Jam 19	Jam 20	Jam 21	Jam 22	Jam 23	Jam 24
Load MW	1400	1300	1200	1050	1000	1100	1200	1400	1300	1100	900	800
Reserve MW	140	130	120	105	100	110	120	140	130	110	90	80

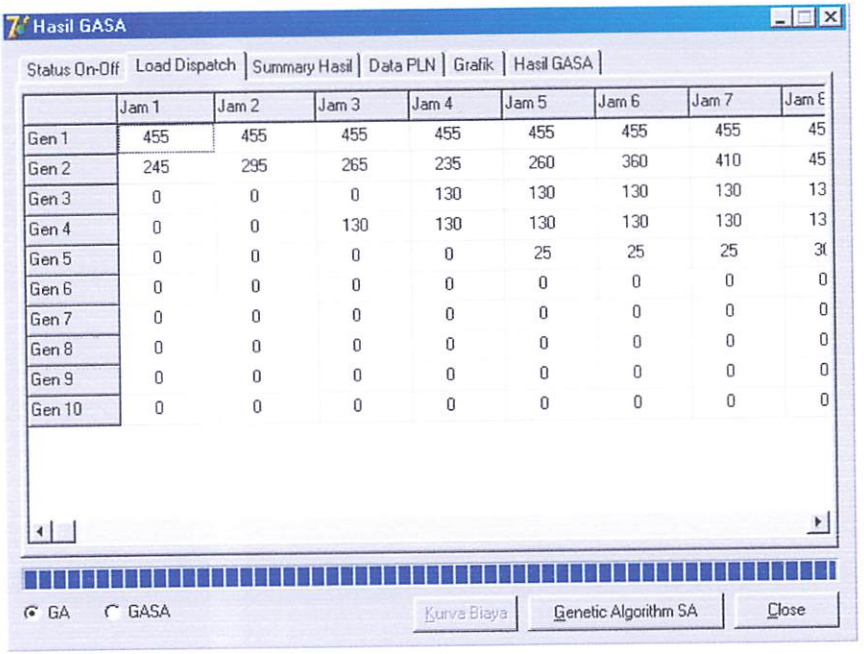
Data yang terdapat pada tabel 3.2 dan 3.3, jika dilakukan eksekusi program optimasi penjadwalan dengan menggunakan metode GA dari program adalah sebagai berikut:



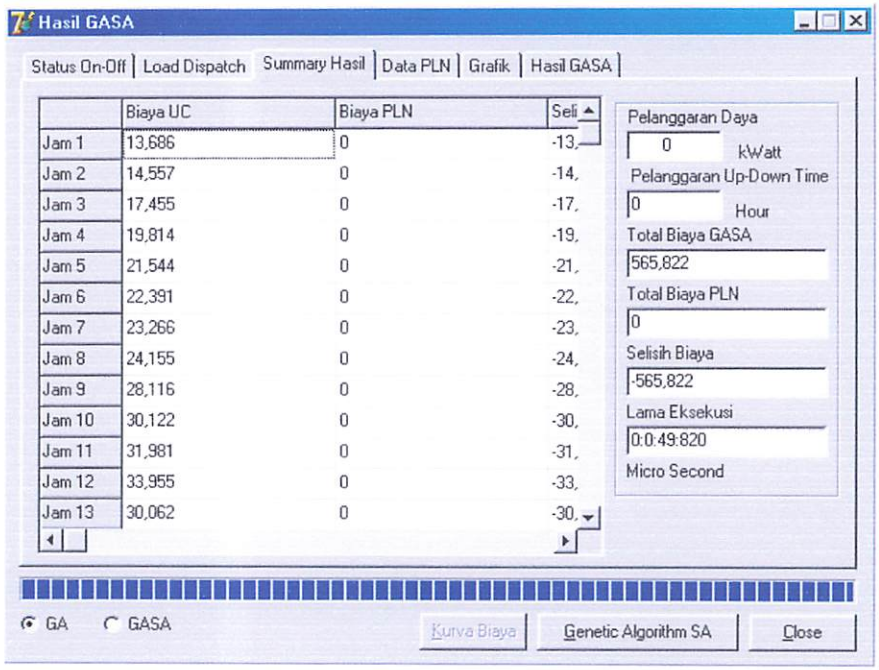
Gambar 3.7. Parameter Untuk Uji Validasi



Gambar 3.8. Status o*n/off* Hasil Uji Validasi



Gambar 3.9. *Load Dispatch* Hasil Uji Validasi



Gambar 3.10. *Summary Hasil* Uji Validasi

Tabel 3.4.
Rekapitulasi Data Hasil Uji Validasi

Jumlah Unit Pembangkit	Hasil GA (Jurnal)	Hasil GA (Program)	Error
10	565825	565822	0,0000053

3.5. Program Komputer Metode GASA

Untuk pemecahan masalah unit komitmen digunakan bantuan program komputer. Program komputer ini sangat berguna untuk mempercepat proses perhitungan yang membutuhkan ketelitian tinggi dan sering melibatkan iterasi yang membutuhkan waktu yang lama bila dikerjakan secara manual. Program komputer ini dibuat menggunakan bahasa pemrograman *Borland Delphi* versi 7.

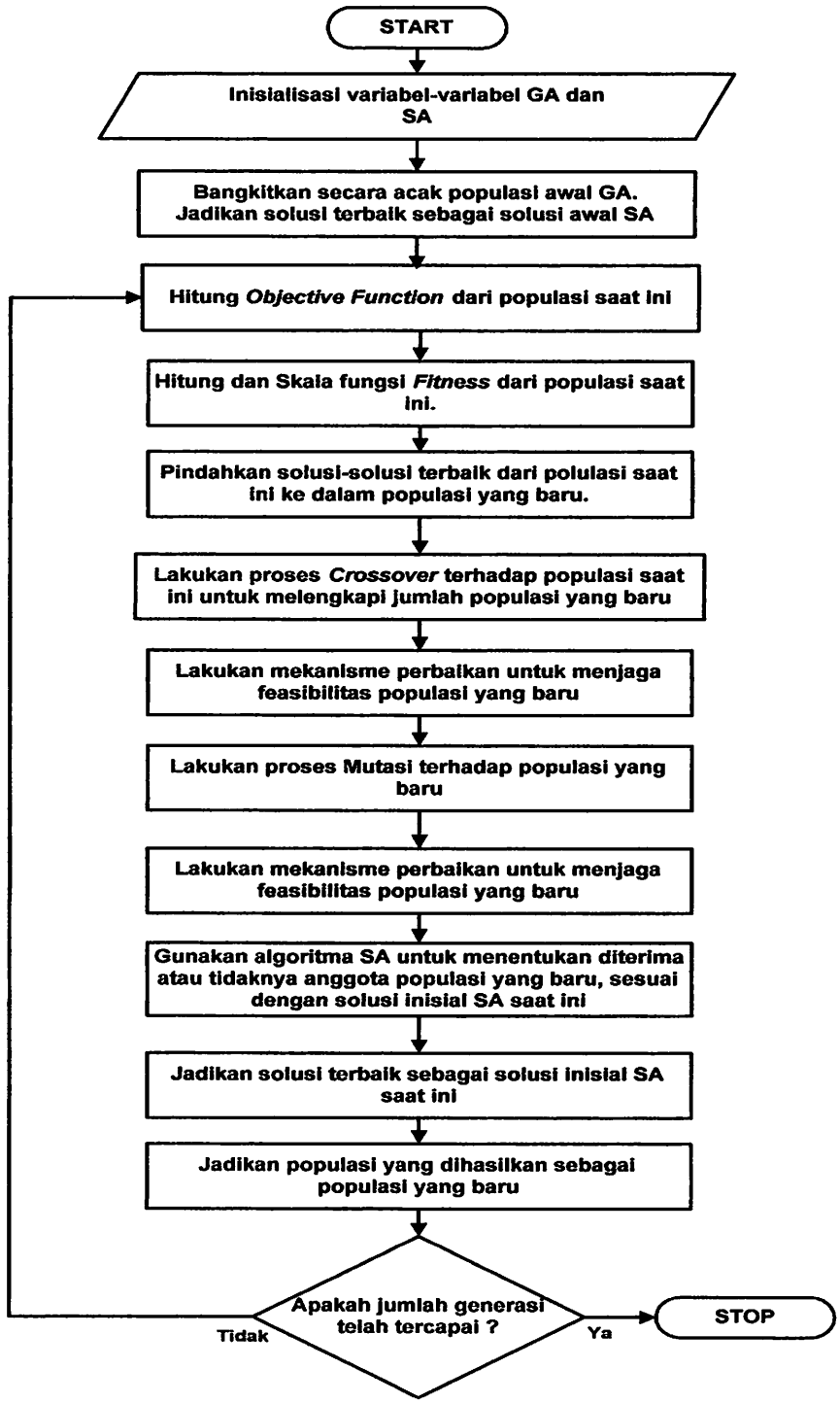
3.6. Algoritma Program

Urutan langkah-langkah dalam program komputer yang digunakan ini dapat dilihat pada algoritma program berikut :

1. Memasukkan data parameter unit pembangkit termal dan data pembebanan harian untuk periode waktu 24 jam

   Data tiap-tiap unit pembangkit termal yang diperlukan adalah jumlah unit pembangkit, daya maksimum dan daya minimum, konstanta persamaan biaya bahan bakar , harga bahan bakar, biaya start-up, minimum up time dan down time
2. Memasukkan parameter GA, dan SA
3. Menentukan generasi awal (gen=0)*
4. Menentukan populasi awal
5. Menghitung nilai *fitness* populasi
6. Apakah jumlah generasi sudah terpenuhi (gen ≥ gen max)
7. Jika "ya" perhitungan selesai
8. Jika "tidak" copy member terbaik pada parent
9. Menerapkan operator *crossover* untuk melengkapi anggota pada populasi baru.
10. Menerapkan operator mutasi untuk populasi baru.
11. Menerapkan algoritma SA untuk menguji anggota pada populasi baru
12. Menambahkan generasi baru , kembali menuju ke langkah 6.

3.7. Diagram Alir Program

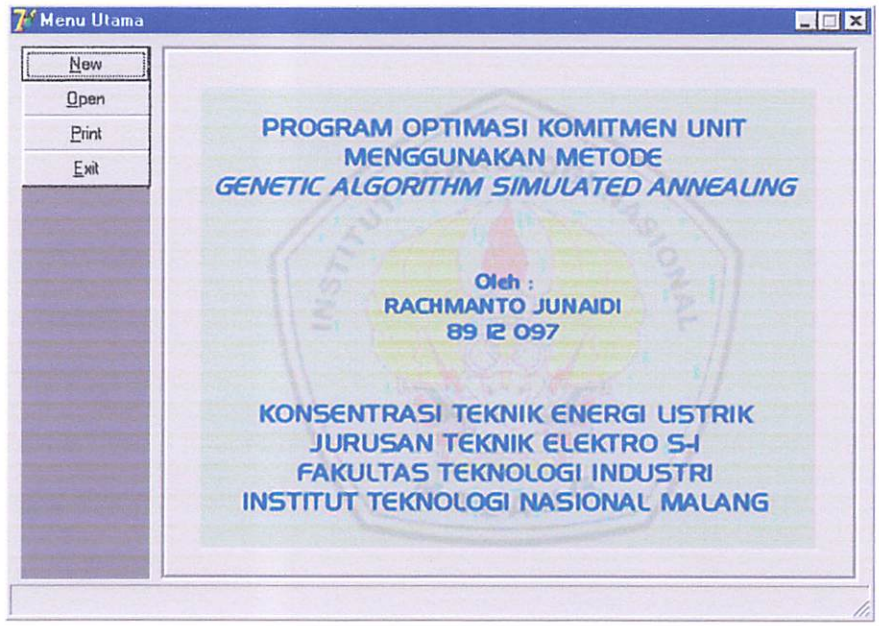


Gambar 3.11. Diagram Alir Metode GASA

3.8. Aplikasi Program Optimasi Komitmen Unit Dengan Metode GASA

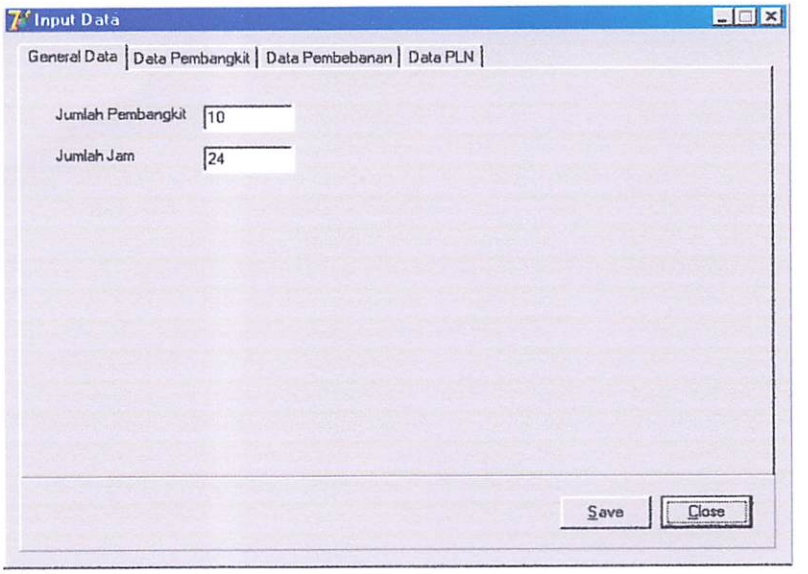
Dalam pemecahan masalah komitmen unit pada PT. Pembangkitan Jawa Bali digunakan bantuan program komputer yang bertujuan mempercepat proses perhitungan dengan tingkat ketelitian yang tinggi. Adapun spesifikasi komputer yang dipakai adalah komputer AMD Duron 1100 Mhz, SDRAM 192 MB.

1. Tampilan utama program



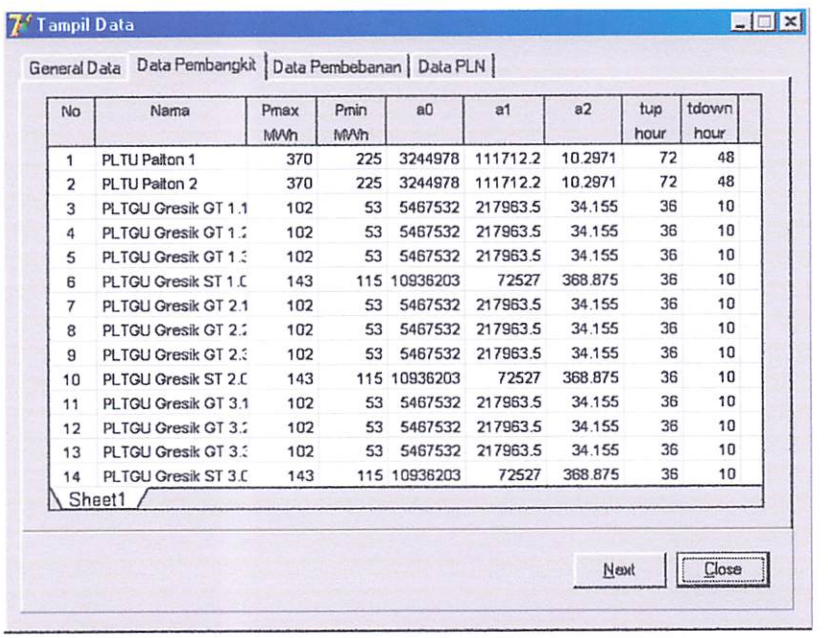
Gambar 3.12. Tampilan Utama Program

2. Tekan tombol "*General data*" untuk memasukan data, berapa jumlah pembangkitan, dan jumlah jam untuk eksekusi perhitungan komitmen unit.



Gambar 3.13. Tampilan *General Data*

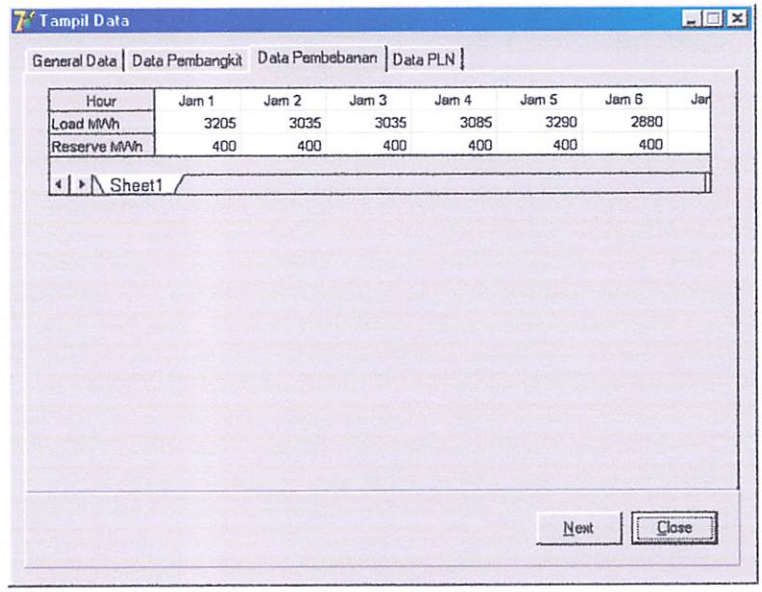
3. Tekan tombol "Data Pembangkitan" , masukan data penawaran pembangkitan yang beroperasi, Pmax, Pmin, Konstanta biaya a0, a1, a2, *minimum Up-Down time*, *Hot start-up, Cold start- up, time cold start,* dan *initial state.*



No	Nama	Pmax MWh	Pmin MWh	a0	a1	a2	tup hour	tdown hour
1	PLTU Paiton 1	370	225	3244978	111712.2	10.2971	72	48
2	PLTU Paiton 2	370	225	3244978	111712.2	10.2971	72	48
3	PLTGU Gresik GT 1.1	102	53	5467532	217963.5	34.155	36	10
4	PLTGU Gresik GT 1.2	102	53	5467532	217963.5	34.155	36	10
5	PLTGU Gresik GT 1.3	102	53	5467532	217963.5	34.155	36	10
6	PLTGU Gresik ST 1.0	143	115	10936203	72527	368.875	36	10
7	PLTGU Gresik GT 2.1	102	53	5467532	217963.5	34.155	36	10
8	PLTGU Gresik GT 2.2	102	53	5467532	217963.5	34.155	36	10
9	PLTGU Gresik GT 2.3	102	53	5467532	217963.5	34.155	36	10
10	PLTGU Gresik ST 2.0	143	115	10936203	72527	368.875	36	10
11	PLTGU Gresik GT 3.1	102	53	5467532	217963.5	34.155	36	10
12	PLTGU Gresik GT 3.2	102	53	5467532	217963.5	34.155	36	10
13	PLTGU Gresik GT 3.3	102	53	5467532	217963.5	34.155	36	10
14	PLTGU Gresik ST 3.0	143	115	10936203	72527	368.875	36	10

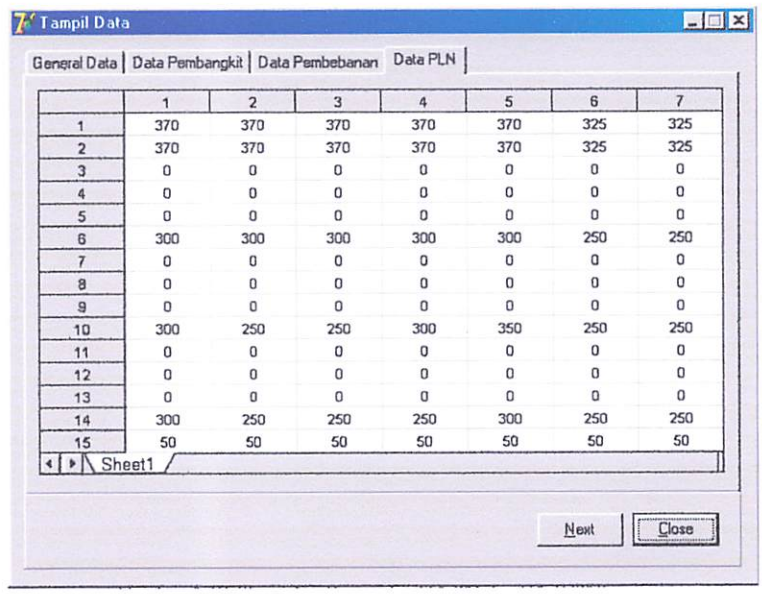
Gambar 3.14. Tampilan Data Pembangkitan

4. Tekan "Data Pembebanan", masukan nilai pembebanan dan *Spinning Reserve*



Gambar 3.15. Tampilan Data Pembebanan

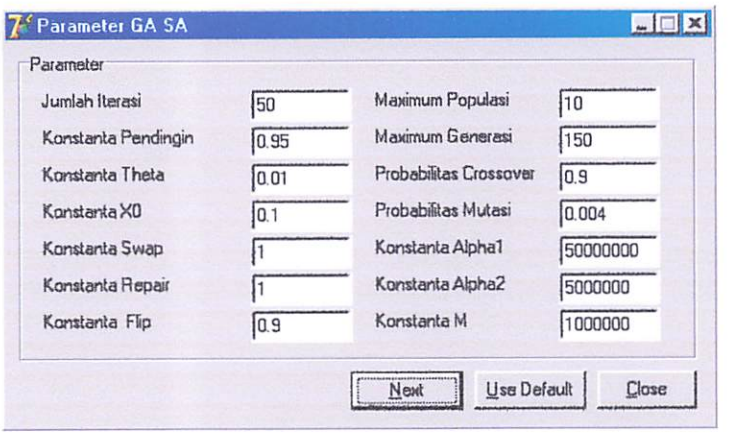
5. Masukan data pembebanan tiap jam tiap unit pembangkit yang beroperasi, dengan menekan tombol "Data PLN" .



	1	2	3	4	5	6	7
1	370	370	370	370	370	325	325
2	370	370	370	370	370	325	325
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	300	300	300	300	300	250	250
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	300	250	250	300	350	250	250
11	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0
14	300	250	250	250	300	250	250
15	50	50	50	50	50	50	50

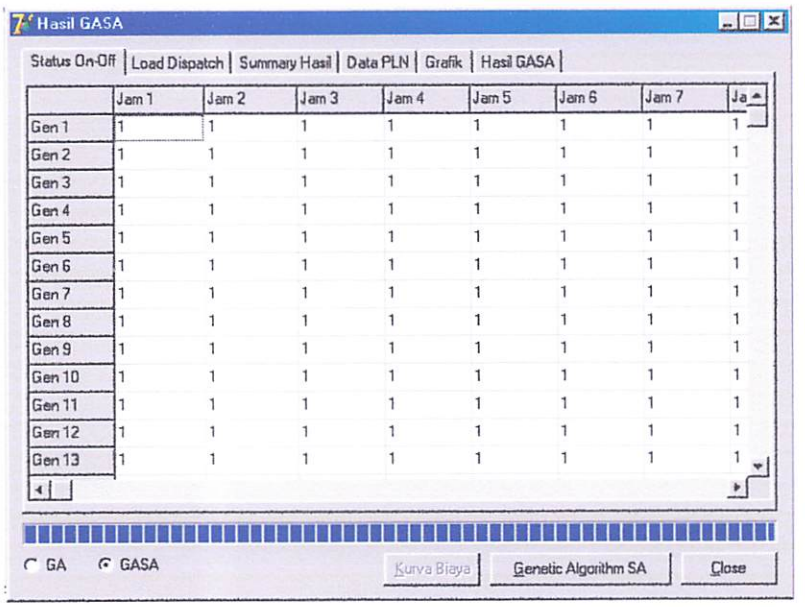
Gambar 3.16. Tampilan Data Pembebanan PLN Tiap Jam

6. Tekan tombol next, Untuk memasukkan "Parameter GASA" pada program

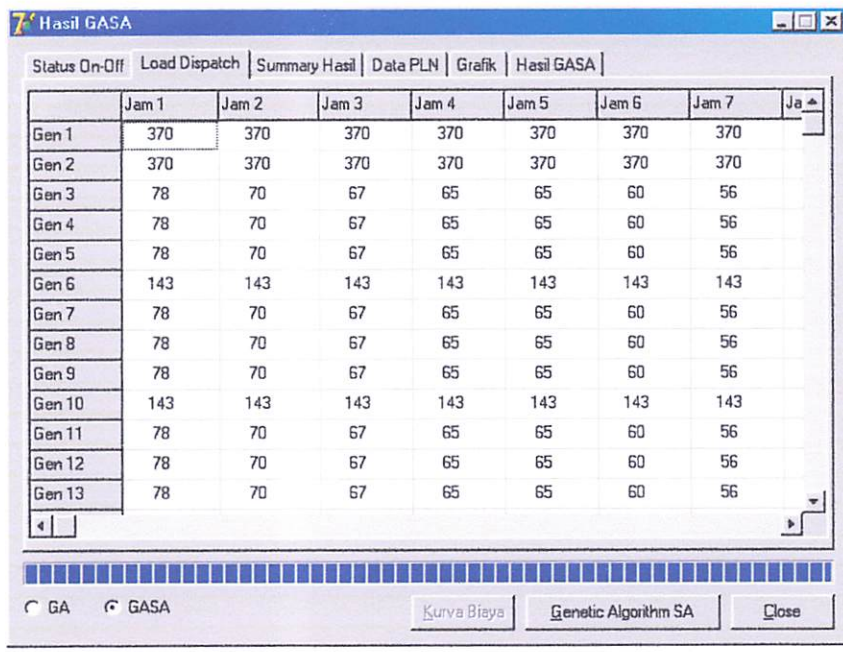


Gambar 3.17. Tampilan Parameter GASA

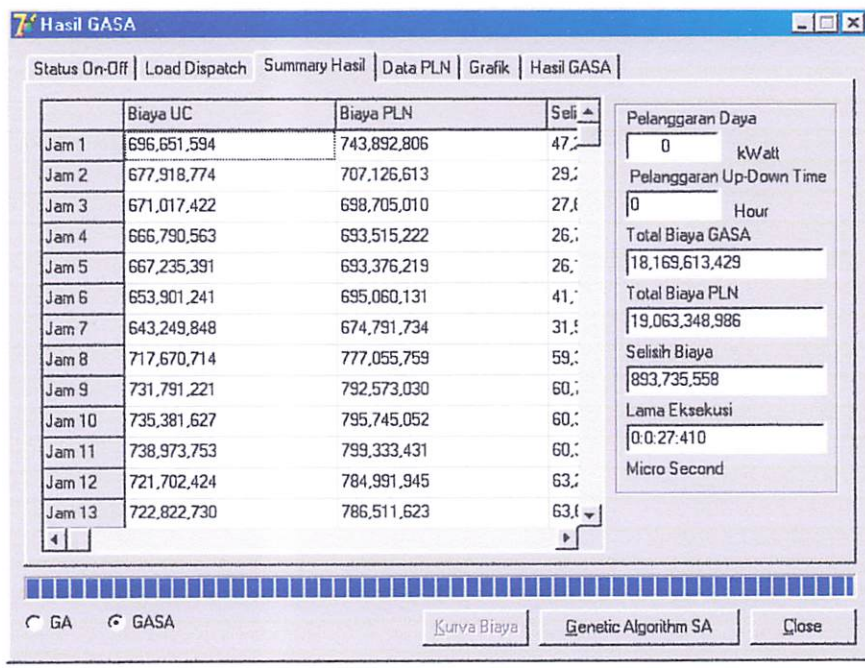
7. Kemudian tekan tombol "*Genetic Algorthm* SA" untuk memulai perhitungan, dan bisa dilihat hasil perhitungan dari : *status on-off, load dispatch,* biaya operasional, *summary result,* kurva beban dan kurva perbandingan biaya dari PLN dengan metode *Genetic Algorithm Simulated Annealing* (GASA).



Gambar 3.18. Status *On-Off* GASA



Gambar 3.19. *Load Dispatch* GASA



Gambar 3.20. *Summary* Hasil GASA

BAB IV
APLIKASI PROGRAM DENGAN METODE *GENETIC ALGORITHM SIMULATED ANNEALING* PADA P.T. PEMBANGKITAN JAWA BALI

4.1. Studi Pustaka

Penjadwalan dalam sistem tenaga listrik merupakan faktor yang paling penting, karena penjadwalan dalam sistem tenaga listrik saling terkait dengan biaya pengoperasian pembangkit (biaya bahan bakar, biaya start up, biaya shut down, dan biaya lainnya). Oleh karena itu, perencanaan penjadwalan operasi unit pembangkit harus sistematis baik dalam perencanaan jangka panjang ataupun perencanaan operasional pembangkit. Sehingga dicapai suatu operasi pembangkitan yang optimal. Kebutuhan energi listrik fluktuatif perubahannya tiap jam selama 24 jam sehari, sehingga sistim pembangkit memerlukan suatu perkiraan pembebanan yang tepat dan akurat. Dengan demikian dapat ditentukan unit pembangkit mana yang beroperasi untuk mencukupi kebutuhan tersebut agar dicapai biaya operasional pembangkitan listrik minimum[1].

4.2. Simulasi Penggunaan Metode GASA Pada PT. PJB

Perhitungan dan analisa ini dilakukan pada kebutuhan energi listrik yang ditanggung oleh PT. Pembangkitan Jawa Bali, pada hari Rabu, Sabtu dan Minggu pada tanggal 10, 13 dan 14 Maret 2004. analisa data dilakukan pada ketiga hari tersebut dikarenakan ketiga hari tersebut mewakili kurva karakteristik beban yang berlainan dengan keterangan sebagai berikut :

- Tanggal 10 Maret 2004 adalah hari Rabu dimana masyarakat lebih banyak melakukan aktivitas rutin khususnya dalam bidang pekerjaan, sehingga pada hari

tersebut kebutuhan energi relatif tinggi, karena sebagian besar industri dan perkantoran melakukan aktivitas kerjanya sepanjang hari. Dengan kata lain bahwa hari tersebut mewakili beban pada hari kerja efektif penuh.

- Tanggal 13 Maret 2004 adalah hari Sabtu dimana masyarakat merata kegiatannya. Karena tidak semua aktivitas industri dan perkantoran berjalan. Ada yang aktifitas kerjanya penuh, ada yang setengah hari dan ada pula yang libur. Sehinngga pada saat itu kebutuhan energi listrik relatif sedang. Sangat tepat sekali apabila hari tersebut mewakili beban pada hari kerja efektif setengah penuh.
- Tanggal 14 Maret 2004 adalah hari Minggu dimana kegiatan industri dan perkantoran tidak berjalan/libur. Sehingga kebutuhan listrik cenderung kecil.

Seluruh unit pembangkit yang beroperasi terdiri dari 38 unit pembangkit. Adapun unit pembangkit yang beroperasi seperti pada tabel 4.1.

Tabel 4.1.
Unit Pembangkit Yang Beroperasi Pada PT. PJB

Unit	Nama Unit	Unit	Nama Unit
Unit 1	PLTU Paiton 1	Unit 20	PLTG Gresik 2
Unit 2	PLTU Paiton 2	Unit 21	PLTG Gresik 3
Unit 3	PLTGU Gresik GT 1.1	Unit 22	PLTG Gilitimur 1
Unit 4	PLTGU Gresik GT 1.2	Unit 23	PLTG Gilitimur 2
Unit 5	PLTGU Gresik GT 1.3	Unit 24	PLTGU M.Karang GT 1.1
Unit 6	PLTGU Gresik ST 1.0	Unit 25	PLTGU M.Karang GT 1.2
Unit 7	PLTGU Gresik GT 2.1	Unit 26	PLTGU M.Karang GT 1.3
Unit 8	PLTGU Gresik GT 2.2	Unit 27	PLTGU M.Karang ST 1.0
Unit 9	PLTGU Gresik GT 2.3	Unit 28	PLTGU M.Tawar GT 1.1
Unit 10	PLTGU Gresik ST 2.0	Unit 29	PLTGU M.Tawar GT 1.2
Unit 11	PLTGU Gresik GT 3.1	Unit 30	PLTGU M.Tawar GT 1.3
Unit 12	PLTGU Gresik GT 3.2	Unit 31	PLTGU M.Tawar GT 2.1
Unit 13	PLTGU Gresik GT 3.3	Unit 32	PLTGU M.Tawar GT 2.2
Unit 14	PLTGU Gresik ST 3.0	Unit 33	PLTGU M.Tawar ST 1.0
Unit 15	PLTU Gresik 1	Unit 34	PLTU M.Karang 1
Unit 16	PLTU Gresik 2	Unit 35	PLTU M.Karang 2
Unit 17	PLTU Gresik 3	Unit 36	PLTU M.Karang 3
Unit 18	PLTU Gresik 4	Unit 37	PLTU M.Karang 4
Unit 19	PLTG Gresik 1	Unit 38	PLTU M.Karang 5

Sumber : Power Plant Data Sheet PT. PJB, Jl. Ketintang Baru No. 11 Surabaya

4.3. Data Unit Pembangkitan Termal

Jumlah unit pembangkit termal di PT. Pembangkitan Jawa Bali berdasarkan survey yang dilakukan di tempat tersebut berjumlah 38 unit pembangkitan, masing-masing terdiri dari 11 unit Pembangkit Listrik Tenaga Uap (PLTU), 5 unit Pembangkit Listrik Tenaga Gas (PLTG), dan 22 unit Pembangkit Listrik Tenaga Gas dan Uap (PLTGU) tersebar di seluruh wilayah Jawa dan Bali. Untuk lebih lengkapnya dapat dilihat pada tabel 4.2, sedangkan untuk data harga bahan bakar berdasar pada statistik PLN tahun 2002, dimana pada saat itu ditetapkan bahwa nilai tukar rupiah berada pada kisaran Rp. 9.000,00 per satu dollar Amerika.

Perincian harga bahan bakar untuk pembangkit termis yang termasuk ke dalam PT. PJB berdasarkan data tahun 2002 adalah sebagai berikut :

Gas Unit Pembangkit (UP) Gresik	: 2,53 US$ / MMBTU
Gas Unit Pembangkit (UP) Muara Karang	: 2,45 US$ / MMBTU
Batubara	: Rp. 235,- / Kg
MFO	: Rp. 1595,5,- / liter
HSD	: Rp. 1595,5,- / liter
Nilai Kalor MFO	: 9777,5 Kcal / liter
Nilai Kalor HSD	: 9063 Kcal / liter
Nilai tukar 1 US$	: Rp. 9000,-

Tabel 4.2.
Data Unit Termal Pada PT. PJB

No.	Nama Unit Pembangkit	Kapasitas (MW)		Lama Waktu (Jam)			
		Min	Max	MUT	MDT	Cold Start	Hot Start
1	PLTU Paiton 1	225	370	72	48	17	4
2	PLTU Paiton 2	225	370	72	48	17	4
3	PLTGU Gresik GT 1.1	53	102	36	10	1	0
4	PLTGU Gresik GT 1.2	53	102	36	10	1	0
5	PLTGU Gresik GT 1.3	53	102	36	10	1	0
6	PLTGU Gresik ST 1.0	115	143	36	10	3	1
7	PLTGU Gresik GT 2.1	53	102	36	10	1	0
8	PLTGU Gresik GT 2.2	53	102	36	10	1	0
9	PLTGU Gresik GT 2.3	53	102	36	10	1	0
10	PLTGU Gresik ST 2.0	115	143	36	10	3	2
11	PLTGU Gresik GT 3.1	53	102	36	10	1	0
12	PLTGU Gresik GT 3.2	53	102	36	10	1	0
13	PLTGU Gresik GT 3.3	53	102	36	10	1	0
14	PLTGU Gresik ST 3.0	115	143	36	10	3	2
15	PLTU Gresik 1	43	85	48	10	9	1
16	PLTU Gresik 2	43	85	48	10	9	1
17	PLTU Gresik 3	90	175	48	10	9	2
18	PLTU Gresik 4	90	175	48	10	9	2
19	PLTG Gresik 1	5	16	3	1	1	0
20	PLTG Gresik 2	5	16	3	1	1	0
21	PLTG Gresik 3	5	16	3	1	1	0
22	PLTG Gilitimur 1	5	16	3	1	1	0
23	PLTG Gilitimur 2	5	16	3	1	1	0
24	PLTGU M.Karang GT 1.1	50	95	36	10	1	0
25	PLTGU M.Karang GT 1.2	50	95	36	10	1	0
26	PLTGU M.Karang GT 1.3	50	95	36	10	1	0
27	PLTGU M.Karang ST 1.0	110	150	36	10	3	1
28	PLTGU M.Tawar GT 1.1	72	138	36	10	0	0
29	PLTGU M.Tawar GT 1.2	72	138	36	10	0	0
30	PLTGU M.Tawar GT 1.3	72	138	36	10	0	0
31	PLTGU M.Tawar GT 2.1	72	138	36	10	0	0
32	PLTGU M.Tawar GT 2.2	72	138	36	10	0	0
33	PLTGU M.Tawar ST 1.0	162	202	36	10	3	1
34	PLTU M.Karang 1	44	85	48	10	6	1
35	PLTU M.Karang 2	44	85	48	10	6	1
36	PLTU M.Karang 3	44	85	48	10	6	1
37	PLTU M.Karang 4	90	165	48	10	10	2
38	PLTU M.Karang 5	90	165	48	10	10	2

Sumber : Power Plant Data Sheet PT. PJB, Jl. Ketintang Baru No. 11 Surabaya

Tabel 4.3.
Parameter Unit Termal Pada PT. PJB

No.	Nama Unit Pembangkit	Biaya Start-Up (Juta Rupiah)		Koefisien Biaya Bahan Bakar		
		Cold Start-Up	Hot Start-Up	C	B	A
1	PLTU Paiton 1	682.98	149.68	32444978	111712.15	10.2971
2	PLTU Paiton 2	682.98	149.68	32444978	111712.15	10.2971
3	PLTGU Gresik GT 1.1	7.82	0	5467532.4	217963.548	34.155
4	PLTGU Gresik GT 1.2	7.82	0	5467532.4	217963.548	34.155
5	PLTGU Gresik GT 1.3	7.82	0	5467532.4	217963.548	34.155
6	PLTGU Gresik ST 1.0	57.68	31.46	10936203.3	72527.004	368.874
7	PLTGU Gresik GT 2.1	7.82	0	5467532.4	217963.548	34.155
8	PLTGU Gresik GT 2.2	7.82	0	5467532.4	217963.548	34.155
9	PLTGU Gresik GT 2.3	7.82	0	5467532.4	217963.548	34.155
10	PLTGU Gresik ST 2.0	57.68	31.46	10936203.3	72527.004	368.874
11	PLTGU Gresik GT 3.1	7.82	0	5467532.4	217963.548	34.155
12	PLTGU Gresik GT 3.2	7.82	0	5467532.4	217963.548	34.155
13	PLTGU Gresik GT 3.3	7.82	0	5467532.4	217963.548	34.155
14	PLTGU Gresik ST 3.0	57.68	31.46	10936203.3	72527.004	368.874
15	PLTU Gresik 1	143.74	40.59	1327126.68	217378.359	132.066
16	PLTU Gresik 2	143.74	40.59	1327126.68	217378.359	132.066
17	PLTU Gresik 3	229.5	92.52	5017369.5	169242.579	193.545
18	PLTU Gresik 4	229.5	92.52	5017369.5	169242.579	193.545
19	PLTG Gresik 1	6.13	0	352707.3	350680.77	903.969
20	PLTG Gresik 2	6.13	0	352707.3	350680.77	903.969
21	PLTG Gresik 3	6.13	0	352707.3	350680.77	903.969
22	PLTG Gilitimur 1	6.13	0	687181.85	683240.965	1762.3893
23	PLTG Gilitimur 2	6.13	0	687181.85	683240.965	1762.3893
24	PLTGU M.Karang GT 1.1	7.35	0	5730795	202052.97	108.045
25	PLTGU M.Karang GT 1.2	7.35	0	5730795	202052.97	108.045
26	PLTGU M.Karang GT 1.3	7.35	0	5730795	202052.97	108.045
27	PLTGU M.Karang ST 1.0	54.22	29.67	11560815	53685.135	460.845
28	PLTGU M.Tawar GT 1.1	0	0	14706521.3	433337.8	49.4605
29	PLTGU M.Tawar GT 1.2	0	0	14706521.3	433337.8	49.4605
30	PLTGU M.Tawar GT 1.3	0	0	14706521.3	433337.8	49.4605
31	PLTGU M.Tawar GT 2.1	0	0	14706521.3	433337.8	49.4605
32	PLTGU M.Tawar GT 2.2	0	0	14706521.3	433337.8	49.4605
33	PLTGU M.Tawar ST 1.0	118.08	64.4	672630	144191.717	519.1757
34	PLTU M.Karang 1	122.58	31.08	2417820.7	473895.41	120.77935
35	PLTU M.Karang 2	122.58	31.08	2417820.7	473895.41	120.77935
36	PLTU M.Karang 3	122.58	31.08	2417820.7	473895.41	120.77935
37	PLTU M.Karang 4	215.34	89.29	2949187.5	205217.145	83.79
38	PLTU M.Karang 5	215.34	89.29	2949187.5	205217.145	83.79

Sumber : Power Plant Data Sheet PT. PJB, Jl. Ketintang Baru No. 11 Surabaya

4.4. Beban Sistem

Proses komitmen unit dengan menggunakan metode Algoritma GASA bertujuan untuk membuat rencana penjadwalan unit pembangkit dalam sistem tenaga listrik yang dapat memenuhi kebutuhan beban dan juga mempunyai nilai ekonomis dan efisiensi yang tinggi.

Untuk mengetahui seberapa besar efisiensi ekonomi dari metode ini, maka dilakukan evaluasi dengan mengambil data unit-unit pembangkit dan beban yang ditanggung oleh PT. Pembangkitan Jawa Bali sebagai bahan pertimbangan. Sedangkan kombinasi jadwal dan daya *output* unit pembangkit tenaga listrik dalam sistem PT. Pembangkitan Jawa Bali pada tanggal 10, 13 dan 14 Maret 2004 terdapat pada lampiran. Untuk beban sistem yang ditanggung PT. Pembangkitan Jawa Bali untuk tanggal 10, 13 dan 14 Maret 2004 dapat dilihat pada tabel 4.7 (beban sistem yang ditanggung oleh pembangkit termis saja).

Besarnya cadangan berputar sistem diterapkan PT. PJB sebesar 400 MW untuk tiap jam. Cadangan ini bertujuan untuk mengantisipasi kemungkinan kehilangan suplai daya akibat gangguan, sedangkan cadangan berputar pada PT. PJB ditentukan sebesar daya unit pembangkit termis yang terbesar "trip" dan harus keluar dari sistem untuk wilayah kerja PT. PJB *single* unit terbesar yang dimiliki adalah PLTU Paiton 1 dan 2 dengan kapasitas daya terpasang sebesar 400 MW.

Tabel 4.4.
Data Permintaan Beban Unit Termis Pada PT. PJB
Pada Hari Rabu, Sabtu dan Minggu, Tanggal 10, 13 dan 14 Maret 2004

Jam	Rabu 10 Maret 2004		Sabtu 13 Maret 2004		Minggu 14 Maret 2004	
	Beban Sistem (MW)	Cadangan Putar (MW)	Beban Sistem (MW)	Cadangan Putar (MW)	Beban Sistem (MW)	Cadangan Putar (MW)
01.00	3108	400	2896	400	2816	400
02.00	3024	400	2864	400	2678	400
03.00	2993	400	2845	400	2675	400
04.00	2974	400	2866	400	2694	400
05.00	2976	400	2921	400	2804	400
06.00	2916	400	2806	400	2611	400
07.00	2868	400	2710	400	2588	400
08.00	3202	400	2856	400	2746	400
09.00	3265	400	3002	400	2802	400
10.00	3281	400	3020	400	2816	400
11.00	3297	400	3026	400	2853	400
12.00	3220	400	3030	400	2789	400
13.00	3225	400	3016	400	2749	400
14.00	3226	400	2901	400	2657	400
15.00	3297	400	2717	400	2613	400
16.00	3372	400	2796	400	2709	400
17.00	3499	400	2869	400	2714	400
18.00	3600	400	3374	400	3255	400
19.00	3657	400	3382	400	3268	400
20.00	3642	400	3373	400	3269	400
21.00	3403	400	3205	400	2982	400
22.00	3388	400	3015	400	2876	400
23.00	3335	400	2929	400	2864	400
24.00	3316	400	2869	400	2822	400

4.5. Eksekusi Program

Program optimalisasi penjadwalan pembebanan unit termal pada sistem PT. PJB dengan menggunakan metode Metode *Genetic Algorithm Simulated Annealing* (GASA), terdiri dari empat tahap yang kesemuanya harus dilakukan secara berurutan. Langkah-langkah kerja program komputer yang digunakan adalah sebagai berikut :

1. Memasukkan jumlah dan data setiap unit pembangkit tenaga listrik serta data pembebanan harian untuk periode 24 jam.

Adapun data-data yang diperlukan adalah jumlah unit pembangkit tenaga listrik, kapasitas daya maksimum(Pmaks), kapasitas daya minimum(Pmin), konstanta (A, B, C), biaya *hot start*, biaya *cold start*, besar nilai *ramp rate*, *Minimum Up Time*, *Minimum Down Time*, t *cold start*. Sedangkan data pembebanan meliputi data beban sistem tiap jam, dan data cadangan berputar sistem tiap jam.

2. Menentukan nilai parameter.

   Menentukan nilai parameter Algoritma GASA, hal ini dimaksudkan untuk mengeset agar jalannya eksekusi dapat maksimal dan mendapatkan hasil penjadwalan yang optimal. Pada dasarnya penentuan nilai parameter ini dilakukan secara coba-coba dan terus-menerus hingga mencapai hasil uji Algoritma GASA yang optimal. Inisialisasi parameter algoritma GASA. Dalam inisialisasi ini adalah sebuah inisial penjadwalan pada "*N*" unit dalam waktu "*T*" jam yang terdiri dari "0" dan "1".

3. Melakukan penjadwalan secara random dan menghitung pembebanan pada masing-masing pembangkit

4. Mencari kombinasi penjadwalan yang biaya operasionalnya rendah atau minimum dengan memperhatikan nilai fitness yang mempunyai nilai maksimum

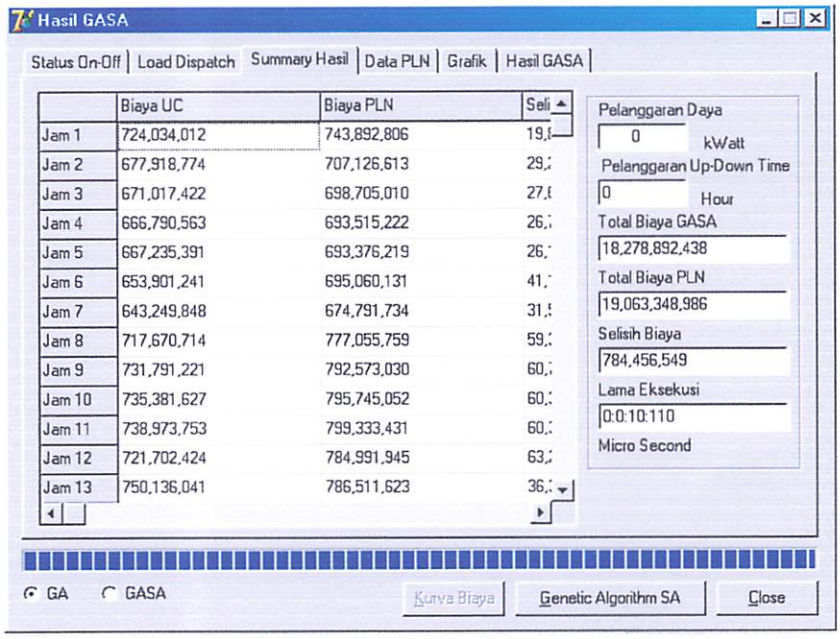
4.6. Hasil Perhitungan

Perhitungan biaya total menggunakan 2 metode, yaitu metode *Genetic Algorithm* (GA) dan metode *Genetic Algorithm Simulated Annealing* (GASA). Hasil perhitungan dengan menggunakan metode GA digunakan sebagai perbandingan sekaligus untuk menunjukkan apakah dengan metode GASA bisa diperoleh hasil yang lebih optimal.

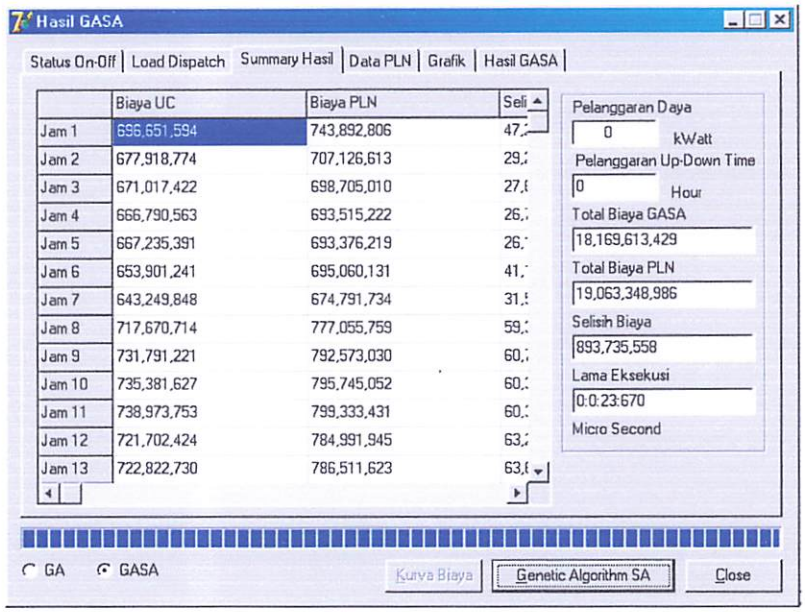
Proses komputasi dilakukan menggunakan CPU AMD Duron 1100, SDRAM 192 MB, dengan parameter: maksimum populasi 10, maksimum generasi 20, probabilitas *Crossover* 0.9 dan probabilitas mutasi 0.009.

Setelah program di eksekusi menggunakan data unit pembangkit yang diperoleh dari PT. PJB, serta dilakukan penjadwalan operasi berdasarkan permintaan beban pada hari Rabu, Sabtu dan Minggu, tanggal 10, 13 dan 14 Maret 2004, diperoleh hasil sebagai berikut:

1. Hari Rabu, 10 Maret 2004



Gambar 4.1.
Hasil Perhitungan Menggunakan Metode GA
Untuk Hari Rabu 10 Maret 2004



Gambar 4.2.
Hasil Perhitungan Menggunakan Metode GASA
Untuk Hari Rabu 10 Maret 2004

Tabel 4.5.
Kombinasi Penjadwalan Unit Termal Pada PT.PJB Menggunakan Metode GASA
Untuk Beban Hari Rabu Tanggal 10 Maret 2004

Jam	Status ON dan OFF Unit Pembangkit																													Beban
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3108
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3024
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2993
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2974
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2976
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2916
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2868
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3202
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3265
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3281
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3297
12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3220
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3225
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3226
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3297
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1	1	3372
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	3499
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	3600
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	3657
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	3642
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	3403
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	3388
23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	3335
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	3316

Keterangan : 1 = ON dan 0 = OFF

Untuk menghitung biaya bahan bakar, digunakan persamaan :

$$F_{it}(P_{it}) = A_i\, P^2_{it} + B_i\, P_{it} + C_i \quad ; \quad Rp / HR \quad \ldots\ldots (4.1)$$

Dan

$$S(t) = \begin{cases} S_h & if -x(t) \le t_{cold\ start} \\ S_c & otherwise \end{cases} \quad \ldots\ldots (4.2)$$

Dimana nilai – nilai untuk A, B dan C dapat dilihat pada tabel 4.3, sedangkan untuk P adalah permintaan beban pada jam tertentu, dapat dilihat pada tabel 4.4.

Sehingga persamaan perhitungan bahan bakar yang digunakan adalah :

$$R = F_{it}(P_{it}) + S(t) \quad \ldots\ldots (4.3)$$

$$= A_i\, P^2_{it} + B_i\, P_{it} + C_i + S(t) \quad \ldots\ldots (4.4)$$

Contoh perhitungan:

Untuk biaya bahan bakar menggunakan metode GA pada hari Rabu, 10 Maret 2004, jam ke 1 adalah :

R = ((10,2971 (370)^2 + 111712,15 (370) + 32444978) x 2) + ((34,155 (69)^2 + 217963,548 + 5467532,4) x 3) + (368,874 (143)^2 +72527,004 (143) + 10936203,3) + ((34,155 (69)^2 + 217963,548 + 5467532,4) x 3) + (368,874 (143)^2 +72527,004 (143) + 10936203,3) + ((34,155 (69)^2 + 217963,548 + 5467532,4) x 3) + (368,874 (143)^2 +72527,004 (143) + 10936203,3) + ((132,066 (43)^2 + 217378,359 (43) + 1327126,68) x 2) + ((193,545 (138)^2 + 169242,579 (138) + 5017369,5) x 2) + ((903,969 (5)^2 + 350680,77 (5) + 352707,3) x 3) + ((108,045 (95)^2 + 202052,97 (95) + 5730795) x 3) + (460,845 (150)^2 + 53685,135 (150) + 11560815) + (519,1757 (162)^2 + 144191,717 (162) + 672630) + ((120,77935 (44)^2 + 473895,41 (44) + 2417820,7) x 3) + ((83,79 (104)^2 + 205217,145 (104) + 2949187,5) x 2) + S(t)

R = 724.034.012

Tabel 4.6.
Perbandingan Biaya Operasional Per Jam PT. PJB Dengan Metode
Genetic Algorithm (GA) dan *Genetic Algorithm Simulated Annealing* (GASA)
Pada Hari Rabu, 10 Maret 2004

JAM	PT. PJB (Rupiah)	GA (Rupiah)	GASA (Rupiah)
1	743.892.806	724.034.012	696.651.584
2	707.126.613	677.918.774	677.918.774
3	698.705.010	671.017.422	671.017.422
4	693.515.222	666.790.563	666.790.563
5	693.376.219	667.235.391	667.235.391
6	695.060.131	653.901.241	653.901.241
7	674.791.734	643.249.848	643.249.848
8	777.055.759	717.670.714	717.670.714
9	792.573.030	731.791.221	731.791.221
10	795.745.052	735.381.627	735.381.627
11	799.333.431	738.973.753	738.973.753
12	784.991.945	721.702.424	721.702.424
13	786.511.623	750.136.041	722.822.730
14	780.284.232	750.359.469	723.046.811
15	805.579.121	766.244.437	738.973.753
16	826.760.396	783.061.114	783.061.176
17	887.879.802	841.580.606	841.580.606
18	919.819.141	864.519.741	864.519.741
19	940.069.733	907.242.570	907.242.570
20	935.744.999	903.811.722	903.811.722
21	843.486.479	850.064.903	850.064.903
22	839.672.980	846.709.931	846.709.931
23	823.658.332	834.867.804	834.867.804
24	817.715.157	830.627.109	830.627.109
Total	19.063.348.986	18.278.892.438	18.169.613.429

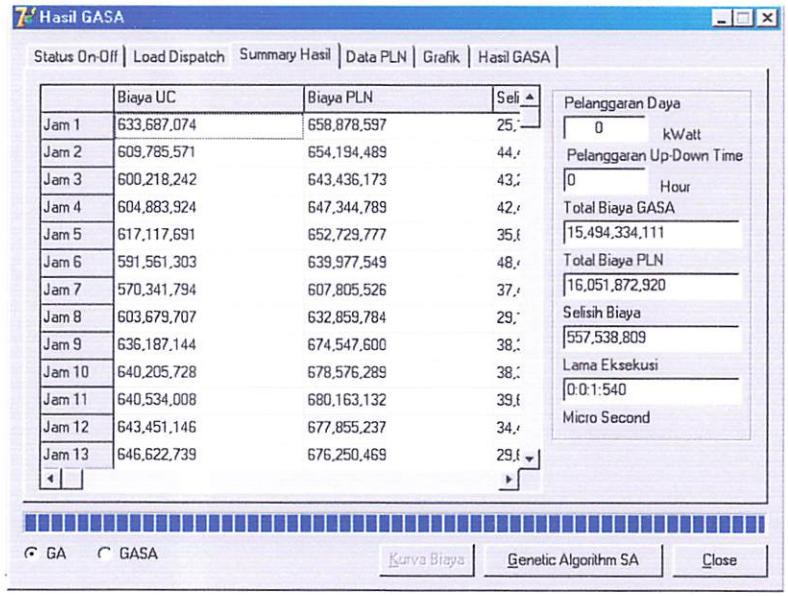
Dari Tabel 4.6. dapat dilihat bahwa menggunakan metode GASA diperoleh hasil perhitungan biaya bahan bakar paling ekonomis, baik bila dibandingkan dengan hasil perhitungan menggunakan metode GA, maupun hasil perhitungan PT. PJB.

Selisih biaya yang diperoleh untuk perhitungan selama 24 jam untuk hari Rabu , 10 Maret 2004 adalah :

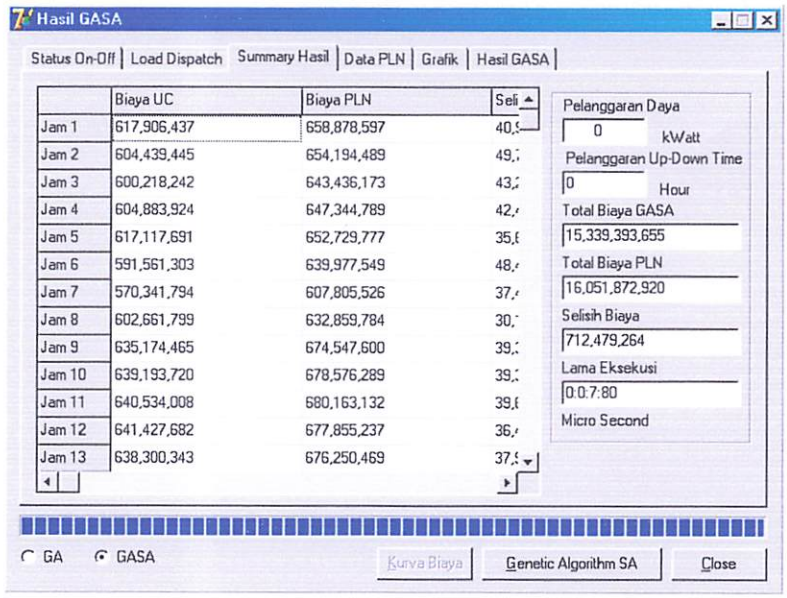
Metode GA : 19.063.348.986 - 18.278.892.438 = Rp. 784.456.550,-

Metode GASA : 19.063.348.986 - 18.169.613.429 = Rp. 893.735.560,-

2. Hari Sabtu, 13 Maret 2004



Gambar 4.3.
Hasil Perhitungan Menggunakan Metode GA
Untuk Hari Sabtu, 13 Maret 2004



Gambar 4.4.
Hasil Perhitungan Menggunakan Metode GASA
Untuk Hari Sabtu, 13 Maret 2004

Tabel 4.7.
Kombinasi Penjadwalan Unit Termal Pada PT.PJB Menggunakan Metode GASA
Untuk Beban Hari Sabtu Tanggal 13 Maret 2004

Jam	Status ON dan OFF Unit Pembangkit	Beban
1	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2896
2	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2864
3	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2845
4	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2866
5	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2921
6	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2806
7	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2710
8	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2856
9	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	3002
10	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	3020
11	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	3026
12	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	3030
13	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	3016
14	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2901
15	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2717
16	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2796
17	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1	2869
18	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1	3374
19	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1	3382
20	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1	3373
21	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1	3205
22	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1	3015
23	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1	2929
24	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1	2869

Keterangan : 1 = ON dan 0 = OFF

Tabel 4.8.
Perbandingan Biaya Operasional Per Jam PT. PJB Dengan Metode *Genetic Algorithm* (GA) dan *Genetic Algorithm Simulated Annealing* (GASA) Pada Hari Sabtu, 13 Maret 2004

JAM	PT. PJB (Rupiah)	GA (Rupiah)	GASA (Rupiah)
1	658.878.597	633.687.074	617.906.437
2	654.194.489	609.785.571	604.439.445
3	643.436.173	600.218.242	600.218.242
4	647.344.789	604.883.924	604.883.924
5	652.729.777	617.117.691	617.117.691
6	639.977.549	591.561.303	591.561.303
7	607.805.526	570.341.794	570.341.794
8	632.859.784	603.679.707	602.661.799
9	674.547.600	636.187.144	635.174.465
10	678.576.289	640.205.728	639.193.720
11	680.163.132	640.534.008	640.534.008
12	677.855.237	643.451.146	641.427.682
13	676.250.469	646.622.739	638.300.343
14	656.832.995	621.053.861	612.666.679
15	600.001.501	578.467.543	571.878.519
16	615.102.712	596.773.414	589.343.229
17	631.923.977	613.953.522	611.914.296
18	780.437.695	767.634.175	754.003.115
19	782.253.343	769.427.773	755.799.229
20	777.934.293	767.409.590	753.778.386
21	717.054.020	729.836.731	716.155.867
22	672.833.869	687.568.289	673.833.615
23	654.056.305	668.529.697	654.751.983
24	638.822.798	655.403.444	641.507.844
Total	16.051.872.920	15.494.334.111	15.339.393.655

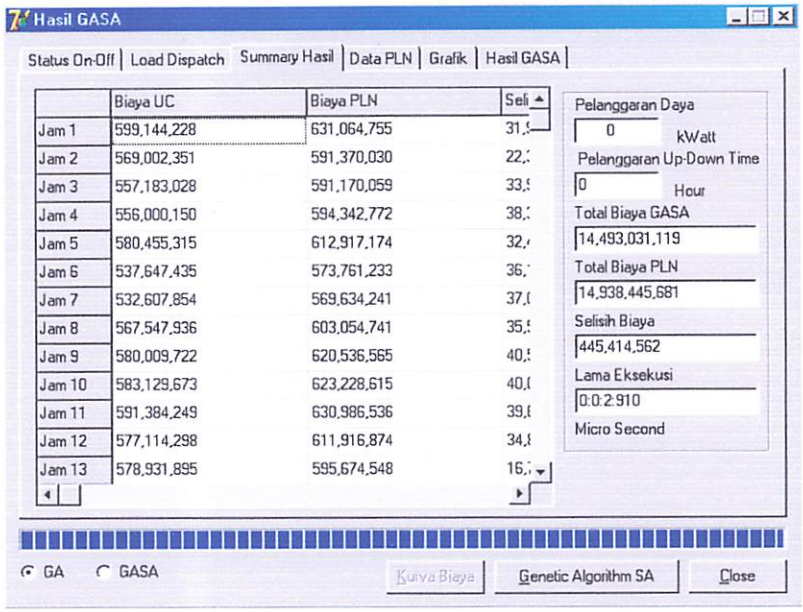
Dari Tabel 4.8. dapat dilihat bahwa menggunakan metode GASA diperoleh hasil perhitungan biaya bahan bakar paling ekonomis, baik bila dibandingkan dengan hasil perhitungan menggunakan metode GA, maupun hasil perhitungan PT. PJB.

Selisih biaya yang diperoleh untuk perhitungan selama 24 jam untuk hari Sabtu , 13 Maret 2004 adalah :

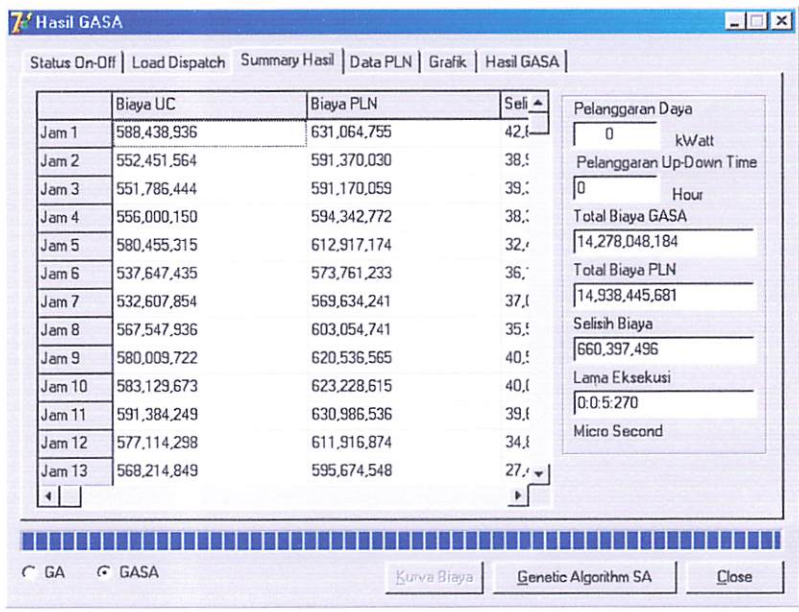
Metode GA : 16.051.872.920 - 15.494.334.111 = Rp. 557.538.810,-

Metode GASA : 16.051.872.920 - 15.339.393.655 = Rp. 712.479.270,-

3. Hari Minggu, 14 Maret 2004



Gambar 4.5.
Hasil Perhitungan Menggunakan Metode GA
Untuk Hari Minggu, 14 Maret 2004



Gambar 4.6.
Hasil Perhitungan Menggunakan Metode GASA
Untuk Hari Minggu, 14 Maret 2004

Tabel 4.9.
Kombinasi Penjadwalan Unit Termal Pada PT.PJB Menggunakan Metode GASA
Untuk Beban Hari Minggu Tanggal 14 Maret 2004

Jam	Status ON dan OFF Unit Pembangkit																															Beban
1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2816
2	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2678
3	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2675
4	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2694
5	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2804
6	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2611
7	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2588
8	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2746
9	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2802
10	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2816
11	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2853
12	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2789
13	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2749
14	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2657
15	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2613
16	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2709
17	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	2714
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3255
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3268
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	3269
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2982
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2876
23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2864
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	2822

Keterangan : 1 = ON dan 0 = OFF

Tabel 4.10.
Perbandingan Biaya Operasional Per Jam PT. PJB Dengan Metode *Genetic Algorithm* (GA) dan *Genetic Algorithm Simulated Annealing* (GASA) Pada Hari Minggu, 14 Maret 2004

JAM	PT. PJB (Rupiah)	GA (Rupiah)	GASA (Rupiah)
1	631.064.755	599.144.228	588.438.936
2	591.370.030	569.002.351	552.451.564
3	591.170.059	557.183.028	551.786.444
4	594.342.772	556.000.150	556.000.150
5	612.917.174	580.455.315	580.455.315
6	573.761.233	537.647.435	537.647.435
7	569.634.241	532.607.854	532.607.854
8	603.054.741	567.547.936	567.547.936
9	620.536.565	580.009.722	580.009.722
10	623.228.615	583.129.673	583.129.673
11	630.986.536	591.384.249	591.384.249
12	611.916.874	577.114.298	577.114.298
13	595.674.548	578.931.895	568.214.849
14	577.580.171	558.790.053	553.226.260
15	568.048.552	556.286.934	543.625.734
16	586.605.717	576.735.852	570.122.480
17	587.757.935	579.912.685	571.219.608
18	746.052.082	713.680.273	713.680.299
19	748.341.268	716.592.857	716.592.857
20	747.934.651	744.128.090	716.816.958
21	652.111.766	680.249.816	652.774.519
22	629.547.083	656.925.103	629.254.848
23	627.003.838	654.318.119	626.599.681
24	617.804.475	645.253.201	617.346.514
Total	14.938.445.681	14.493.031.119	14.278.048.184

Dari Tabel 4.10. dapat dilihat bahwa menggunakan metode GASA diperoleh hasil perhitungan biaya bahan bakar paling ekonomis, baik bila dibandingkan dengan hasil perhitungan menggunakan metode GA, maupun hasil perhitungan PT. PJB.

Selisih biaya yang diperoleh untuk perhitungan selama 24 jam untuk hari Sabtu , 13 Maret 2004 adalah :

Metode GA : 14.938.445.681 - 14.493.031.119 = Rp. 445.414.570,-

Metode GASA : 14.938.445.681 - 14.278.048.184 = Rp. 660.397.500,-

4.7. Rekapitulasi Data Hasil Perhitungan

Berdasarkan data yang diperoleh dari hasil eksekusi program yang dilakukan untuk permintaan beban pada hari Rabu, Sabtu dan Minggu, tanggal 10,13, dan 14 Maret 2004, didapatkan suatu rekapitulasi data sebagai berikut :

Tabel 4.11.
Perbandingan Total Biaya Operasional PT. PJB Dengan Hasil Perhitungan Menggunakan Metode GA dan GASA

Periode Waktu (24 jam)	Total Biaya Operasional PT. PJB (Rupiah)	Total Biaya Operasional GA (Rupiah)	Total Biaya Operasional GASA (Rupiah)
Rabu, 10 Maret 2004	19.063.348.986	18.278.892.438	18.169.613.429
Sabtu, 13 Maret 2004	16.051.872.920	15.494.334.111	15.339.393.655
Minggu, 14 Maret 2004	14.938.445.681	14.493.031.119	14.278.048.184

Pada Tabel 4.11. dapat dilihat bahwa dengan menggunakan metode GASA diperoleh hasil perhitungan biaya bahan bakar paling ekonomis, baik bila dibandingkan dengan hasil perhitungan menggunakan metode GA, maupun hasil perhitungan PT. PJB.

Tabel 4.12.
Selisih Total Biaya Operasional PT. PJB Dengan Hasil Perhitungan Menggunakan Metode GA dan GASA

Periode Waktu (24 jam)	Selisih dengan GA (Rupiah)	Selisih dengan GASA (Rupiah)
Rabu, 10 Maret 2004	784.456.550	893.735.560
Sabtu, 13 Maret 2004	557.538.810	712.479.270
Minggu, 14 Maret 2004	445.414.570	660.397.500

Pada Tabel 4.12. dapat dilihat bahwa dengan menggunakan metode GASA didapat selisih biaya bahan bakar lebih besar bila dibandingkan dengan menggunakan metode GA.

Tabel 4.13.
Prosentase Penekanan Biaya Operasional PT. PJB Menggunakan Perhitungan Metode GA dan GASA

Periode Waktu (24 jam)	Metode GA (%)	Metode GASA (%)
Rabu, 10 Maret 2004	4,11	4,69
Sabtu, 13 Maret 2004	3,47	4,44
Minggu, 14 Maret 2004	2,98	4,42

Pada Tabel 4.13. dapat dilihat bahwa dengan menggunakan metode GASA diperoleh prosentase penekanan biaya bahan bakar lebih besar bila dibandingkan dengan metode GA.

Rata-rata prosentase penekanan biaya bahan bakar untuk hari Rabu, Sabtu dan Minggu, tanggal 10,13 dan 14 Maret 2004 adalah sebagai berikut :

Metode GA:

$$\frac{4{,}11\ \% + 3{,}47\ \% + 2{,}98\ \%}{3} = 3{,}52\ \%$$

Metode GASA :

$$\frac{4{,}69\ \% + 4{,}44\ \% + 4{,}42\ \%}{3} = 4{,}52\ \%$$

Tabel 4.14.
Waktu Proses Komputasi Menggunakan Metode GA dan GASA

Periode Waktu (24 jam)	Metode GA (detik)	Metode GASA (detik)
Rabu, 10 Maret 2004	10,11	23,73
Sabtu, 13 Maret 2004	1,54	7,80
Minggu, 14 Maret 2004	2,91	5,27

Pada Tabel 4.14. dapat dilihat bahwa perhitungan menggunakan metode GASA memerlukan waktu perhitungan lebih lama bila dibandingkan dengan perhitungan menggunakan metode GA.

Rata-rata waktu perhitungan yang diperlukan untuk menghitung biaya bahan bakar untuk hari Rabu, Sabtu dan Minggu, tanggal 10,13 dan 14 Maret 2004 adalah sebagai berikut :

Metode GA:

$$\frac{10,11 + 1,54 + 2,91}{3} = 4,85 \text{ detik}$$

Metode GASA :

$$\frac{23,73 + 7,80 + 5,27}{3} = 12,27 \text{ detik}$$

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil eksekusi program dan analisa terhadap hasil perhitungan menggunakan metode *Genetic Algorithm* (GA) dan *Genetic Algorithm Simulated Annealing* (GASA) dan dibandingkan hasilnya dengan hasil perhitungan biaya berdasarkan data yang diperoleh dari PT. Pembangkitan Jawa-Bali (PT. PJB) dapat diambil kesimpulan sebagai berikut :

1. Metode GASA menekan total biaya bahan bakar lebih besar baik bila dibandingkan dengan hasil perhitungan PT. PJB maupun hasil perhitungan menggunakan metode GA. Metode GA menekan biaya bahan bakar rata-rata sebesar 3,52 %, sedangkan metode GASA menekan biaya bahan bakar rata-rata sebesar 4,52 %.
2. Proses komputasi dengan metode GASA memerlukan waktu yang lebih lama bila dibandingkan dengan metode GA. Metode GA memerlukan waktu komputasi rata-rata selama 4,85 detik, sedangkan metode GASA memerlukan waktu komputasi rata-rata selama 12,27 detik.
3. Memungkinkan untuk diterapkannya metode GASA pada PT. PJB karena optimasi yang diharapkan terpenuhi dengan biaya yang dihasilkan lebih ekonomis dan eksekusi perhitungan yang relatif cepat.

5.2. Saran-saran

1. Penggunaan program dengan metode GASA ini akan lebih praktis apabila input data permintaan beban dapat dilakukan untuk beberapa hari sekaligus, sehingga operator tidak perlu melakukan hal yang sama setiap hari.
2. Pada simulasi program dianggap bahwa keseluruhan unit pembangkit dalam keadaan siap untuk beroperasi, padahal pada kenyataan sesungguhnya tidak menutup kemungkinan adanya satu atau lebih unit pembangkit yang sedang mengalami kerusakan atau sedang mengalami perbaikan rutin. Oleh karena itu akan lebih baik bila pada input data pembangkit disediakan opsi untuk menentukan bahwa suatu unit pembangkit siap beroperasi atau tidak tanpa harus menghapus data unit pembangkit tersebut. Untuk selanjutnya opsi tersebut dapat digunakan untuk menentukan ikut atau tidaknya unit pembangkit tersebut dalam proses perhitungan.
3. Tidak menutup kemungkinan untuk mengembangkan program dengan metode GASA ini untuk dapat di implementasikan penggunaannya sebagai bagian dari sistem Kecerdasan Buatan, mengingat dewasa ini sistem Kecerdasan Buatan atau biasa disebut dengan *Artificial Intelligence* (AI) telah digunakan secara luas, termasuk untuk menyelesaikan masalah optimasi Komitmen Unit.

DAFTAR PUSTAKA

[1]. Djiteng Marsudi, Ir., "Operasi Sistem Tenaga Listrik", ISTN 1990

[2]. Allan J Wood and Wollenberg B.F. " *Power Generation, Operation, and Control* ", John Wiley & Sons, Inc., 1996.

[3]. Kazarlis, S.A., Bakirtzis,A.G., Petridis, V.,"*A Genetic Algorithm Solution To The Unit Commitment Problem*", IEEE Trans. On Power Systems, Vol.11, No.1, Feb. 1996.

[4]. Mantawy, A.H., Abdel-Magid, Y.L., Selim, S.Z., "*Integrating Genetic Algorithms, Tabu Search, And Simulated Annealing For The Unit Commitment Problem*", IEEE Trans on Power Systems, Vol.14,No.3, Aug. 1999.

[5]. Mantawy, A.H., Abdel-Magid, Y.L., Selim, S.Z., "*A Simulated Annealing Algorithm For Unit Commitment*", IEEE Trans on Power Systems, Vol.13, No.1, Feb. 1998.

[6]. Orero,S.O., Irving, M.R., "*A Genetic Algorithm For Generator Scheduling In Power Systems*", Electrical Power & Energy Systems, Vol.18, No.1, 1996

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : Rachmanto Junaidi
Nim : 89.12.097
Masa Bimbingan : 1 Maret 2004 – 1 September 2004

Judul Skripsi : **OPTIMASI KOMITMEN UNIT MENGGUNAKAN METODE *GENETIC ALGORITHM SIMULATED ANNEALING* (GASA) PADA PEMBANGKITAN JAWA BALI II (PJB II)**

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	14-07-2004	BAB IV. Validasi Program, membandingkan biaya program dengan jurnal	
2.	26-07-2004	BAB IV. Validasi Program, selisih biaya program masih terlalu besar	
3.	02-08-2004	BAB IV. Validasi Program, relatif *error* pada program kurang kecil	
4.	05-09-2004	BAB IV. Modifikasi program, hasil keluaran GA ditampilkan	
5.	09-09-2004	BAB IV. Prosentasi total biaya perhitungan hasil eksekusi program dengan PT.PJB	
6.	10-09-2004	BAB V. Kesimpulan, selisih total biaya pada eksekusi program, parameter yang digunakan, dan waktu	
7.	25-09-2004	Membuat Makalah Seminar Hasil	
8.	01-10-2004	Memperbaiki sistim penulisan Abstraksi. Batasan Masalah yang digunakan, melengkapi Tinjauan Pustaka.	
9.	20-10-2004	Koreksi BAB I, Pendahuluan, dan BAB II, Tinjauan Pustaka, untuk BAB III dan BAB IV, gambar dicetak berwarna	
10.	25-10-2004	Acc Untuk Ujian Skripsi	

Malang, 25 Oktober 2004
Dosen Pembimbing

(**Ir. Yusuf Ismail Nakhoda, MT**)

Form.S-4b

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

1. Nama : Rachmanto Junaidi
2. NIM : 8912097
3. Jurusan : Teknik Elektro
4. Konsentrasi : Teknik Energi Listrik S-1
5. Judul Skripsi : OPTIMASI KOMITMEN UNIT MENGGUNAKAN METODE *GENETIC ALGORITHM SIMULATED ANNEALING* (GASA) PADA PEMBANGKITAN JAWA BALI II (PJB II)

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

6. Hari : Jumat
7. Tanggal : 23 Maret 2007
8. Dengan Nilai : 79,2 (B+)

Panitia Majelis Penguji

Ketua

(Ir. Mochtar Asroni, MSME)
NIP.Y.101 81 00036

Sekretaris

(Ir. F. Yudi Limpraptono, MT)
NIP.Y.103 95 00274

Anggota Majelis Penguji

Penguji I

(Ir. Djojo Priatmono, MT)
NIP.Y.101 85 00107

Penguji II

(Bambang Prio Hartono, ST)
NIP.Y.102 84 00082

Lampiran

PJB

DATA PENAWARAN
PT PLN PEMBANGKITAN JAWA BALI
AGUSTUS 2002

No.	NAMA PEMBANGKIT	KAPASITAS			LAMA WAKTU (JAM)				BIAYA START UP (JUTA Rp)		KOEFISIEN BIAYA BAHAN BAKAR		
		Daya Terpasang	MIN (MW)	MAX (MW)	MIN UP TIME	MIN DOWN TIME	COLD START UP	HOT START UP	COLD START UP	HOT START UP	a0	a1	a2
P	**UP. PAITON**												
	PLTU #1/2 (COAL)	2 x 400	225	370	72	48	17	4	682.98	149.68	3244978	111712.15	10.2971
2	**UP. GRESIK**												
	GT 1-9 OC (GAS)	9 x 112	53	102	36	10	1	0	7.82	0	5467532.4	217963.548	34.155
	CC - 1.1.1 (GAS)		115	143	36	10	3	1	57.68	31.46	10936203.3	72527.004	368.874
	CC - 2.2.1 (GAS)		164	314	36	10	3	2	65.5	39.28	11795770.8	152515.737	6.831
	CC - 3.3.1 (GAS)	3 x 526	250	480	36	10	3	2	73.32	47.1	17177460.3	145165.581	4.554
	PLTU # 1/2 (GAS)	100	43	85	48	10	9	1	143.74	40.59	1327126.68	217378.359	132.066
	PLTU # 3/4 (GAS)	200	90	175	48	10	9	2	229.5	92.52	5017369.5	169242.579	193.545
	PLTG GRESIK 1-3 (GAS)	3 x 20	5	16	3	1	1	0	6.13	0	352707.3	350680.77	903.969
	PLTG GILITIMUR 1-2 (HSD)	2 x 20	5	16	3	1	1	0	6.33	0	687181.85	683240.965	1762.3893
3	**UP. MUARA KARANG**												
	GT 1/2/3 - OC	3 x 107	50	95	36	10	1	0	7.35	0	5730795	202052.97	108.045
	CC - 1.1.1 (GAS)	153	110	150	36	10	3	1	54.22	29.67	11560815	53685.135	460.845
	CC - 2.2.1 (GAS)	317	200	300	36	10	3	2	61.57	36.92	16010064	127208.655	35.28
	CC - 3.3.1 (GAS)	508	300	465	36	10	3	2	68.92	44.27	31017735	87825.15	57.33
	MTW GT 1/2 - OC (HSD)	2 x 140	72	138	36	10	0	0	0	0	14706521.25	433337.8	49.4605
	MTW CC - 1.1.1 (HSD)	200	162	202	36	10	3	1	118.08	64.4	672630	144191.717	519.1757
	MTW CC - 2.2.1 (HSD)	420	210	403	36	10	3	2	134.1	80.42	30123040	303208.82	11.64715
	MTW CC - 3.3.1 (HSD)	640	315	605	36	10	3	2	160.1	96.42	43043399	288609.995	7.6584
	PLTU # 1/2/3 (MFO)	3 x 100	44	85	48	10	6	1	122.58	31.08	2417820.7	473895.41	120.77935
	PLTU # 4/5 (Gas)	2 x 200	90	165	48	10	11	2	215.34	89.29	2949187.5	205217.145	83.79

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL: RABU, 10 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	00.30	01.00	01.30	02.00	02.30	03.00	03.30	04.00	04.30	05.00	05.30	06.00	06.30	07.00	07.30	08.00	08.30	09.00	09.30	10.00	10.30	11.00	11.30	12.00	12.30
PLTGU	MKRNG10C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG30C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG1CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG2CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3CC		435	435	410	410	410	410	410	410	425	425	410	410	410	410	435	435	435	435	435	435	435	435	435	435	435
PLTU	MKRNG	#4	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165
		#5	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165
	MKRNG	#1	80	65	65	65	65	65	65	60	60	60	60	60	60	95	95	95	95	95	95	95	95	95	95	95	95
		#2	80	65	65	65	65	65	65	60	60	60	60	60	60	95	95	95	95	95	95	95	95	95	95	95	95
		#3	80	65	65	65	65	65	65	60	60	60	60	60	60	95	95	95	95	95	95	95	95	95	95	95	95
PLTGU	MTWAR110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR130C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR11CC		200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	185	185
	MTWAR12CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMRELIAN DARI LUAR PLN																										
PLTP	GSLAK	#4	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182
		#5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CIKARANG		100	100	100	100	100	100	100	100	100	100	100	100	75	75	75	110	110	110	110	110	110	110	100	100	100
PLTU	Karakatau Steel		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1			4655	4553	4508	4453	4453	4453	4453	4453	4550	4550	4443	4343	4125	4125	4328	4493	4591	4591	4591	4591	4591	4591	4573	4533	4453
(**) Beban Area -1			4482	4436	4379	4323	4224	4180	4143	4105	4309	4228	4093	4093	4151	4207	4445	4718	4819	4780	4859	4888	4830	4885	4861	4708	4541
Selisih (*) - (**)			173	117	129	130	229	273	310	348	241	322	350	250	-26	-82	-117	-225	-228	-189	-268	-297	-239	-294	-288	-175	-88
Cadangan Seketika			54	108	104	121	121	121	121	121	70	70	130	125	172	172	126	123	50	50	50	50	50	50	50	70	75
Cadangan Putar			169	272	318	372	372	372	372	372	275	275	382	482	675	675	472	342	244	244	244	244	244	244	244	284	364

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL: RABU, 10 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

	Jam	13.00	13.30	14.00	14.30	15.00	15.30	16.00	16.30	17.00	17.30	18.00	18.30	19.00	19.30	20.00	20.30	21.00	21.30	22.00	22.30	23.00	23.30	24.00	Rata-2
PLTGU	MKRNG10C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG10C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG30C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG1CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG2CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3CC	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435	435
PLTU	MKRNG #4	165	165	165	165	165	165	165	165	165	175	175	175	175	175	175	165	165	165	165	165	165	165	165	165
	#5	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165
	MKRNG #1	95	95	95	95	95	95	95	95	95	101	101	103	103	103	103	95	95	95	95	95	95	95	95	95
	#2	95	95	95	95	95	95	95	95	95	101	101	103	103	103	103	95	95	95	95	95	95	95	95	95
	#3	95	95	95	95	95	95	95	95	95	101	101	103	103	103	103	95	95	95	95	95	95	95	95	95
PLTGU	MTWAR110C	0	0	0	0	0	0	0	72	72	72	72	72	72	72	72	0	0	0	0	0	0	0	0	0
	MTWAR10C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR20C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR120C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR130C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR11CC	200	200	200	255	255	275	275	285	285	300	300	320	320	320	320	300	300	300	300	285	285	277	277	250
	MTWAR12CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR13CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR GT 2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR GT 2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMBELIAN DARI LUAR PLN																								
PLTP	GSLAK #4	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182
	#5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CIKARANG	110	110	110	110	110	110	110	110	110	150	150	150	150	150	150	150	150	150	150	150	150	100	100	110
PLTU	Karakatau Steel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(*) Pembangkitan Area -1		4423	4443	4398	4398	4345	4345	4353	4358	4483	4603	5028	5028	5028	5028	5028	4953	4833	4823	4823	4816	4753	4703	4703	4582
(**) Beban Area -1		4637	4685	4656	4656	4604	4591	4562	4533	4702	5013	5100	5081	5104	5123	5118	5119	4918	4609	4688	4639	4496	4407	4325	4609
Selisih (*) - (**)		-214	-242	-258	-258	-259	-246	-209	-175	-219	-210	-72	-53	-76	-95	-90	-166	-85	14	135	177	255	296	378	-27
Cadangan Seketika		73	74	72	72	69	69	70	70	76	87	44	44	44	44	44	53	88	88	88	88	140	140	140	86
Cadangan Putar		404	384	429	429	599	599	591	591	544	329	114	114	114	114	114	189	309	319	319	319	382	382	382	346

SUB SISTEM REGION_4

RENCANA: HARI/TANGGAL: RABU, 10 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

	Jam	00.30	01.00	01.30	02.00	02.30	03.00	03.30	04.00	04.30	05.00	05.30	06.00	06.30	07.00	07.30	08.00	08.30	09.00	09.30	10.00	10.30	11.00	11.30	12.00	12.30
PLTA	Area 4	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
PLTA	SUTAMI	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
PLTA	BRANTAS	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	73	73	73	73	73	73	73	73	73	73
PLTU	PITON #1	370	370	370	370	370	370	370	370	370	370	325	325	325	325	370	370	372	372	372	372	372	372	350	350	350
	PITON #2	370	370	370	370	370	370	370	370	370	370	325	325	325	325	370	370	370	370	374	374	374	374	350	350	350
	GRSIK110C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK120C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK130C	87	87	85	85	78	78	76	76	85	85	103	103	101	101	84	84	86	86	102	102	103	103	102	102	102
	GRSIK11CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK12CC	285	285	276	276	268	268	258	258	250	250	250	250	250	250	300	300	300	300	300	300	300	300	300	300	300
	GRSIK13CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 2	GRSIK210C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK220C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK230C	87	87	85	85	91	91	92	92	89	89	84	84	85	85	86	86	89	89	85	85	96	96	97	97	98
	GRSIK21CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK22CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 3	GRSIK310C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK320C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK330C	89	89	87	87	86	86	91	91	95	95	94	94	97	97	92	92	93	93	93	93	97	97	96	96	95
	GRSIK31CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK32CC	285	285	286	286	265	265	252	252	250	250	250	250	250	250	275	275	300	300	300	300	300	300	300	300	275
	GRSIK33CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTU	GRSIK #3	90	90	90	90	90	90	90	90	90	90	90	90	90	90	135	135	165	165	165	165	165	165	165	165	165
	#4	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
	GRSIK #1	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75
	#2	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75
PLTG	GLTMR #1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK #1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1		3245	3245	3245	3245	3245	3245	3245	3426	3497	3497	3282	2974	2844	2844	3067	3272	3422	3522	3522	3522	3522	3522	3522	3482	3287
(**) Beban Area -1		2074	2065	2054	2052	2032	2032	2030	2031	2032	2030	2030	1757	1717	1665	1816	1892	1962	2082	2069	2030	2092	1996	2038	2073	2027
Selisih (*) - (**)		1171	1180	1191	1193	1213	1213	1215	1395	1465	1467	1252	1217	1127	1179	1251	1380	1460	1440	1453	1492	1430	1526	1484	1409	1260
Cadangan Seketika		85	85	100	106	108	106	106	106	72	72	67	93	92	92	98	59	61	61	61	61	61	61	61	67	63
Cadangan Putar		604	604	644	695	695	695	695	695	624	624	624	624	624	952	1054	854	704	604	604	604	604	604	604	644	839

SUB SISTEM REGION_4

RENCANA: HARI/TANGGAL: RABU, 10 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

			Jam	13.00	13.30	14.00	14.30	15.00	15.30	16.00	16.30	17.00	17.30	18.00	18.30	19.00	19.30	20.00	20.30	21.00	21.30	22.00	22.30	23.00	23.30	24.00	Rata-2
PLTA	Area 4			20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	23	21	21	21	21	21	21	20
PLTA	SUTAMI			90	90	90	90	90	90	90	90	100	100	100	100	100	100	100	100	90	90	90	90	90	90	90	92
PLTA	BRANTAS			93	93	93	93	93	93	93	93	116	116	116	116	116	116	116	116	116	116	86	86	86	86	86	86
PLTU	PITON	#1		350	375	375	370	370	370	370	370	370	370	370	370	370	370	370	370	350	350	350	350	350	350	350	370
	PITON	#2		350	372	372	370	370	370	370	370	370	370	370	370	370	370	370	370	350	350	350	350	350	350	350	370
	GRSIK110C			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C1			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C1			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK120C			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK130C			102	90	90	95	95	99	99	103	103	105	105	108	108	108	108	105	105	103	103	100	100	98	98	102
	GRSIK11CC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK12CC			300	275	275	265	265	265	265	275	275	285	285	300	300	300	300	285	285	285	285	275	275	275	275	285
	GRSIK13CC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 2	GRSIK210C			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C2			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C2			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK220C			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK230C			98	87	87	85	85	94	94	105	105	115	115	125	125	125	125	107	107	105	105	105	105	101	101	98
	GRSIK21CC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK22CC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23CC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 3	GRSIK310C			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C3			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C3			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK320C			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK330C			95	97	97	92	92	94	94	99	99	105	105	108	108	108	108	101	101	98	98	98	98	97	97	97
	GRSIK31CC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK32CC			275	275	275	290	290	300	300	300	300	300	300	300	300	285	285	285	285	277	277	273	273	273	273	300
	GRSIK33CC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTU	GRSIK	#3		165	165	165	165	165	165	165	165	165	165	165	165	165	165	165	155	155	155	155	134	134	130	130	165
		#4		90	90	90	110	110	140	140	155	155	165	165	165	165	165	165	165	165	165	165	165	165	165	165	165
	GRSIK	#1		75	75	75	75	75	75	75	75	75	85	85	85	85	85	85	75	75	75	75	75	75	75	75	75
		#2		75	75	75	75	75	75	75	75	75	85	85	85	85	85	85	75	75	75	75	75	75	75	75	75
PLTG	GLTMR	#1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK	#1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1				2869	2854	2952	2960	2965	2965	2965	2965	3250	4125	4185	4258	4589	4630	4338	4233	4098	3942	3727	3702	3562	3482	3342	3587
(**) Beban Area -1				1594	1621	1651	1658	1678	1680	1680	1680	1697	1693	1725	1752	2583	2881	2841	2747	2668	2525	2416	2246	2138	2162	2088	1965
Selisih (*) - (**)				1275	1233	1301	1302	1287	1285	1285	1285	1553	2427	2460	2506	2006	1749	1497	1486	1430	1417	1311	1456	1424	1320	1254	1622
Cadangan Seketika				85	82	82	82	82	85	87	90	90	95	96	98	96	98	75	78	98	95	90	52	41	32	30	75
Cadangan Putar				1065	1025	1065	1142	1142	1142	1142	685	80	80	80	82	82	82	285	345	358	365	719	802	805	764	804	#REFF

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL: SABTU, 13 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	00.30	01.00	01.30	02.00	02.30	03.00	03.30	04.00	04.30	05.00	05.30	06.00	06.30	07.00	07.30	08.00	08.30	09.00	09.30	10.00	10.30	11.00	11.30	12.00	12.30
PLTGU	MKRNG10C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG30C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG1CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG2CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3CC		420	420	450	450	422	422	423	423	410	410	415	415	350	350	375	375	436	436	436	436	436	438	422	422	421
PLTU	MKRNG	#4	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
		#5	150	150	150	150	150	150	150	150	165	165	150	150	150	150	155	155	167	167	165	165	165	165	167	167	167
	MKRNG	#1	75	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65
		#2	75	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65
		#3	75	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65
PLTGU	MTWAR110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR130C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR11CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR12CC		335	335	335	335	365	365	365	365	350	350	320	320	315	315	386	386	385	385	387	387	385	385	377	377	387
	MTWAR13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMBELIAN DARI LUAR PLN																										
PLTP	GSLAK	#4	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175
		#5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CIKARANG		50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
PLTU	Karakatau Steel		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1			4600	4550	4500	4453	4450	4453	4453	4453	4453	4453	4453	4453	4125	4125	4328	4493	4591	4591	4591	4591	4591	4591	4573	4533	4453
(**) Beban Area -1			4482	4436	4379	4323	4224	4180	4143	4105	4303	4228	4093	4093	4151	4207	4445	4718	4819	4780	4859	4883	4830	4885	4861	4708	4541
Selisih (*) - (**)			118	114	121	130	226	273	310	348	144	225	360	360	-26	-82	-117	-225	-228	-189	-268	-297	-239	-294	-288	-175	-88
Cadangan Seketika			52	100	104	121	121	121	121	121	70	70	130	125	172	172	126	123	50	50	50	50	50	50	50	70	75
Cadangan Puter			103	272	315	365	365	370	372	372	372	372	372	372	372	372	472	342	244	244	244	244	244	244	244	284	364

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL: SABTU, 13 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	13.00	13.30	14.00	14.30	15.00	15.30	16.00	16.30	17.00	17.30	18.00	18.30	19.00	19.30	20.00	20.30	21.00	21.30	22.00	22.30	23.00	23.30	24.00	Rata-2
PLTGU	MKRNG10C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG30C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG1CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG2CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3CC		421	432	432	325	325	325	325	350	350	445	445	445	445	435	435	415	415	410	410	400	400	385	385	420
PLTU	MKRNG	#4	90	90	90	90	90	90	90	90	90	135	150	150	150	150	150	150	150	150	150	150	150	150	150	107
		#5	167	165	165	150	150	150	150	150	150	170	170	170	170	170	170	162	162	162	162	155	155	155	155	155
	MKRNG	#1	65	65	65	65	65	65	65	65	65	90	90	90	90	90	90	65	65	65	65	65	65	65	65	65
		#2	65	65	65	65	65	65	65	65	65	90	90	90	90	90	90	65	65	65	65	65	65	65	65	65
		#3	65	65	65	65	65	65	65	65	65	90	90	90	90	90	90	65	65	65	65	65	65	65	65	65
PLTGU	MTWAR110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR130C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR11CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR12CC		387	385	385	315	315	325	325	345	345	390	390	390	390	390	390	365	365	362	362	320	320	320	320	385
	MTWAR13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMBELIAN DARI LUAR PLN																									
PLTP	GSLAK	#4	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175
		#5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CIKARANG		50	50	50	50	50	50	50	50	150	150	150	150	150	150	150	150	150	150	150	150	150	100	100	90
PLTU	Kerakatau Steel		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(*) Pembangkitan Area -1			4425	4432	4398	4392	4345	4345	4353	4358	4483	4678	5028	5028	5028	5028	5028	4953	4833	4823	4823	4813	4753	4700	4658	4582
(**) Beban Area -1			4637	4685	4656	4656	4604	4591	4562	4533	4702	5013	5100	5081	5104	5123	5118	5119	4918	4809	4688	4639	4498	4407	4325	4609
Selisih (*) - (**)			-212	-253	-258	-264	-259	-246	-209	-175	-219	-335	-72	-53	-76	-95	-90	-166	-85	14	135	174	255	293	333	-27
Cadangan Seketika			73	74	72	72	69	69	70	70	76	87	44	44	44	44	44	53	88	88	88	88	140	140	140	86
Cadangan Putar			404	384	429	429	599	599	591	591	544	329	114	114	114	114	114	189	309	319	319	319	382	382	382	346

SUB SISTEM REGION_4

RENCANA: HARI/TANGGAL: SABTU, 13 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	00.30	01.00	01.30	02.00	02.30	03.00	03.30	04.00	04.30	05.00	05.30	06.00	06.30	07.00	07.30	08.00	08.30	09.00	09.30	10.00	10.30	11.00	11.30	12.00	12.30
PLTA	Area 4		19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
PLTA	SUTAMI		90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
PLTA	BRANTAS		68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	73	73	73	73	73	73	73	73	73	73
PLTU	PITON	#1	370	370	350	350	350	350	350	350	370	370	370	316	316	316	350	360	360	360	360	360	360	360	360	360	360
	PITON	#2	370	370	350	350	350	350	350	350	370	370	370	316	316	316	350	360	360	360	360	360	360	360	360	360	360
	GRSIK110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK130C		85	85	78	78	78	78	77	77	87	87	102	102	100	100	85	85	90	90	103	103	105	105	110	110	112
	GRSIK11CC		155	155	155	155	131	131	148	148	155	155	180	180	180	180	161	161	177	177	175	175	181	181	182	182	182
	GRSIK12CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 2	GRSIK210C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK220C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK230C		87	87	91	91	90	90	92	92	87	87	83	83	85	85	85	85	87	87	93	93	95	95	102	102	103
	GRSIK21CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK22CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLTGU 3	GRSIK310C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C3		88	88	85	85	95	95	92	92	97	97	94	94	89	89	84	84	93	93	96	96	100	100	100	100	96
	GRSIK320C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK330C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK31CC		134	134	147	147	142	142	146	146	158	158	158	158	136	136	132	132	155	155	154	154	147	147	157	157	155
	GRSIK32CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK33CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTU	GRSIK	#3	120	120	122	122	121	121	121	121	121	121	121	121	121	121	121	121	121	121	120	120	121	121	121	121	121
		#4	121	121	120	120	120	120	121	121	120	120	120	120	121	121	121	121	120	120	120	120	120	120	121	121	121
	GRSIK	#1	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73
		#2	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73
PLTG	GLTMR	#1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK	#1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1			3304	3300	3258	3259	3287	3287	3287	3287	3287	3287	3282	2974	2844	2844	3087	3325	3422	3522	3522	3522	3522	3522	3522	3482	3287
(**) Beban Area -1			2066	2049	2022	1999	2093	2096	2077	2073	2075	2182	1963	1757	1717	1685	1816	1892	1962	2082	2069	2030	2092	1996	2038	2073	2027
Selisih (*) - (**)			1238	1251	1236	1260	1194	1191	1210	1214	1212	1105	1319	1217	1127	1179	1251	1433	1460	1440	1453	1492	1430	1526	1484	1409	1260
Cadangan Seketika			65	65	100	106	106	106	106	106	72	72	67	93	92	92	98	59	61	61	61	61	61	61	61	67	63
Cadangan Puter			604	604	644	695	695	695	695	695	624	624	839	1135	952	952	1054	854	704	604	604	743	743	743	744	745	839

SUB SISTEM REGION_4

RENCANA: HARI/TANGGAL: SABTU, 13 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	13.00	13.30	14.00	14.30	15.00	15.30	16.00	16.30	17.00	17.30	18.00	18.30	19.00	19.30	20.00	20.30	21.00	21.30	22.00	22.30	23.00	23.30	24.00	Rata-2
PLTA	Area 4		21	21	21	21	21	21	21	21	23	23	23	23	23	23	23	23	23	21	21	21	21	21	21	20
PLTA	SUTAMI		90	90	90	90	90	90	90	90	100	100	100	100	100	100	100	100	100	90	90	90	90	90	90	92
PLTA	BRANTAS		93	93	93	93	93	93	93	93	116	116	116	116	116	116	116	116	116	86	86	86	86	86	86	84
PLTU	PITON	#1	360	360	360	360	360	360	360	360	360	370	370	370	370	370	370	370	370	370	370	370	370	370	370	360
	PITON	#2	360	360	360	360	360	360	360	360	360	350	370	370	370	370	370	370	370	370	370	370	370	370	370	359
	GRSIK110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK130C		112	90	90	102	102	99	99	101	101	115	115	115	115	116	116	105	105	102	102	99	99	88	88	87
	GRSIK11CC		182	100	100	153	153	175	175	175	175	185	185	186	186	186	186	175	175	174	174	166	166	150	150	175
	GRSIK12CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 2	GRSIK210C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK220C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK230C		103	93	93	83	83	97	97	96	96	115	115	115	115	116	116	115	115	97	97	92	92	87	87	87
	GRSIK21CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK22CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 3	GRSIK310C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C3		96	87	87	89	89	96	96	98	98	105	105	106	106	105	105	97	97	87	87	85	85	85	85	87
	GRSIK320C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK330C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK31CC		155	156	156	126	126	157	157	157	157	162	162	163	163	162	162	156	156	156	156	146	146	135	135	155
	GRSIK32CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK33CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTU	GRSIK	#3	121	121	121	121	121	120	120	119	119	172	172	175	175	175	175	174	174	114	114	115	115	114	114	121
		#4	121	121	121	122	122	121	121	125	125	179	179	180	130	181	181	180	180	122	122	120	120	120	120	121
	GRSIK	#1	73	73	73	73	73	73	73	73	73	88	88	89	89	89	89	89	89	72	72	73	73	72	72	73
		#2	73	73	73	73	73	73	73	73	73	88	88	88	88	88	88	87	87	72	72	73	73	73	73	73
PLTG	GLTMR	#1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK	#1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1			3517	3510	3477	3426	3426	3426	3426	3426	3426	3426	4353	4353	4353	4353	4338	4233	4098	3942	3727	3702	3562	3482	3342	3543
(**) Beban Area -1			2050	2204	2210	2230	1858	1870	1894	1939	2199	2820	2887	2907	2897	2881	2841	2747	2668	2525	2416	2246	2138	2162	2088	2157
Selisih (*) - (**)			1467	1306	1267	1196	1568	1556	1532	1487	1227	603	1466	1446	1456	1472	1497	1486	1430	1417	1311	1456	1424	1320	1254	1386
Cadangan Seketika			65	63	62	62	106	106	110	112	115	68	37	37	37	37	44	62	56	75	69	68	66	70	68	69
Cadangan Putar			604	829	855	855	855	855	865	965	1035	552	227	227	227	227	242	347	482	567	719	744	784	764	804	#REFF

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL: MINGGU, 14 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	00.30	01.00	01.30	02.00	02.30	03.00	03.30	04.00	04.30	05.00	05.30	06.00	06.30	07.00	07.30	08.00	08.30	09.00	09.30	10.00	10.30	11.00	11.30	12.00	12.30
PLTGU	MKRNG10C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG30C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG1CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG2CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3CC		410	410	400	400	400	400	400	400	400	400	350	350	350	350	400	400	425	425	425	425	425	425	400	400	360
PLTU	MKRNG	#4	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
		#5	150	150	150	150	150	150	150	150	165	165	150	150	150	150	155	155	165	165	165	165	165	165	150	150	150
	MKRNG	#1	65	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
		#2	65	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
		#3	65	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
PLTGU	MTWAR110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR130C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR11CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR12CC		350	350	325	325	325	325	325	325	350	350	300	300	300	300	350	350	350	350	350	350	350	350	350	350	350
	MTWAR13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMBELIAN DARI LUAR PLN																										
PLTP	GSLAK	#4	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150
		#5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CIKARANG		100	100	100	100	100	100	100	100	100	100	100	100	75	75	75	110	110	110	110	110	110	110	100	100	100
PLTU	Karakatau Steel		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1			4520	4520	4587	4453	4453	4453	4453	4453	4550	4550	4443	4425	4125	4125	4328	4457	4591	4591	4591	4591	4591	4591	4573	4533	4453
(**) Beban Area -1			4482	4436	4379	4323	4224	4180	4143	4105	4309	4432	4093	4015	4120	4207	4445	4718	4819	4780	4859	4888	4830	4885	4861	4708	4541
Selisih (*) - (**)			38	84	208	130	229	273	310	348	241	118	350	410	5	-82	-117	-261	-228	-189	-268	-297	-239	-294	-288	-175	-88
Cadangan Seketika			54	106	104	121	121	121	121	121	70	70	130	125	172	172	126	123	50	50	50	50	50	50	50	70	75
Cadangan Putar			169	272	318	372	372	372	372	372	275	275	382	482	675	675	472	342	244	244	244	244	244	244	244	264	364

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL: MINGGU, 14 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	13.00	13.30	14.00	14.30	15.00	15.30	16.00	16.30	17.00	17.30	18.00	18.30	19.00	19.30	20.00	20.30	21.00	21.30	22.00	22.30	23.00	23.30	24.00	Rata-2
PLTGU	MKRNG10C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG20C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG30C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG1CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG2CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3CC		360	360	360	350	350	350	350	350	350	425	425	425	425	425	425	400	400	400	400	400	400	400	400	350
PLTU	MKRNG	#4	90	90	90	90	90	90	90	90	90	150	150	150	150	150	150	150	150	150	150	150	150	150	150	90
		#5	150	150	150	150	150	150	150	150	150	165	165	165	165	165	165	160	160	150	150	150	150	150	150	150
	MKRNG	#1	50	50	50	50	50	50	50	50	50	85	85	85	85	85	85	50	50	50	50	50	50	50	50	50
		#2	50	50	50	50	50	50	50	50	50	85	85	85	85	85	85	50	50	50	50	50	50	50	50	50
		#3	50	50	50	50	50	50	50	50	50	85	85	85	85	85	85	50	50	50	50	50	50	50	50	50
PLTGU	MTWAR110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR130C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR11CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR12CC		350	325	325	292	292	300	300	300	300	390	390	390	390	390	390	350	350	350	350	350	350	350	350	325
	MTWAR13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR	GT 2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMBELIAN DARI LUAR PLN																									
PLTP	GSLAK	#4	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150	150
		#5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CIKARANG		110	110	110	110	110	110	110	110	110	150	150	150	150	150	150	150	150	150	150	150	150	150	100	114
PLTU	Karakatau Steel		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1			3845	3957	3968	3968	3968	3968	3968	3968	3968	4520	4562	4582	4952	4952	4936	4862	4752	4689	4653	4628	4369	4258	4256	4472
(**) Beban Area -1			4052	4033	4656	4656	4604	4591	4562	4533	4652	4652	4753	4750	4750	4725	4732	4720	4918	4809	4688	4639	4498	4407	4325	4378
Selisih (*) - (**)			-207	-76	-688	-688	-636	-623	-594	-565	-684	-132	-191	-168	202	227	204	142	-166	-120	-35	-11	-129	-149	-69	94
Cadangan Seketika			73	74	72	72	69	69	70	70	76	37	44	44	44	44	44	53	88	88	88	88	140	140	140	86
Cadangan Putar			404	384	429	429	599	599	591	591	544	329	114	114	114	114	114	189	309	319	319	319	382	382	382	346

SUB SISTEM REGION_4

RENCANA: HARI/TANGGAL: MINGGU, 14 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

	Jam	00.30	01.00	01.30	02.00	02.30	03.00	03.30	04.00	04.30	05.00	05.30	06.00	06.30	07.00	07.30	08.00	08.30	09.00	09.30	10.00	10.30	11.00	11.30	12.00	12.30
PLTA	Area 4	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
PLTA	SUTAMI	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
PLTA	BRANTAS	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	73	73	73	73	73	73	73	73	73	73
PLTU	PITON #1	370	370	350	350	350	350	350	350	370	370	370	316	316	316	350	360	360	360	360	360	360	360	360	360	360
	PITON #2	370	370	350	350	350	350	350	350	370	370	370	316	316	316	350	360	360	360	360	360	360	360	360	360	360
	GRSIK110C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK120C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK130C	60	60	73	73	73	73	72	72	82	82	97	97	95	95	80	60	81	81	87	87	100	100	105	105	103
	GRSIK11CC	140	140	130	130	126	126	143	143	150	150	175	175	175	175	156	156	156	156	156	156	176	176	148	148	150
	GRSIK12CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK13CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 2	GRSIK210C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK220C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK230C	88	88	90	90	90	90	92	92	85	85	82	82	86	86	81	81	85	85	82	82	94	94	97	97	97
	GRSIK21CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK22CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 3	GRSIK310C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C3	89	89	88	88	94	94	91	91	99	99	92	92	89	89	82	82	91	91	93	93	86	86	93	93	93
	GRSIK320C	129	129	142	142	137	137	141	141	153	153	153	153	131	131	127	127	134	134	143	143	142	142	146	146	146
	GRSIK330C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK31CC	134	134	147	147	142	142	146	146	158	158	158	158	136	136	132	132	155	155	154	154	147	147	157	157	155
	GRSIK32CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK33CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTU	GRSIK #3	165	165	150	150	150	150	150	150	150	150	150	150	150	150	165	165	165	165	165	165	165	165	150	150	150
	#4	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
	GRSIK #1	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
	#2	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
PLTG	GLTMR #1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK #1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1		3517	3517	3477	3426	3426	3426	3426	3426	3497	3497	3282	2974	2844	2844	3067	3272	3422	3522	3522	3522	3522	3522	3522	3482	3287
(**) Beban Area -1		2066	2049	2022	1999	2093	2096	2077	2073	2075	2182	1963	1757	1717	1665	1816	1892	1962	2082	2069	2030	2092	1996	2038	2073	2027
Selisih (*) - (**)		1451	1468	1455	1427	1333	1330	1349	1353	1422	1315	1319	1217	1127	1179	1251	1380	1460	1440	1453	1492	1430	1526	1484	1409	1260
Cadangan Seketika		54	56	95	106	108	108	108	106	72	72	65	93	92	92	98	59	61	61	61	61	61	61	61	67	63
Cadangan Putar		523	600	620	695	695	695	695	695	624	624	758	1147	952	952	1054	854	704	604	604	604	604	604	604	644	839

SUB SISTEM REGION_4

RENCANA: HARI/TANGGAL: MINGGU, 14 MARET 2004

PT. PLN PEMBANGKITAN TENAGA LISTRIK JAWA-BALI

		Jam	13.00	13.30	14.00	14.30	15.00	15.30	16.00	16.30	17.00	17.30	18.00	18.30	19.00	19.30	20.00	20.30	21.00	21.30	22.00	22.30	23.00	23.30	24.00	Rata-2
PLTA	Area 4		21	21	21	21	21	21	21	21	23	23	23	23	23	23	23	23	23	21	21	21	21	21	21	20
PLTA	SUTAMI		90	90	90	90	90	90	90	90	100	100	100	100	100	100	100	100	100	90	90	90	90	90	90	92
PLTA	BRANTAS		93	93	93	93	93	93	93	93	116	116	116	116	116	116	116	116	116	86	86	86	86	86	86	84
PLTU	PITON	#1	360	360	360	360	360	360	360	360	360	370	370	370	370	370	370	370	370	370	370	370	370	370	370	370
	PITON	#2	360	360	360	360	360	360	360	360	360	350	370	370	370	370	370	370	370	370	370	370	370	370	370	370
	GRSIK110C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK120C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK130C		103	85	85	87	87	94	94	96	96	100	100	102	102	102	102	100	100	97	97	94	94	83	83	87
	GRSIK11CC		150	141	141	143	143	170	170	172	172	176	176	176	176	175	175	175	175	176	176	162	162	152	152	175
	GRSIK12CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK13CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 2	GRSIK210C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK220C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK230C		82	82	82	81	81	97	97	98	98	117	117	116	116	117	117	115	115	99	99	95	95	82	82	97
	GRSIK21CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK22CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTGU 3	GRSIK310C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK10C3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK20C3		93	87	87	89	89	96	96	98	98	103	103	102	102	103	103	95	95	82	82	87	87	83	83	87
	GRSIK320C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK330C		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK31CC		146	127	127	121	121	152	152	150	150	134	134	147	147	162	162	142	142	127	127	131	131	132	132	152
	GRSIK32CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK33CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTU	GRSIK	#3	150	150	150	150	150	150	150	150	150	165	165	165	165	165	165	165	165	165	165	165	165	150	150	150
		#4	90	90	90	90	90	90	90	90	90	165	165	165	165	150	150	140	140	90	90	90	90	90	90	90
	GRSIK	#1	50	50	50	50	50	50	50	50	50	85	85	85	85	85	85	50	50	50	50	50	50	50	50	50
		#2	50	50	50	50	50	50	50	50	50	85	85	85	85	85	85	50	50	50	50	50	50	50	50	50
PLTG	GLTMR	#1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK	#1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(*) Pembangkitan Area -1			3309	3325	3310	3313	3313	3313	3303	3303	3482	4054	4353	4353	4353	4353	4338	4233	4098	3942	3727	3702	3562	3482	3342	3543
(**) Beban Area -1			1936	1950	1912	1825	1858	1870	1894	1939	2199	2820	2887	2907	2897	2881	2841	2747	2668	2525	2416	2246	2138	2162	2088	2157
Selisih (*) - (**)			1373	1375	1398	1488	1455	1443	1409	1364	1283	1234	1466	1446	1456	1472	1497	1486	1430	1417	1311	1456	1424	1320	1254	1386
Cadangan Seketika			63	63	62	62	62	62	61	61	68	68	37	37	37	37	44	62	56	75	69	68	68	70	68	69
Cadangan Putar			853	820	825	825	825	825	825	825	935	952	257	257	257	257	245	347	482	567	719	744	784	764	705	#REFF

Lampiran 1. Listing Program GASA

```
unit Utama;

interface

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

type
  TfrmUtama = class(TForm)
    pnlbtn: TPanel;
    btnNew: TButton;
    btnOpen: TButton;
    btnPrint: TButton;
    btnExit: TButton;
    OpenDialog1: TOpenDialog;
    Panel1: TPanel;
    StatusBar1: TStatusBar;
    Image1: TImage;
    procedure btnExitClick(Sender: TObject);
    procedure btnNewClick(Sender: TObject);
    procedure btnOpenClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmUtama: TfrmUtama;

implementation

uses Input, Tampil, UnitGenerator, Fitness, IGA;

{$R *.dfm}

procedure TfrmUtama.btnExitClick(Sender: TObject);
begin
  try
    gasa.Free;
    FucFitness.Free;
  finally
    Application.Terminate;
  end;
end;

procedure TfrmUtama.btnNewClick(Sender: TObject);
```

```
begin
  frmInput.Show;
end;

procedure TfrmUtama.btnOpenClick(Sender: TObject);
var Output:TextFile;
    Pmax,Pmin,a0,a1,a2,Sh,Sc,Load1,Res1:double;
    tup,tdown,tcold,iniSt,Ngen1,Njam1:integer;
    Nama,NamaFile:string;
    gen:TGenArr;
    i,j:byte;
begin
  FucFitness:=TFitness.Create;
  gasa:=TIGA.Create;
  if OpenDialog1.Execute then
  begin
    NamaFile:=OpenDialog1.FileName;
    AssignFile(output,Namafile);
    Reset(Output);
    Readln(output,Ngen1);
    Readln(output,Njam1);
    FucFitness.fitNgen:=Ngen1;
    FucFitness.fitNjam:=Njam1;
    gasa.gaNparam:=Ngen1;
    gasa.gaLchrom:=Njam1;
    SetLength(Beban,Njam1+1);
    SetLength(Res,Njam1+1);
    SetLength(PLN,Ngen1+1,Njam1+1);
    SetLength(gen,Ngen1+1);
    for i:=1 to FucFitness.fitNgen do
    begin
      Readln(output,Pmax,Pmin,a0,a1,a2,tup,tdown,Sh,Sc,tcold,
      iniSt,Nama);
      gen[i]:=TPembangkit.Create;
      gen[i].GenNama:=Nama;
      gen[i].GenPmax:=Pmax;
      gen[i].GenPmin:=Pmin;
      gen[i].Gena0:=a0;
      gen[i].Gena1:=a1;
      gen[i].Gena2:=a2;
      gen[i].GenTup:=tup;
      gen[i].GenTdown:=tdown;
      gen[i].GenSh:=Sh;
      gen[i].GenSc:=Sc;
      gen[i].GenTcold:=tcold;
      gen[i].GenIniSt:=iniSt;
    end;
    FucFitness.fitGen:=gen;
    for i:=1 to FucFitness.fitNjam do
    begin
```

```
    Readln(Output,Load1,Res1);
    Beban[i]:=Load1;
    Res[i]:=Res1;
   end;
   FucFitness.fitBeban:=Beban;
   FucFitness.fitRes:=Res;
   for i:=1 to FucFitness.fitNgen do
   begin
    gen[i].Free;
    for j:=1 to FucFitness.fitNjam do
    begin
     Read(Output,PLN[i,j]);
    end;
    Readln(output);
   end;
   CloseFile(output);
   frmTampil.show;
  end;
end;

end.


unit Input;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, AxCtrls, OleCtrls, VCF1, ComCtrls;

type
  TfrmInput = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    lblSumUnit: TLabel;
    lblSumJam: TLabel;
    edtUnit: TEdit;
    edtJam: TEdit;
    TabSheet2: TTabSheet;
    fgUnit: TF1Book;
    TabSheet3: TTabSheet;
    fgBeban: TF1Book;
    TabSheet4: TTabSheet;
    fgPLN: TF1Book;
    Panel1: TPanel;
    btnClose: TButton;
    btnSave: TButton;
    SaveDialog1: TSaveDialog;
    procedure btnCloseClick(Sender: TObject);
```

```
    procedure btnSaveClick(Sender: TObject);
    procedure edtUnitChange(Sender: TObject);
    procedure edtJamChange(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmInput: TfrmInput;

implementation

{$R *.dfm}

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

procedure TfrmInput.btnSaveClick(Sender: TObject);
var NamaFile,Nama:string;
    Input:textfile;
    Pmax,Pmin,a0,a1,a2,Sh,Sc,Load,Reserv,PLN:double;
    i,j,Nunit,Njam,tup,tdown,tcold,initial,No:integer;
begin
  if SaveDialog1.Execute then
  begin
    try
      NamaFile:=SaveDialog1.FileName;
      AssignFile(Input,NamaFile+'.txt');
      Rewrite(Input);
      Nunit:=StrToInt(edtUnit.Text);
      Njam:=StrToInt(edtJam.Text);
      Writeln(input,Nunit);
      Writeln(input,Njam);
      for i:=1 to Nunit do
      begin
        No:=trunc(fgUnit.NumberRC[i+2,1]);
        Nama:=fgUnit.TextRC[i+2,2];
        Pmax:=fgUnit.NumberRC[i+2,3];
        Pmin:=fgUnit.NumberRC[i+2,4];
        a0:=fgUnit.NumberRC[i+2,5];
        a1:=fgUnit.NumberRC[i+2,6];
        a2:=fgUnit.NumberRC[i+2,7];
        tup:=trunc(fgUnit.NumberRC[i+2,8]);
        tdown:=trunc(fgUnit.NumberRC[i+2,9]);
        Sh:=fgUnit.NumberRC[i+2,10];
```

```
      Sc:=fgUnit.NumberRC[i+2,11];
      tcold:=trunc(fgUnit.NumberRC[i+2,12]);
      initial:=trunc(fgUnit.NumberRC[i+2,13]);
      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,' ',initial,' ',Nama);
    end;
    for i:=1 to Njam do
    begin
      Load:=fgBeban.NumberRC[2,i+1];
      Reserv:=fgBeban.NumberRC[3,i+1];
      Writeln(Input,Load:7:0,' ',Reserv:7:0);
    end;
    for i:=1 to Nunit do
    begin
      for j:=1 to Njam do
      begin
        PLN:=fgPLN.NumberRC[i+1,j+1];
        Write(Input,PLN:7:2,' ');
      end;
      Writeln(Input,'');
    end;
    CloseFile(input);
  except
    Application.MessageBox('Tidak jadi Simpan File','Peringatan',
    MB_OK);
  end;
 end;
end;

procedure TfrmInput.edtUnitChange(Sender: TObject);
var i:byte;
begin
 if edtUnit.Text='' then
 begin
   fgUnit.MaxRow:=3;
   fgUnit.NumberRC[3,1]:=1;
   fgPLN.MaxRow:=2;
 end
 else
 begin
   fgUnit.MaxRow:=StrToInt(edtUnit.Text)+2;
   fgPLN.MaxRow:=StrToInt(edtUnit.Text)+1;
   for i:=1 to StrToInt(edtUnit.Text) do
   begin
     fgUnit.NumberRC[i+2,1]:=i;
     fgPLN.NumberRC[i+1,1]:=i;
   end;
 end;
end;
```

```
procedure TfrmInput.edtJamChange(Sender: TObject);
var i:byte;
begin
 if edtJam.Text='' then
 begin
  fgBeban.MaxCol:=2;
  fgBeban.NumberRC[1,2]:=1;
  fgPLN.MaxCol:=2;
 end
 else
 begin
  fgBeban.MaxCol:=StrToInt(edtJam.Text)+1;
  fgPLN.MaxCol:=StrToInt(edtJam.Text)+1;
  for i:=1 to StrToInt(edtJam.Text) do
  begin
   fgBeban.NumberRC[1,i+1]:=i;
   fgPLN.NumberRC[1,i+1]:=i;
  end
 end;
end;

procedure TfrmInput.FormActivate(Sender: TObject);
begin
 edtUnit.SetFocus;
end;

end.


unit GeneticAlgorithm;

interface

uses Komplex;

type
 TGA=class
 private
  maxgen,popsize,lchrom,Nparam:integer;
  pcross,pmutat,pflip,ka:double;
  function GetMaxgen:integer;
  function GetPopsize:integer;
  function GetLchrom:integer;
  function GetNparam:integer;
  function GetPcross:double;
  function GetPmutat:double;
  function GetPflip:double;
  function GetKa:double;
  procedure SetMaxgen(dMaxgen:integer);
```

```
    procedure SetPopsize(dPopsize:integer);
    procedure SetLchrom(dLchrom:integer);
    procedure SetNparam(dNparam:integer);
    procedure SetPcross(dPcross:double);
    procedure SetPmutat(dPmutat:double);
    procedure SetPflip(dPflip:double);
    procedure SetKa(dKa:double);
  public
    constructor Create;
    function GetFlip(const param:double):boolean;
    function GetRandom(const min,max:integer):integer;
    destructor Destroy;override;
    property gaMaxgen:integer read GetMaxgen write SetMaxgen;
    property gaPopsize:integer read GetPopsize write SetPopsize;
    property gaLchrom:integer read GetLchrom write SetLchrom;
    property gaNparam:integer read GetNparam write SetNparam;
    property gaPcross:double read GetPcross write SetPcross;
    property gaPmutat:double read getPmutat write SetPmutat;
    property gaPflip:double read GetPflip write SetPflip;
    property gaKa:double read GetKa write SetKa;
  end;

implementation

//constructor
constructor TGA.Create;
begin
  inherited Create;
  maxgen:=1;
  popsize:=1;
  Nparam:=1;
  ka:=1;
end;

//data accessing
function TGA.GetMaxgen:integer;
begin
  result:=maxgen;
end;

function TGA.GetPopsize:integer;
begin
  result:=popsize;
end;

function TGA.GetLchrom:integer;
begin
  result:=lchrom;
end;
```

```
function TGA.GetNparam:integer;
begin
  result:=Nparam;
end;

function TGA.GetPcross:double;
begin
  result:=pcross;
end;

function TGA.GetPmutat:double;
begin
  result:=pmutat;
end;

function TGA.GetPflip:double;
begin
  result:=pflip;
end;

function TGA.GetKa:double;
begin
  result:=ka;
end;

procedure TGA.SetMaxgen(dMaxgen:integer);
begin
  maxgen:=dMaxgen;
end;

procedure TGA.SetPopsize(dPopsize:integer);
begin
  popsize:=dpopsize;
end;

procedure TGA.SetLchrom(dLchrom:integer);
begin
  lchrom:=dLchrom;
end;

procedure TGA.SetNparam(dNparam:integer);
begin
  Nparam:=dNparam;
end;

procedure TGA.SetPcross(dPcross:double);
begin
  pcross:=dPcross;
end;
```

```
procedure TGA.SetPmutat(dPmutat:double);
begin
  pmutat:=dPmutat;
end;

procedure TGA.SetPflip(dPflip:double);
begin
  pflip:=dPflip;
end;

procedure TGA.SetKa(dKa:double);
begin
  ka:=dka;
end;

//data processing
function TGA.GetFlip(const param:double):boolean;
var rand:double;
begin
  rand:=random;
  if rand<=param then
  begin
    result:=true;
  end
  else
  begin
    result:=false;
  end;
end;

function TGA.GetRandom(const min,max:integer):integer;
var rand:double;
begin
  rand:=random;
  result:=round(min+rand*(max-min));
end;

destructor TGA.Destroy;
begin
  inherited Destroy;
end;

end.


unit EcoDispath;

interface

uses Komplex,UnitGenerator,TypDatGA;
```

```
type
 TEcoDis=class
 private
  Ngen:integer;
  Beban:double;
  chrom:TChromBin1;
  gen:TGenArr;
  PL:Arr1;
  function GetNgen:integer;
  function GetBeban:double;
  function GetChrom:TChromBin1;
  function GetGen:TGenArr;
  function GetPL:Arr1;
  procedure SetNgen(const dNgen:integer);
  procedure SetBeban(const dBeban:double);
  procedure SetChrom(const dChrom:TChromBin1);
  procedure SetGen(const dGen:TGenArr);
  procedure doHitung;
 public
  constructor Create;overload;
  constructor Create(const dNgen:integer;const dBeban:double);overload;
  function IsServe:byte;
  destructor Destroy;override;
  property ecoNgen:integer read GetNgen write SetNgen;
  property ecoBeban:double read GetBeban write SetBeban;
  property ecoChrom:TChromBin1 read GetChrom write SetChrom;
  property ecoGen:TGenArr read GetGen write SetGen;
  property ecoPL:Arr1 read GetPL;
 end;

implementation

//constructors
constructor TEcoDis.Create;
begin
 inherited Create;
 Ngen:=0;
 Beban:=0;
end;

constructor TEcoDis.Create(const dNgen:integer;const dBeban:double);
begin
 inherited Create;
 Ngen:=dNgen;
 Beban:=dBeban;
end;

//data accessing
function TEcoDis.GetNgen:integer;
```

```
begin
  result:=Ngen;
end;

function TEcoDis.GetBeban:double;
begin
  result:=Beban;
end;

function TEcoDis.GetChrom:TChromBin1;
var i:integer;
begin
  SetLength(result,Ngen+1);
  for i:=1 to Ngen do
  begin
    result[i]:=chrom[i];
  end;
end;

function TEcoDis.GetGen:TGenArr;
var i:integer;
begin
  SetLength(result,Ngen+1);
  for i:=1 to Ngen do
  begin
    result[i]:=TPembangkit.Create;
    result[i].GenNama:=Gen[i].GenNama;
    result[i].GenPmax:=Gen[i].GenPmax;
    result[i].GenPmin:=Gen[i].GenPmin;
    result[i].Gena2:=Gen[i].Gena2;
    result[i].Gena1:=Gen[i].Gena1;
    result[i].Gena0:=Gen[i].Gena0;
    result[i].GenSh:=Gen[i].GenSh;
    result[i].GenSc:=Gen[i].GenSc;
    result[i].GenTup:=Gen[i].GenTup;
    result[i].GenTdown:=Gen[i].GenTdown;
    result[i].GenTcold:=Gen[i].GenTcold;
    result[i].GenIniSt:=Gen[i].GenIniSt;
  end;
end;

procedure TEcoDis.SetNgen(const dNgen:integer);
begin
  Ngen:=dNgen;
end;

procedure TEcoDis.SetBeban(const dBeban:double);
begin
  Beban:=dBeban;
end;
```

```
procedure TEcoDis.SetChrom(const dChrom:TChromBin1);
var i:integer;
begin
  SetLength(chrom,Ngen+1);
  for i:=1 to Ngen do
  begin
    chrom[i]:=dChrom[i];
  end;
end;

procedure TEcoDis.SetGen(const dGen:TGenArr);
var i:integer;
begin
  SetLength(Gen,Ngen+1);
  for i:=1 to Ngen do
  begin
    Gen[i]:=TPembangkit.Create;
    Gen[i].GenNama:=dGen[i].GenNama;
    Gen[i].GenPmax:=dGen[i].GenPmax;
    Gen[i].GenPmin:=dGen[i].GenPmin;
    Gen[i].Gena2:=dGen[i].Gena2;
    Gen[i].Gena1:=dGen[i].Gena1;
    Gen[i].Gena0:=dGen[i].Gena0;
    Gen[i].GenSh:=dGen[i].GenSh;
    Gen[i].GenSc:=dGen[i].GenSc;
    Gen[i].GenTup:=dGen[i].GenTup;
    Gen[i].GenTdown:=dGen[i].GenTdown;
    Gen[i].GenTcold:=dGen[i].GenTcold;
    Gen[i].GenIniSt:=dGen[i].GenIniSt;
  end;
end;

//data processing
function TEcoDis.IsServe:byte;
var i:integer;
    sPmin,sPmax:double;
begin
  result:=3;
  sPmin:=0;
  sPmax:=0;
  for i:=1 to Ngen do
  begin
    if chrom[i]=true then
    begin
      sPmin:=sPmin+Gen[i].GenPmin;
      sPmax:=sPmax+Gen[i].GenPmax;
    end;
  end;
  if Beban<sPmin then
```

```
  begin
    result:=1;
  end
  else if Beban>sPmax then
  begin
    result:=2;
  end;
end;

procedure TEcoDis.doHitung;
var i,j:integer;
    Status:TChromBin1;
    LoadCek,Pa,Pb,Lmd,LoadSplit:double;
    diffa2,diffa1,Cek,tes:double;
begin
  SetLength(Status,Ngen+1);
  for i:=1 to Ngen do
  begin
    Status[i]:=Chrom[i];
    Gen[i].GenDaya:=0;
  end;
  LoadCek:=Beban;
  LoadSplit:=Beban;
  for i:=1 to 15 do
  begin
    Pa:=0;
    Pb:=0;
    for j:=1 to Ngen do
    begin
      if Status[j] then
      begin
        diffa2:=Gen[j].Gena2*2;
        diffa1:=Gen[j].Gena1;
        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 Ngen do
    begin
      if Status[j] then
      begin
```

```
    diffa2:=2*Gen[j].Gena2;
    diffa1:=Gen[j].Gena1;
    Gen[j].GenDaya:=(Lmd-diffa1)/diffa2;
    if Gen[j].GenDaya<Gen[j].GenPMin then
    begin
      Gen[j].GenDaya:=Gen[j].GenPMin;
    end;
    if Gen[j].GenDaya>Gen[j].GenPMax then
    begin
      Gen[j].GenDaya:=Gen[j].GenPMax;
    end;
  end;
  Cek:=Cek+Gen[j].GenDaya;
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 Ngen do
  begin
    if Status[j] then
    begin
      if Gen[j].GenDaya=Gen[j].GenPMax then
      begin
        Status[j]:=false;
        LoadSplit:=LoadSplit-Gen[j].GenDaya;
        if LoadSplit<0 then
        begin
          LoadSplit:=LoadSplit+Gen[j].GenDaya;
          Status[j]:=true;
        end;
      end;
    end;
  end;
end
else if tes<0 then
begin
  for j:=1 to Ngen do
  begin
    if Status[j] then
    begin
      if Gen[j].GenDaya=Gen[j].GenPMin then
      begin
        Status[j]:=false;
        LoadSplit:=LoadSplit-Gen[j].GenDaya;
        if LoadSplit<0 then
        begin
```

```
          LoadSplit:=LoadSplit+Gen[j].GenDaya;
          Status[j]:=true;
         end;
        end;
       end;
     end;
   end;
  end;
  SetLength(PL,Ngen+1);
  for i:=1 to Ngen do
  begin
   PL[i]:=0;
   if chrom[i] then
   begin
    PL[i]:=Gen[i].GenDaya;
   end;
  end;
end;

//data output
function TEcoDis.GetPL:Arr1;
var i:integer;
   serve:byte;
begin
  serve:=isServe;
  SetLength(PL,Ngen+1);
  if serve=1 then
  begin
   for i:=1 to Ngen do
   begin
    if chrom[i]=true then
    begin
     PL[i]:=Gen[i].GenPmin;
    end;
   end;
  end
  else if serve=2 then
  begin
   for i:=1 to Ngen do
   begin
    if chrom[i]=true then
    begin
     PL[i]:=Gen[i].GenPmax;
    end;
   end;
  end
  else if serve=3 then
  begin
   doHitung;
  end;
```

```
 SetLength(result,Ngen+1);
 for i:=1 to Ngen do
 begin
  result[i]:=PL[i];
 end;
end;

//destroying object
destructor TEcoDis.Destroy;
var i:integer;
begin
 try
  for i:=1 to Ngen do
  begin
   Gen[i].Free;
  end;
 finally
  inherited Destroy;
 end;
end;

end.


unit UnitGenerator;

interface

uses Komplex;

type
 TPembangkit=class
 private
  Nama:string;
  Pmax,Pmin,a2,a1,a0,Sh,Sc,AFLC,Daya:double;
  tup,tdown,tcold,IniSt:integer;
  function GetNama:string;
  function GetPmax:double;
  function GetPmin:double;
  function Geta2:double;
  function Geta1:double;
  function Geta0:double;
  function GetSh:double;
  function GetSc:double;
  function GetTup:integer;
  function GetTdown:integer;
  function GetTcold:integer;
  function GetIniSt:integer;
  procedure SetNama(const dNama:string);
  procedure SetPmax(const dPmax:double);
```

```
  procedure SetPmin(const dPmin:double);
  procedure Seta2(const da2:double);
  procedure Seta1(const da1:double);
  procedure Seta0(const da0:double);
  procedure SetSh(const dSh:double);
  procedure SetSc(const dSc:double);
  procedure SetTup(const dtup:integer);
  procedure SetTdown(const dtdown:integer);
  procedure SetTcold(const dtcold:integer);
  procedure SetIniSt(const dIniSt:integer);
  procedure SetDaya(const dDaya:double);
  procedure HitungAFLC;
  function GetAFLC:double;
  function GetDaya:double;
 public
  function GetBiaya(const dDaya:double):double;
  property GenNama:string read GetNama write SetNama;
  property GenPmax:double read GetPmax write SetPmax;
  property GenPmin:double read GetPmin write SetPmin;
  property Gena2:double read Geta2 write Seta2;
  property Gena1:double read Geta1 write Seta1;
  property Gena0:double read Geta0 write Seta0;
  property GenSh:double read GetSh write SetSh;
  property GenSc:double read GetSc write SetSc;
  property GenTup:integer read GetTup write SetTup;
  property GenTdown:integer read GetTdown write SetTdown;
  property GenTcold:integer read GetTcold write SetTcold;
  property GenIniSt:integer read GetIniSt write SetIniSt;
  property GenDaya:double read GetDaya write SetDaya;
  property GenAFLC:double read GetAFLC;
 end;

 TGenArr=array of TPembangkit;

implementation

function TPembangkit.GetNama:string;
begin
  Result:=Nama;
end;

function TPembangkit.GetPmax:double;
begin
  Result:=Pmax;
end;

function TPembangkit.GetPmin:double;
begin
  Result:=Pmin;
end;
```

```
function TPembangkit.Geta2:double;
begin
  Result:=a2;
end;

function TPembangkit.Geta1:double;
begin
  Result:=a1;
end;

function TPembangkit.Geta0:double;
begin
  Result:=a0;
end;

function TPembangkit.GetSh:double;
begin
  Result:=Sh;
end;

function TPembangkit.GetSc:double;
begin
  Result:=Sc;
end;

function TPembangkit.GetTup:integer;
begin
  Result:=tup;
end;

function TPembangkit.GetTdown:integer;
begin
  Result:=tdown;
end;

function TPembangkit.GetTcold:integer;
begin
  Result:=tcold;
end;

function TPembangkit.GetIniSt:integer;
begin
  Result:=IniSt;
end;

function TPembangkit.GetBiaya(const dDaya:double):double;
begin
  Result:=0;
  if dDaya<>0 then
```

```
 begin
   Result:=a2*sqr(dDaya)+a1*dDaya+a0;
 end;
end;

function TPembangkit.GetAFLC:double;
begin
 HitungAFLC;
 Result:=AFLC;
end;

function TPembangkit.GetDaya:double;
begin
 Result:=Daya;
end;

procedure TPembangkit.SetNama(const dNama:string);
begin
 Nama:=dNama;
end;

procedure TPembangkit.SetPmax(const dPmax:double);
begin
 Pmax:=dPmax;
end;

procedure TPembangkit.SetPmin(const dPmin:double);
begin
 Pmin:=dPmin;
end;

procedure TPembangkit.Seta2(const da2:double);
begin
 a2:=da2;
end;

procedure TPembangkit.Seta1(const da1:double);
begin
 a1:=da1;
end;

procedure TPembangkit.Seta0(const da0:double);
begin
 a0:=da0;
end;

procedure TPembangkit.SetSh(const dSh:double);
begin
 Sh:=dSh
end;
```

```
procedure TPembangkit.SetSc(const dSc:double);
begin
  Sc:=dSc;
end;

procedure TPembangkit.SetTup(const dtup:integer);
begin
  tup:=dtup;
end;

procedure TPembangkit.SetTdown(const dtdown:integer);
begin
  tdown:=dtdown;
end;

procedure TPembangkit.SetTcold(const dtcold:integer);
begin
  tcold:=dtcold;
end;

procedure TPembangkit.SetIniSt(const diniSt:integer);
begin
  iniSt:=diniSt;
end;

procedure TPembangkit.SetDaya(const dDaya:double);
begin
  Daya:=dDaya;
end;

procedure TPembangkit.HitungAFLC;
begin
  AFLC:=a0/Pmax+a1+a2*Pmax;
end;

end.


unit Param;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TfrmParam = class(TForm)
    gbxParam: TGroupBox;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    edtIterasi: TEdit;
    edtR: TEdit;
    edtTheta: TEdit;
    edtT0: TEdit;
    edtps: TEdit;
    edtpr: TEdit;
    edtflip: TEdit;
    edtAlpha1: TEdit;
    edtAlpha2: TEdit;
    edtPopSize: TEdit;
    edtMaxGen: TEdit;
    edtpcross: TEdit;
    edtpmutat: TEdit;
    edtKonstM: TEdit;
    btnNext: TButton;
    btnDefault: TButton;
    btnClose: TButton;
    procedure btnCloseClick(Sender: TObject);
    procedure btnDefaultClick(Sender: TObject);
    procedure btnNextClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmParam: TfrmParam;

implementation

uses IGA, Fitness, Hasil;

{$R *.dfm}

procedure TfrmParam.btnCloseClick(Sender: TObject);
```

```
begin
  Close;
end;

procedure TfrmParam.btnDefaultClick(Sender: TObject);
begin
  edtIterasi.Text:='50';
  edtR.Text:='0.95';
  edtTheta.Text:='0.01';
  edtT0.Text:='0.1';
  edtps.Text:='1';
  edtpr.Text:='1';
  edtpopsize.Text:='2';
  edtmaxgen.Text:='150';
  edtpcross.Text:='1';
  edtpmutat.Text:='0.01';
  edtAlpha1.Text:='50000000';
  edtAlpha2.Text:='5000000';
  edtflip.Text:='0.1';
  edtKonstM.Text:='1000000';
end;

procedure TfrmParam.btnNextClick(Sender: TObject);
begin
  gasa.gaR:=StrToFloat(edtR.Text);
  gasa.gaTheta:=StrToFloat(edtTheta.Text);
  //gasa.gaMaxgen:=StrToInt(edtIterasi.Text);
  //gasa.saX0:=StrToFloat(edtT0.Text);
  FucFitness.fitps:=StrToFloat(edtps.Text);
  FucFitness.fitpr:=StrToFloat(edtpr.Text);
  gasa.gaPopsize:=StrToInt(edtpopsize.Text);
  gasa.gaMaxgen:=StrToInt(edtmaxgen.Text);
  gasa.gaPcross:=StrToFloat(edtpcross.Text);
  gasa.gaPmutat:=StrToFloat(edtpmutat.Text);
  FucFitness.fitAlpha1:=StrToFloat(edtAlpha1.Text);
  FucFitness.fitAlpha2:=StrToFloat(edtAlpha2.Text);
  FucFitness.fitflip:=StrToFloat(edtFlip.Text);
  gasa.gaKa:=StrToFloat(edtKonstM.Text);
  frmHasil.Show;
end;

end.


unit ChromCost;

interface

uses Komplex,UnitGenerator,TypDatGa,EcoDispath,TotalCost;
```

```
type
 TChromCost=class(TtotalCost)
 private
  Ngen,NJam:integer;
  gen:TGenArr;
  Beban:Arr1;
  PL:Arr2;
  chrom:TChromBin2;
  function GetChrom:TChromBin2;
  procedure SetChrom(const dChrom:TChromBin2);
 public
  procedure InitData;
  procedure hitungPL;
  destructor Destroy;override;
  property toChrom:TChromBin2 read GetChrom write SetChrom;
 end;

implementation

//data accessing
procedure TChromCost.InitData;
begin
 Ngen:=toNgen;
 NJam:=toNJam;
 gen:=toGen;
 Beban:=toBeban;
end;

function TChromCost.GetChrom:TChromBin2;
var i,j:integer;
begin
 SetLength(result,Ngen+1,NJam+1);
 for i:=1 to Ngen do
 begin
  for j:=1 to NJam do
  begin
   result[i,j]:=chrom[i,j];
  end;
 end;
end;

procedure TChromCost.SetChrom(const dchrom:TChromBin2);
var i,j:integer;
begin
 SetLength(chrom,Ngen+1,NJam+1);
 for i:=1 to Ngen do
 begin
  for j:=1 to NJam do
  begin
   chrom[i,j]:=dChrom[i,j];
```

```
    end;
  end;
end;

//data processing
procedure TchromCost.hitungPL;
var i,j:integer;
    chrom1:TchromBin1;
    dbeban:double;
    PLa:Arr1;
    ecodis:TEcoDis;
begin
  SetLength(chrom1,Ngen+1);
  SetLength(PLa,Ngen+1);
  SetLength(PL,Ngen+1,NJam+1);
  ecoDis:=TEcoDis.Create;
  ecoDis.ecoNgen:=Ngen;
  ecoDis.ecoGen:=gen;
  for i:=1 to Njam do
  begin
    for j:=1 to Ngen do
    begin
      chrom1[j]:=chrom[j,i];
    end;
    dbeban:=Beban[i];
    ecoDis.ecoBeban:=dbeban;
    ecoDis.ecoChrom:=chrom1;
    PLa:=ecoDis.ecoPL;
    for j:=1 to Ngen do
    begin
      PL[j,i]:=PLa[j];
    end;
  end;
  toPL:=PL;
  doHitung;
  ecodis.Free;
end;

//destructor Object
destructor TchromCost.Destroy;
var i:integer;
begin
  try
    for i:=1 to Ngen do
    begin
      gen[i].Free;
    end;
  finally
    inherited Destroy;
  end;
```

```
end;

end.


unit Fitness;

interface

uses Komplex,TypDatGA,UnitGenerator,ChromCost;

type
  TFitness=class
  private
    Ngen,Njam:integer;
    ps,pr,flip,alpha1,alpha2,FIT:double;
    chrom:TChromBin2;
    gen:TGenArr;
    Beban,Res,AFLC:Arr1;
    sAFLC:iArr1;
    function GetNgen:integer;
    function GetNjam:integer;
    function Getps:double;
    function Getpr:double;
    function GetAlpha1:double;
    function GetAlpha2:double;
    function Getflip:double;
    function GetGen:TGenArr;
    function GetBeban:Arr1;
    function GetRes:Arr1;
    procedure SetNgen(const dNgen:integer);
    procedure SetNjam(const dNjam:integer);
    procedure Setps(const dps:double);
    procedure Setpr(const dpr:double);
    procedure SetAlpha1(const dAlpha1:double);
    procedure SetAlpha2(const dAlpha2:double);
    procedure Setflip(const dflip:double);
    procedure SetGen(const dGen:TGenArr);
    procedure SetBeban(const dBeban:Arr1);
    procedure SetRes(const dRes:Arr1);
    function flipo(batas:double):boolean;
    procedure HitungAFLC;
    procedure OperatorSwap(var dchrom:TchromBin2);
    procedure OperatorRepair(var dchrom:TchromBin2);
    function CekStatusLoad(const chr:TchromBin1;
          const dBeban,dRes:double):boolean;
    function CekStatusLoad2(const chr:TchromBin1;
       const dBeban,dRes:double):integer;
    procedure InitState(var dchrom:TchromBin2);
    function GetChrom:TChromBin2;
```

```
  function GetFIT:double;
 public
  constructor Create;overload;
  constructor Create(const dNgen,dNjam:integer);overload;
  procedure RepairSt(var dchrom:TchromBin2);
  procedure HitungFUC(var dchrom:TchromBin2;
        var dPinTime:integer;
        var dPinLoad,dFIT:double);
  procedure doHitung;
  destructor Destroy;override;
  property fitNgen:integer read GetNgen write SetNgen;
  property fitNjam:integer read GetNjam write SetNjam;
  property fitps:double read Getps write Setps;
  property fitpr:double read Getpr write Setpr;
  property fitflip:double read Getflip write Setflip;
  property fitAlpha1:double read GetAlpha1 write SetAlpha1;
  property fitAlpha2:double read GetAlpha2 write SetAlpha2;
  property fitChrom:TChromBin2 read GetChrom;
  property fitGen:TGenArr read GetGen write SetGen;
  property fitBeban:Arr1 read GetBeban write SetBeban;
  property fitRes:Arr1 read GetRes write SetRes;
  property fitFIT:double read GetFIT;
 end;

var fucFitness:TFitness;
  Beban,Res:Arr1;
  PLN:Arr2;

implementation

//constructor
constructor TFitness.Create;
begin
 inherited Create;
 Ngen:=0;
 Njam:=0;
end;

constructor TFitness.Create(const dNgen,dNjam:integer);
begin
 inherited Create;
 Ngen:=dNgen;
 Njam:=dNjam;
 SetLength(chrom,Ngen+1,Njam+1);
end;

//data accessing
function TFitness.GetNgen:integer;
begin
 result:=Ngen;
```

```
end;

function TFitness.GetNjam:integer;
begin
  result:=Njam;
end;

function TFitness.Getps:double;
begin
  result:=ps;
end;

function TFitness.Getpr:double;
begin
  result:=pr;
end;

function TFitness.Getflip:double;
begin
  result:=flip;
end;

function TFitness.GetAlpha1:double;
begin
  result:=Alpha1;
end;

function TFitness.GetAlpha2:double;
begin
  result:=Alpha2;
end;

function TFitness.GetGen:TGenArr;
var i:integer;
begin
  SetLength(result,Ngen+1);
  for i:=1 to Ngen do
  begin
    result[i]:=TPembangkit.Create;
    result[i].GenNama:=gen[i].GenNama;
    result[i].GenPmax:=gen[i].GenPmax;
    result[i].GenPmin:=gen[i].GenPmin;
    result[i].Gena2:=gen[i].Gena2;
    result[i].Gena1:=gen[i].Gena1;
    result[i].Gena0:=gen[i].Gena0;
    result[i].GenSh:=gen[i].GenSh;
    result[i].GenSc:=gen[i].GenSc;
    result[i].GenTup:=gen[i].GenTup;
    result[i].GenTdown:=gen[i].GenTdown;
    result[i].GenTcold:=gen[i].GenTcold;
```

```
    result[i].GenIniSt:=gen[i].GenIniSt;
  end;
end;

function TFitness.GetBeban:Arr1;
var i:integer;
begin
  SetLength(result,Njam+1);
  for i:=1 to Njam do
  begin
    result[i]:=Beban[i];
  end;
end;

function TFitness.GetRes:Arr1;
var i:integer;
begin
  SetLength(result,Njam+1);
  for i:=1 to Njam do
  begin
    result[i]:=Res[i];
  end;
end;

procedure TFitness.SetNgen(const dNgen:integer);
begin
  Ngen:=dNgen;
end;

procedure TFitness.SetNjam(const dNjam:integer);
begin
  Njam:=dNjam;
end;

procedure TFitness.Setps(const dps:double);
begin
  ps:=dps;
end;

procedure TFitness.Setpr(const dpr:double);
begin
  pr:=dpr;
end;

procedure TFitness.SetAlpha1(const dAlpha1:double);
begin
  Alpha1:=dAlpha1;
end;

procedure TFitness.SetAlpha2(const dAlpha2:double);
```

```
begin
  Alpha2:=dAlpha2;
end;

procedure TFitness.Setflip(const dflip:double);
begin
  flip:=dflip;
end;

procedure TFitness.SetGen(const dGen:TGenArr);
var i:integer;
begin
  SetLength(gen,Ngen+1);
  for i:=1 to Ngen do
  begin
    gen[i]:=TPembangkit.Create;
    gen[i].GenNama:=dgen[i].GenNama;
    gen[i].GenPmax:=dgen[i].GenPmax;
    gen[i].GenPmin:=dgen[i].GenPmin;
    gen[i].Gena2:=dgen[i].Gena2;
    gen[i].Gena1:=dgen[i].Gena1;
    gen[i].Gena0:=dgen[i].Gena0;
    gen[i].GenSh:=dgen[i].GenSh;
    gen[i].GenSc:=dgen[i].GenSc;
    gen[i].GenTup:=dgen[i].GenTup;
    gen[i].GenTdown:=dgen[i].GenTdown;
    gen[i].GenTcold:=dgen[i].GenTcold;
    gen[i].GenIniSt:=dgen[i].GenIniSt;
  end;
end;

procedure TFitness.SetBeban(const dBeban:Arr1);
var i:integer;
begin
  SetLength(Beban,Njam+1);
  for i:=1 to Njam do
  begin
    Beban[i]:=dBeban[i];
  end;
end;

procedure TFitness.SetRes(const dRes:Arr1);
var i:integer;
begin
  SetLength(Res,Njam+1);
  for i:=1 to Njam do
  begin
    Res[i]:=dRes[i];
  end;
end;
```

```
//data processing

function TFitness.flipo(batas:double):boolean;
var rand:double;
begin
  result:=false;
  rand:=random;
  if rand<batas then result:=true;
end;

procedure TFitness.HitungAFLC;
var i,j,tmp:integer;
begin
  SetLength(AFLC,Ngen+1);
  SetLength(sAFLC,Ngen+1);
  for i:=1 to Ngen do
  begin
    AFLC[i]:=gen[i].GenAFLC;
  end;
  for i:=1 to Ngen do
  begin
    sAFLC[i]:=i;
  end;
  for i:=1 to Ngen-1 do
  begin
    for j:=i to Ngen do
    begin
      if AFLC[i]>AFLC[j] then
      begin
        tmp:=sAFLC[i];
        sAFLC[i]:=sAFLC[j];
        sAFLC[j]:=tmp;
      end;
    end;
  end;
end;

procedure TFitness.OperatorSwap(var dchrom:TChromBin2);
var i,j,k,genmax:integer;
    maxAFLC:double;
begin
  for i:=1 to Njam do
  begin
    for j:=1 to Ngen do
    begin
      if not dchrom[j,i] then
      begin
        maxAFLC:=AFLC[j];
        genmax:=j;
```

```
      for k:=1 to Ngen do
      begin
       if dchrom[k,i] then
       begin
        if maxAFLC<AFLC[k] then
        begin
         maxAFLC:=AFLC[k];
         genmax:=k;
        end;
       end;
      end;
      if flipo(ps) then
      begin
       dchrom[j,i]:=true;
       dchrom[genmax,i]:=false;
      end;
     end;
   end;
  end;
end;

procedure TFitness.OperatorRepair(var dchrom:TchromBin2);
var i,j,Xs:integer;
begin
 for i:=1 to Ngen do
 begin
  Xs:=Gen[i].GenIniSt;
  for j:=1 to Njam do
  begin
   if dchrom[i,j] then
   begin
    if Xs>0 then
    begin
     Xs:=Xs+1;
    end
    else if Xs<0 then
    begin
     if abs(Xs)<Gen[i].GenTdown then
     begin
      if flipo(pr) then
      begin
       dchrom[i,j]:=false;
       Xs:=Xs-1;
      end;
     end
     else
     begin
      Xs:=1;
     end;
    end;
```

```
    end
    else if not dchrom[i,j] then
    begin
     if Xs>0 then
     begin
      if Xs<Gen[i].GenTup then
      begin
       if flipo(pr) then
       begin
        dchrom[i,j]:=true;
        Xs:=Xs+1;
       end;
      end
      else
      begin
       Xs:=-1;
      end;
     end
     else if Xs<0 then
     begin
      Xs:=Xs-1;
     end;
    end;
   end;
  end;
end;

function TFitness.CekStatusLoad(const chr:TchromBin1;
      const dBeban,dRes:double):boolean;
var i:integer;
   sPmax,sPmin:double;
begin
 result:=true;
 sPmax:=0;
 sPmin:=0;
 for i:=1 to Ngen do
 begin
  if chr[i] then
  begin
   sPmax:=sPmax+Gen[i].GenPmax;
   sPmin:=sPmin+Gen[i].GenPmin;
  end;
 end;
 if (dBeban+dRes)>sPmax then
 begin
  result:=false;
 end;
 if (dBeban+dRes)<sPmin then
 begin
  result:=false;
```

```
  end;
end;

function TFitness.CekStatusLoad2(const chr:TchromBin1;
        const dBeban,dRes:double):integer;
var i:integer;
    sPmax,sPmin:double;
begin
  result:=0;
  sPmax:=0;
  sPmin:=0;
  for i:=1 to Ngen do
  begin
    if chr[i] then
    begin
      sPmax:=sPmax+Gen[i].GenPmax;
      sPmin:=sPmin+Gen[i].GenPmin;
    end;
  end;
  if (dBeban+dRes)>sPmax then
  begin
    result:=2;
  end;
  if (dBeban+dRes)<sPmin then
  begin
    result:=1;
  end;
end;

procedure TFitness.InitState(var dchrom:TchromBin2);
var i,j,k,ia:integer;
    Xs:iArr1;
    Stat:TchromBin1;
    cekLoad:boolean;
begin
  SetLength(dchrom,Ngen+1,Njam+1);
  SetLength(Stat,Ngen+1);
  SetLength(Xs,Ngen+1);
  for i:=1 to Njam do
  begin
    for j:=1 to Ngen do
    begin
      if i=1 then
      begin
        Xs[j]:=Gen[j].GenIniSt;
      end;
      dchrom[j,i]:=false;
    end;
  end;
  for i:=1 to Njam do
```

```
begin
 for j:=1 to Ngen do
 begin
  ia:=sAFLC[j];
  for k:=1 to Ngen do
  begin
   Stat[k]:=dchrom[k,i];
  end;
  cekLoad:=CekStatusLoad(Stat,Beban[i],Res[i]);
  if cekLoad then
  begin
   dchrom[ia,i]:=flipo(flip);
   if dchrom[ia,i] then
   begin
    for k:=1 to Ngen do
    begin
     Stat[k]:=dchrom[k,i];
    end;
    cekLoad:=CekStatusLoad(Stat,Beban[i],Res[i]);
    if cekLoad then
    begin
     if Xs[ia]<0 then
     begin
      if abs(Xs[ia])>=Gen[ia].GenTdown then
      begin
       Xs[ia]:=1;
      end
      else
      begin
       dchrom[ia,i]:=false;
       Xs[ia]:=Xs[ia]-1;
      end;
     end
     else if Xs[ia]>0 then
     begin
      Xs[ia]:=Xs[ia]+1;
     end;
    end
    else if not cekLoad then
    begin
     dchrom[ia,i]:=false;
     if Xs[ia]<0 then
     begin
      Xs[ia]:=Xs[ia]-1;
     end
     else if Xs[ia]>0 then
     begin
      if Xs[ia]>=Gen[ia].GenTup then
      begin
       Xs[ia]:=-1;
```

```
      end
      else
      begin
        dchrom[ia,i]:=true;
        Xs[ia]:=Xs[ia]+1;
      end;
    end;
  end;
 end
 else if not dchrom[ia,i] then
 begin
  if Xs[ia]<0 then
  begin
    Xs[ia]:=Xs[ia]-1;
  end
  else if Xs[ia]>0 then
  begin
    if Xs[ia]>=Gen[ia].GenTup then
    begin
      Xs[ia]:=-1;
    end
    else
    begin
      dchrom[ia,i]:=true;
      Xs[ia]:=Xs[ia]+1;
    end;
  end;
 end;
end
else
begin
  if Xs[ia]<0 then
 begin
   if abs(Xs[ia])>=Gen[ia].GenTdown then
   begin
     dchrom[ia,i]:=true;
     Xs[ia]:=1;
   end
   else
   begin
     dchrom[ia,i]:=false;
     Xs[ia]:=Xs[ia]-1;
   end;
 end
 else if Xs[ia]>0 then
 begin
   dchrom[ia,i]:=true;
   Xs[ia]:=Xs[ia]+1;
 end;
end;
```

```
    end;
  end;
end;

procedure TFitness.RepairSt(var dchrom:TchromBin2);
var ia,i,j,k,cek:integer;
    chr1:TChromBin1;
begin
  OperatorSwap(dchrom);
  OperatorRepair(dchrom);
  SetLength(chr1,Ngen+1);
  for i:=1 to Njam do
  begin
    for j:=1 to Ngen do
    begin
      chr1[j]:=dchrom[j,i];
    end;
    cek:=CekStatusLoad2(chr1,Beban[i],Res[i]);
    if cek=2 then
    begin
      for k:=1 to Ngen do
      begin
        ia:=sAFLC[k];
        if not chr1[ia] then
        begin
          chr1[ia]:=true;
          cek:=CekStatusLoad2(chr1,Beban[i],Res[i]);
          if cek=0 then break;
        end;
      end;
    end
    else if cek=1 then
    begin
      for k:=Ngen downto 1 do
      begin
        ia:=sAFLC[k];
        if chr1[ia] then
        begin
          chr1[ia]:=false;
          cek:=CekStatusLoad2(chr1,Beban[i],Res[i]);
          if cek=0 then break;
        end;
      end;
    end;
  end;
end;

procedure TFitness.HitungFUC(var dchrom:TchromBin2;
        var dPinTime:integer;
        var dPinLoad,dFIT:double);
```

```
var i:integer;
   Status:TchromBin1;
   PL:Arr1;
   FCost:double;
   Ftotal:TChromCost;
begin
 SetLength(Status,Ngen+1);
 SetLength(PL,Ngen+1);
 for i:=1 to Ngen do
 begin
  Status[i]:=false;
  PL[i]:=0;
 end;
 dPinLoad:=0;
 dPinTime:=0;
 OperatorSwap(dchrom);
 OperatorRepair(dchrom);
 Ftotal:=TChromCost.Create;
 Ftotal.toNgen:=Ngen;
 Ftotal.toNJam:=Njam;
 Ftotal.toGen:=gen;
 Ftotal.toBeban:=Beban;
 Ftotal.toRes:=Res;
 Ftotal.InitData;
 Ftotal.toChrom:=dchrom;
 Ftotal.hitungPL;
 Ftotal.doHitung;
 FCost:=Ftotal.toFUC;
 dPinTime:=Ftotal.toPinTime;
 dPinLoad:=Ftotal.toPinDaya;
 dFIT:=FCost+alpha1*dPinLoad+alpha2*dPinTime*sqrt(Ngen);
 Ftotal.Free;
end;

procedure TFitness.doHitung;
var PinTime:integer;
   PinLoad:double;
begin
 HitungAFLC;
 repeat
  InitState(chrom);
  HitungFUC(chrom,PinTime,PinLoad,FIT);
 until (PinTime=0) and (PinLoad<0.000001);
end;

//data output
function TFitness.GetChrom:TChromBin2;
var i,j:integer;
begin
 SetLength(result,Ngen+1,Njam+1);
```

```
 for i:=1 to Ngen do
 begin
  for j:=1 to Njam do
  begin
   result[i,j]:=chrom[i,j];
  end;
 end;
end;

function TFitness.GetFIT:double;
begin
 result:=FIT;
end;


//destructor
destructor TFitness.Destroy;
var i:integer;
begin
 try
  for i:=1 to Ngen do
  begin
   gen[i].Free;
  end;
 finally
  inherited Destroy;
 end;
end;

end.


unit IGA;

interface

uses Komplex,TypDatGA,GeneticAlgorithm,Fitness,
   hasil;

type
 Tchromosome=TChromBin2;
 Tindividu=TIndividuBin2;
 Tpopulasi=array of Tindividu;

 TIGA=class(TGA)
 private
  oldpop,offspring:Tpopulasi;
  maxpop,tempop:Tindividu;
  typeGA:integer;
  sumfitness,min,avg,max,r,theta:double;
```

```
    min1,avg1,max1:Arr1;
    procedure SetTypeGA(const dTypeGA:integer);
    function GetR:double;
    function GetTheta:double;
    procedure SetR(const dR:double);
    procedure SetTheta(const dTheta:double);
    procedure InitGA;
    function CariIndividuMax:TIndividu;
    procedure Statistik;
    function Seleksi:integer;
    function Mutasi(const allele:boolean):boolean;
    procedure Crossover(const parent1,parent2:Tchromosome;
            var child1,child2:Tchromosome);
    function GantiIndividu(const dIndi:TIndividu):Tindividu;
    procedure MasukChildKeParent(child:Tindividu);
    procedure Generation;
    procedure GantiPopulasi(const dgen:integer);
    function GetChromHasil:Tchromosome;
    procedure doHitung;
    function GetMin:Arr1;
    function GetAvg:Arr1;
    function GetMax:Arr1;
  public
    constructor Create;
    destructor Destroy;override;
    property gaR:double read GetR write SetR;
    property gaTheta:double read GetTheta write SetTheta;
    property gaChromHasil:Tchromosome read GetChromHasil;
    property gaMin:Arr1 read GetMin;
    property gaAvg:Arr1 read GetAvg;
    property gaMax:Arr1 read GetMax;
    property gaTypeGA:integer read typeGA write SetTypeGA;
  end;

var gasa:TIGA;

implementation

//constructor
constructor TIGA.Create;
begin
  inherited Create;
  typeGA:=1;
end;

//data acessing
procedure TIGA.SetTypeGA(const dTypeGA:integer);
begin
  typeGA:=dTypeGA;
end;
```

```
function TIGA.GetR:double;
begin
  result:=R;
end;

function TIGA.GetTheta:double;
begin
  result:=theta;
end;

procedure TIGA.SetR(const dR:double);
begin
  R:=dR;
end;

procedure TIGA.SetTheta(const dTheta:double);
begin
  theta:=dTheta;
end;

//data processing
procedure TIGA.InitGA;
var i:integer;
begin
  SetLength(oldpop,gapopsize+1);
  SetLength(offspring,gapopsize+1);
  SetLength(min1,gamaxgen+1);
  SetLength(avg1,gamaxgen+1);
  SetLength(max1,gamaxgen+1);
  for i:=1 to gapopsize do
  begin
    SetLength(oldpop[i].chrom,gaNparam+1,galchrom+1);
    SetLength(offspring[i].chrom,gaNparam+1,galchrom+1);
    FucFitness.doHitung;
    oldpop[i].chrom:=FucFitness.fitChrom;
    oldpop[i].fitness:=gaKa/FucFitness.fitFIT;
  end;
  SetLength(maxpop.chrom,gaNparam+1,galchrom+1);
  SetLength(tempop.chrom,gaNparam+1,galchrom+1);
end;

function TIGA.CariIndividuMax:TIndividu;
var i,j,max:integer;
begin
  SetLength(result.chrom,gaNparam+1,galchrom+1);
  max:=1;
  for i:=2 to gapopsize do
  begin
    if oldpop[max].fitness<oldpop[i].fitness then
```

```
  begin
    max:=i;
  end;
 end;
 for i:=1 to gaNparam do
 begin
  for j:=1 to galchrom do
  begin
    result.chrom[i,j]:=oldpop[max].chrom[i,j];
  end;
 end;
 result.fitness:=oldpop[max].fitness;
end;

procedure TIGA.Statistik;
var i:integer;
begin
 min:=oldpop[1].fitness;
 max:=oldpop[1].fitness;
 sumfitness:=oldpop[1].fitness;
 for i:=2 to gapopsize do
 begin
  sumfitness:=sumfitness+oldpop[i].fitness;
  if oldpop[i].fitness>max then max:=oldpop[i].fitness;
  if oldpop[i].fitness<min then min:=oldpop[i].fitness;
 end;
 avg:=sumfitness/gapopsize;
end;

function TIGA.Seleksi:integer;
var rand,partsum:double;
   i:integer;
begin
 partsum:=0;
 i:=0;
 rand:=random*sumfitness;
 repeat
  i:=i+1;
  partsum:=partsum+oldpop[i].fitness;
 until (partsum>rand) or (i=gapopsize);
 Result:=i;
end;

function TIGA.Mutasi(const allele:boolean):boolean;
begin
 if GetFlip(gapmutat)=true then
 begin
  result:=not allele;
 end
 else
```

```
  begin
    result:=allele;
  end;
end;

procedure TIGA.Crossover(const parent1,parent2:Tchromosome;
        var child1,child2:Tchromosome);
var cross:boolean;
    tmp,i,j,point1,point2:integer;
begin
  SetLength(child1,gaNparam+1,galchrom+1);
  SetLength(child2,gaNparam+1,galchrom+1);
  cross:=GetFlip(gapcross);
  if cross then
  begin
    point1:=GetRandom(1,gaNparam);
    repeat
      point2:=GetRandom(1,gaNparam);
    until point1<>point2;
    if point1>point2 then
    begin
      tmp:=point1;
      point1:=point2;
      point2:=tmp;
    end;
    for i:=1 to point1 do
    begin
      for j:=1 to galchrom do
      begin
        child1[i,j]:=mutasi(parent1[i,j]);
        child2[i,j]:=mutasi(parent2[i,j]);
      end;
    end;
    for i:=point1+1 to point2 do
    begin
      for j:=1 to galchrom do
      begin
        child1[i,j]:=mutasi(parent2[i,j]);
        child2[i,j]:=mutasi(parent1[i,j]);
      end;
    end;
    if point2<gaNparam then
    begin
      for i:=point2+1 to gaNparam do
      begin
        for j:=1 to galchrom do
        begin
          child1[i,j]:=mutasi(parent1[i,j]);
          child2[i,j]:=mutasi(parent2[i,j]);
        end;
```

```
    end;
   end;
  end
  else
  begin
   for i:=1 to gaNparam do
   begin
    for j:=1 to galchrom do
    begin
     child1[i,j]:=mutasi(parent1[i,j]);
     child2[i,j]:=mutasi(parent2[i,j]);
    end;
   end;
  end;
end;

function TIGA.GantiIndividu(const dindi:Tindividu):Tindividu;
var i,j:integer;
begin
 SetLength(result.chrom,gaNparam+1,galchrom+1);
 for i:=1 to gaNparam do
 begin
  for j:=1 to galchrom do
  begin
   result.chrom[i,j]:=dindi.chrom[i,j];
  end;
 end;
 result.fitness:=dindi.fitness;
end;

procedure TIGA.MasukChildKeParent(child:Tindividu);
var i,NoMin:integer;
   Minu:double;
begin
 NoMin:=1;
 Minu:=oldpop[1].fitness;
 for i:=2 to gapopsize do
 begin
  if oldpop[i].fitness<minu then
  begin
   NoMin:=i;
   Minu:=oldpop[i].fitness;
  end;
 end;
 if child.fitness>oldpop[NoMin].fitness then
 begin
  oldpop[NoMin]:=GantiIndividu(child);
 end;
 Statistik;
end;
```

```
procedure TIGA.Generation;
var i,mate1,mate2,Pt:integer;
   PL,fita:double;
   child1,child2:Tindividu;
begin
 i:=1;
 SetLength(child1.chrom,gaNparam+1,galchrom+1);
 SetLength(child2.chrom,gaNparam+1,galchrom+1);
 repeat
  mate1:=Seleksi;
  mate2:=Seleksi;
  crossover(oldpop[mate1].chrom,oldpop[mate2].chrom,
        child1.chrom,child2.chrom);
  FucFitness.RepairSt(child1.chrom);
  FucFitness.HitungFUC(child1.chrom,Pt,PL,fita);
  child1.fitness:=gaKa/fita;
  if (Pt=0) and (PL<0.000001) then
  begin
   offspring[i]:=GantiIndividu(child1);
   MasukChildKeParent(child1);
   i:=i+1;
  end;
  if i>gapopsize then exit;
  FucFitness.RepairSt(child2.chrom);
  FucFitness.HitungFUC(child2.chrom,Pt,PL,fita);
  child2.fitness:=gaKa/fita;
  if (Pt=0) and (PL<0.000001) then
  begin
   offspring[i]:=GantiIndividu(child2);
   MasukChildKeParent(child2);
   i:=i+1;
  end;
 until i>gapopsize;
end;

procedure TIGA.GantiPopulasi(const dgen:integer);
var i,j,NoMin:integer;
   minu,ran,the:double;
begin
 if typeGA=2 then
 begin
  for i:=1 to gapopsize do
  begin
   oldpop[i]:=GantiIndividu(offspring[i]);
  end;
 end
 else if typeGA=1 then
 begin
  the:=pangkat(r,dgen-1)*theta;
```

```
  for i:=1 to gapopsize do
  begin
    NoMin:=1;
    minu:=oldpop[1].fitness;
    for j:=2 to gapopsize do
    begin
      if oldpop[j].fitness<minu then
      begin
        NoMin:=j;
        minu:=oldpop[j].fitness;
      end;
    end;
    if offspring[i].fitness>minu then
    begin
      oldpop[NoMin]:=GantiIndividu(offspring[i]);
    end
    else
    begin
      ran:=random;
      if ran>the then
      begin
        oldpop[NoMin]:=GantiIndividu(offspring[i]);
      end;
    end;
  end;
 end;
end;

procedure TIGA.doHitung;
var gen:integer;
begin
 InitGA;
 Statistik;
 gen:=0;
 maxpop:=CariIndividuMax;
 repeat
   generation;
   gantipopulasi(gen);
   tempop:=CariIndividuMax;
   if maxpop.fitness<tempop.fitness then
   begin
     maxpop:=GantiIndividu(tempop);
   end;
   gen:=gen+1;
   frmHasil.pbIterasi.StepBy(1);
   min1[gen]:=min;
   avg1[gen]:=avg;
   max1[gen]:=max;
 until gen>=gamaxgen;
end;
```

```
//data output
function TIGA.GetChromHasil:Tchromosome;
var i,j:integer;
begin
  SetLength(result,gaNparam+1,galchrom+1);
  doHitung;
  for i:=1 to gaNparam do
  begin
    for j:=1 to galchrom do
    begin
      result[i,j]:=maxpop.chrom[i,j];
    end;
  end;
end;

function TIGA.GetMin;
var i:integer;
begin
  SetLength(result,gamaxgen+1);
  for i:=1 to gamaxgen do
  begin
    result[i]:=min1[i];
  end;
end;

function TIGA.GetAvg:Arr1;
var i:integer;
begin
  SetLength(result,gamaxgen+1);
  for i:=1 to gamaxgen do
  begin
    result[i]:=avg1[i];
  end;
end;

function TIGA.GetMax:Arr1;
var i:integer;
begin
  SetLength(result,gamaxgen+1);
  for i:=1 to gamaxgen do
  begin
    result[i]:=max1[i];
  end;
end;

//destructor
destructor TIGA.Destroy;
begin
  inherited Destroy;
```

```
end;

end.


unit Komplex;

interface

uses Dialogs;

type
  Arr1=array of double;
  Arr2=array of array of double;
  iArr1=array of integer;
  iArr2=array of array of integer;

  TKomplex=class
  private
    re,im:double;
    function GetRe:double;
    function GetIm:double;
    procedure SetRe(const dRe:double);
    procedure SetIm(const dIm:double);
  public
    constructor Create;overload;
    constructor Create(const aRe:double);overload;
    constructor Create(const aRe,aIm:double);overload;
    constructor Create(const r,theta:extended);overload;
    destructor Destroy;override;
    function GetStringJ(len:byte):string;
    function GetStringI(len:byte):string;
    function GetAbs:double;
    function GetAngleRad:double;
    function GetAngleDeg:double;
    procedure Assign(const aValue:TKomplex);overload;
    procedure Assign(const aRe,aIm:double);overload;
    procedure DoTambah(const aValue:TKomplex);overload;
    procedure DoTambah(const aRe,aIm:double);overload;
    procedure DoTambah(const aValue1,aValue2:TKomplex);overload;
    procedure DoTambah(const aRe1,aIm1,aRe2,aIm2:double);overload;
    procedure DoKurang(const aValue:TKomplex);overload;
    procedure DoKurang(const aRe,aIm:double);overload;
    procedure DoKurang(const aValue1,aValue2:TKomplex);overload;
    procedure DoKurang(const aRe1,aIm1,aRe2,aIm2:double);overload;
    procedure DoKali(const aValue:TKomplex);overload;
    procedure DoKali(const aRe,aIm:double);overload;
    procedure DoKali(const aValue1,aValue2:TKomplex);overload;
    procedure DoKali(const aRe1,aIm1,aRe2,aIm2:double);overload;
    procedure DoBagi(const aValue:TKomplex);overload;
```

```
  procedure DoBagi(const aRe,aIm:double);overload;
  procedure DoBagi(const aValue1,aValue2:TKomplex);overload;
  procedure DoBagi(const aRe1,aIm1,aRe2,aIm2:double);overload;
  procedure DoConj;overload;
  procedure DoConj(const aValue:TKomplex);overload;
  procedure DoConj(const aRe,aIm:double);overload;
  procedure DoNegative;overload;
  procedure DoNegative(const aValue:TKomplex);overload;
  procedure DoNegative(const aRe,aIm:double);overload;
  procedure DoPangkat(pangkat:double);
  property xRe:double read GetRe write SetRe;
  property xIm:double read GetIm write SetIm;
 end;

 CArr1=array of TKomplex;
 CArr2=array of array of TKomplex;

function RealToStr(Num:double;Pecahan:byte):String;
function StrToReal(Huruf:string):double;
function Pangkat(Val,pangkat:double):double;

implementation

function RealToStr(Num:double;Pecahan:byte):String;
var Hasil:String;
    le:byte;
begin
  le:=sizeof(Num);
  Str(Num:le:Pecahan,Hasil);
  Result:=Hasil;
end;

function Pangkat(Val,pangkat:double):double;
begin
  Result:=exp(Pangkat*ln(Val));
end;

function StrToReal(Huruf:string):double;
var Temp:double;
    Code:integer;
begin
  val(Huruf,Temp,Code);
  Result:=Temp;
end;

constructor TKomplex.Create;
begin
  inherited Create;
  re:=0;
  im:=0;
```

```
end;

constructor TKomplex.Create(const aRe:double);
begin
  inherited Create;
  re:=aRe;
  im:=0;
end;

constructor TKomplex.Create(const aRe,aIm:double);
begin
  inherited Create;
  re:=aRe;
  im:=aIm;
end;

constructor TKomplex.Create(const r,theta:extended);
begin
  inherited Create;
  re:=r*cos(theta);
  im:=r*sin(theta);
end;

destructor TKomplex.Destroy;
begin
  inherited Destroy;
end;

function TKomplex.GetRe:double;
begin
  result:=re;
end;

function TKomplex.GetIm:double;
begin
  result:=im;
end;

function TKomplex.GetStringJ(len:byte):string;
begin
  result:=RealToStr(re,len);
  if im<0 then
  begin
    result:=result+' - j'+RealToStr(abs(im),len);
  end
  else if im>0 then
  begin
    result:=result+' + j'+RealToStr(abs(im),len);
  end;
end;
```

```
function TKomplex.GetStringI(len:byte):string;
begin
  result:=RealToStr(re,len);
  if im<0 then
  begin
    result:=result+' - '+RealToStr(abs(im),len)+'i';
  end
  else if im>0 then
  begin
    result:=result+' + '+RealToStr(abs(im),len)+'i';
  end;
end;

function TKomplex.GetAbs:double;
begin
  result:=sqrt(sqr(re)+sqr(im));
end;

function TKomplex.GetAngleRad:double;
begin
  try
    result:=arctan(im/re);
  except
    result:=0;
    MessageDlg('Bilangan tidak bisa dicari sudutnya!',mtWarning,[mbOK],0);
  end;
end;

function TKomplex.GetAngleDeg:double;
var pi:double;
begin
  try
    pi:=4*arctan(1);
    result:=arctan(im/re)*180/pi;
  except
    result:=0;
    MessageDlg('Bilangan tidak bisa dicari sudutnya!',mtWarning,[mbOK],0);
  end;
end;

procedure TKomplex.Assign(const aValue:TKomplex);
begin
  re:=aValue.re;
  im:=aValue.im;
end;

procedure TKomplex.Assign(const aRe,aIm:double);
begin
  re:=aRe;
```

```
 im:=aIm;
end;

procedure TKomplex.SetRe(const dRe:double);
begin
 re:=dRe;
end;

procedure TKomplex.SetIm(const dIm:double);
begin
 im:=dIm;
end;

procedure TKomplex.DoTambah(const aValue:TKomplex);
begin
 re:=re+aValue.re;
 im:=im+aValue.im;
end;

procedure TKomplex.DoTambah(const aRe,aIm:double);
begin
 re:=re+aRe;
 im:=im+aIm;
end;

procedure TKomplex.DoTambah(const aValue1,aValue2:TKomplex);
begin
 re:=aValue1.re+aValue2.re;
 im:=aValue1.im+aValue2.im;
end;

procedure TKomplex.DoTambah(const aRe1,aIm1,aRe2,aIm2:double);
begin
 re:=aRe1+aRe2;
 im:=aIm1+aIm2;
end;

procedure TKomplex.DoKurang(const aValue:TKomplex);
begin
 re:=re-aValue.re;
 im:=im-aValue.im;
end;

procedure TKomplex.DoKurang(const aRe,aIm:double);
begin
 re:=re-aRe;
 im:=im-aIm;
end;

procedure TKomplex.DoKurang(const aValue1,aValue2:TKomplex);
```

```
begin
  re:=aValue1.re-aValue2.re;
  im:=aValue1.im-aValue2.im;
end;

procedure TKomplex.DoKurang(const aRe1,aIm1,aRe2,aIm2:double);
begin
  re:=aRe1-aRe2;
  im:=aIm1-aIm2;
end;

procedure TKomplex.DoKali(const aValue:TKomplex);
var tmpRe,tmpIm:double;
begin
  tmpRe:=re;
  tmpIm:=im;
  re:=tmpRe*aValue.re-tmpIm*aValue.im;
  im:=tmpRe*aValue.im+tmpIm*aValue.re;
end;

procedure TKomplex.DoKali(const aRe,aIm:double);
var tmpRe,tmpIm:double;
begin
  tmpRe:=re;
  tmpIm:=im;
  re:=tmpRe*aRe-tmpIm*aIm;
  im:=tmpRe*aIm+tmpIm*aRe;
end;

procedure TKomplex.DoKali(const aValue1,aValue2:TKomplex);
var tmpRe1,tmpIm1,tmpRe2,tmpIm2:double;
begin
  tmpRe1:=aValue1.xRe;tmpIm1:=aValue1.xIm;
  tmpRe2:=aValue2.xRe;tmpIm2:=aValue2.xIm;
  re:=tmpRe1*tmpRe2-tmpIm1*tmpIm2;
  im:=tmpRe1*tmpIm2+tmpIm1*tmpRe2;
end;

procedure TKomplex.DoKali(const aRe1,aIm1,aRe2,aIm2:double);
begin
  re:=aRe1*aRe2-aIm1*aIm2;
  im:=aRe1*aIm2+aIm1*aRe2;
end;

procedure TKomplex.DoBagi(const aValue:TKomplex);
var tmpRe,tmpIm:double;
begin
  try
    tmpRe:=re;
    tmpIm:=im;
```

```
   re:=(tmpRe*aValue.re+tmpIm*aValue.im)/(sqr(aValue.re)+sqr(aValue.im));
   im:=(tmpIm*aValue.re-tmpRe*aValue.im)/(sqr(aValue.re)+sqr(aValue.im));
  except
   MessageDlg('Pembagian salah atau bilangan pembagi nol!',
          mtwarning,[mbOK],0);
  end;
end;

procedure TKomplex.DoBagi(const aRe,aIm:double);
var tmpRe,tmpIm:double;
begin
  try
   tmpRe:=re;
   tmpIm:=im;
   re:=(tmpRe*aRe+tmpIm*aIm)/(sqr(aRe)+sqr(aIm));
   im:=(tmpIm*aRe-tmpRe*aIm)/(sqr(aRe)+sqr(aIm));
  except
   MessageDlg('Pembagian salah atau bilangan pembagi nol!',
          mtwarning,[mbOK],0);
  end;
end;

procedure TKomplex.DoBagi(const aValue1,aValue2:TKomplex);
var tmpRe1,tmpIm1,tmpRe2,tmpIm2:double;
begin
  try
   tmpRe1:=aValue1.xRe;tmpIm1:=aValue1.xIm;
   tmpRe2:=aValue2.xRe;tmpIm2:=aValue2.xIm;
   re:=(tmpRe1*tmpRe2+tmpIm1*tmpIm2)/(sqr(tmpRe2)+sqr(tmpIm2));
   im:=(tmpIm1*tmpRe2-tmpRe1*tmpIm2)/(sqr(tmpRe2)+sqr(tmpIm2));
  except
   MessageDlg('Pembagian salah atau bilangan pembagi nol!',
          mtwarning,[mbOK],0);
  end;
end;

procedure TKomplex.DoBagi(const aRe1,aIm1,aRe2,aIm2:double);
begin
  try
   re:=(aRe1*aRe2+aIm1*aIm2)/(sqr(aRe2)+sqr(aIm2));
   im:=(aIm1*aRe2-aRe1*aIm2)/(sqr(aRe2)+sqr(aIm2));
  except
   MessageDlg('Pembagian salah atau bilangan pembagi nol!',
          mtwarning,[mbOK],0);
  end;
end;

procedure TKomplex.DoConj;
begin
  re:=re;
```

```
 im:=-im;
end;

procedure TKomplex.DoConj(const aValue:TKomplex);
begin
  re:=aValue.xRe;
  im:=-aValue.xIm;
end;

procedure TKomplex.DoConj(const aRe,aIm:double);
begin
  re:=aRe;
  im:=-aIm;
end;

procedure TKomplex.DoNegative;
begin
  re:=-re;
  im:=-im;
end;

procedure TKomplex.DoNegative(const aValue:TKomplex);
begin
  re:=-aValue.xRe;
  im:=-aValue.xIm;
end;

procedure TKomplex.DoNegative(const aRe,aIm:double);
begin
  re:=-aRe;
  im:=-aIm;
end;

procedure TKomplex.DoPangkat(pangkat:double);
var theta,sum:double;
begin
  try
    theta:=arctan(im/re);
    sum:=exp((pangkat/2)*ln(sqr(re)+sqr(im)));
    re:=sum*cos(pangkat*theta);
    im:=sum*sin(pangkat*theta);
  except
    MessageDlg('Bilangan tidak bisa dipangkatkan!',mtWarning,[mbOK],0);
  end;
end;

end.


unit TypDatGA;
```

```
interface

uses Komplex;

type
 TchromBin1=array of boolean;
 TchromBin2=array of array of boolean;
 TchromFloat1=Arr1;
 TchromFloat2=Arr2;
 TchromInt1=iArr1;
 TchromInt2=iArr2;

 TindividuBin1=record
  chrom:TchromBin1;
  fitness:double;
 end;

 TindividuBin2=record
  chrom:TchromBin2;
  fitness:double;
 end;

 TindividuFloat1=record
  chrom:TchromFloat1;
  fitness:double;
 end;

 TindividuFloat2=record
  chrom:TchromFloat2;
  fitness:double;
 end;

 TindividuInt1=record
  chrom:TchromInt1;
  fitness:double;
 end;

 TindividuInt2=record
  chrom:TchromInt2;
  fitness:double;
 end;

implementation

end.


unit TotalCost;
```

```
interface

uses Komplex,UnitGenerator;

type
 TTotalCost=class
 private
  Ngen,Njam,pinTime:integer;
  PL:Arr2;
  Gen:TGenArr;
  CostPerJam,Beban,Res:Arr1;
  FUC,pinDaya:double;
  function GetNgen:integer;
  function GetNJam:integer;
  function GetGen:TGenArr;
  function GetBeban:Arr1;
  function GetRes:Arr1;
  function GetPL:Arr2;
  function GetCostPerJam:Arr1;
  function GetFUC:double;
  function GetPinDaya:double;
  function GetPinTime:integer;
  procedure SetNgen(const dNgen:integer);
  procedure SetNJam(const dNJam:integer);
  procedure SetGen(const dGen:TGenArr);
  procedure SetBeban(const dBeban:Arr1);
  procedure SetRes(const dRes:Arr1);
  procedure SetPL(const dPL:Arr2);

 public
  constructor Create;overload;
  constructor Create(const dNgen,dNJam:integer;const dPL:Arr2;
        const dGen:TGenArr;const dBeban,dRes:Arr1);overload;
  procedure doHitung;
  destructor Destroy;override;
  property toNgen:integer read GetNgen write SetNgen;
  property toNJam:integer read GetNJam write SetNJam;
  property toPL:Arr2 read GetPL write SetPL;
  property toGen:TGenArr read GetGen write SetGen;
  property toBeban:Arr1 read GetBeban write SetBeban;
  property toRes:Arr1 read GetRes write SetRes;
  property toCostPerJam:Arr1 read GetCostPerJam;
  property toFUC:double read GetFUC;
  property toPinDaya:double read GetPinDaya;
  property toPinTime:integer read GetPinTime;
 end;

implementation

//constructor object
```

```
constructor TtotalCost.Create;
begin
  inherited Create;
  Ngen:=0;
  NJam:=0;
  pinDaya:=0;
  pinTime:=0;
end;

constructor TtotalCost.Create(const dNgen,dNJam:integer;const dPL:Arr2;
          const dGen:TGenArr;const dBeban,dRes:Arr1);
var i,j:integer;
begin
  inherited Create;
  Ngen:=dNgen;
  NJam:=dNJam;
  SetLength(PL,Ngen+1,NJam+1);
  SetLength(gen,Ngen+1);
  SetLength(Beban,NJam+1);
  SetLength(Res,NJam+1);
  for i:=1 to Ngen do
  begin
    gen[i]:=TPembangkit.Create;
    Gen[i].GenNama:=dGen[i].GenNama;
    Gen[i].GenPmax:=dGen[i].GenPmax;
    Gen[i].GenPmin:=dGen[i].GenPmin;
    Gen[i].Gena2:=dGen[i].Gena2;
    Gen[i].Gena1:=dGen[i].Gena1;
    Gen[i].Gena0:=dGen[i].Gena0;
    Gen[i].GenSh:=dGen[i].GenSh;
    Gen[i].GenSc:=dGen[i].GenSc;
    Gen[i].GenTup:=dGen[i].GenTup;
    Gen[i].GenTdown:=dGen[i].GenTdown;
    Gen[i].GenTcold:=dGen[i].GenTcold;
    Gen[i].GenIniSt:=dGen[i].GenIniSt;
    for j:=1 to NJam do
    begin
      PL[i,j]:=dPL[i,j];
    end;
  end;
  for i:=1 to NJam do
  begin
    Beban[i]:=dBeban[i];
    Res[i]:=dRes[i];
  end;
  pinDaya:=0;
  pinTime:=0;
end;

 //data accessing
```

```
function TtotalCost.GetNgen:integer;
begin
  result:=Ngen;
end;

function TtotalCost.GetNJam:integer;
begin
  result:=NJam;
end;

function TtotalCost.GetGen:TGenArr;
var i:integer;
begin
  SetLength(result,Ngen+1);
  for i:=1 to Ngen do
  begin
    result[i]:=TPembangkit.Create;
    result[i].GenNama:=Gen[i].GenNama;
    result[i].GenPmax:=Gen[i].GenPmax;
    result[i].GenPmin:=Gen[i].GenPmin;
    result[i].Gena2:=Gen[i].Gena2;
    result[i].Gena1:=Gen[i].Gena1;
    result[i].Gena0:=Gen[i].Gena0;
    result[i].GenSh:=Gen[i].GenSh;
    result[i].GenSc:=Gen[i].GenSc;
    result[i].GenTup:=Gen[i].GenTup;
    result[i].GenTdown:=Gen[i].GenTdown;
    result[i].GenTcold:=Gen[i].GenTcold;
    result[i].GenIniSt:=Gen[i].GenIniSt;
  end;
end;

function TtotalCost.GetBeban:Arr1;
var i:integer;
begin
  SetLength(result,NJam+1);
  for i:=1 to NJam do
  begin
    result[i]:=Beban[i];
  end;
end;

function TtotalCost.GetRes:Arr1;
var i:integer;
begin
  SetLength(result,NJam+1);
  for i:=1 to NJam do
  begin
    result[i]:=Res[i];
  end;
```

```
end;

function TtotalCost.GetPL:Arr2;
var i,j:integer;
begin
  SetLength(result,Ngen+1,NJam+1);
  for i:=1 to Ngen do
  begin
    for j:=1 to NJam do
    begin
      result[i,j]:=PL[i,j];
    end;
  end;
end;

procedure TtotalCost.SetNgen(const dNgen:integer);
begin
  Ngen:=dNgen;
end;

procedure TtotalCost.SetNJam(const dNJam:integer);
begin
  NJam:=dNJam;
end;

procedure TtotalCost.SetGen(const dGen:TGenArr);
var i:integer;
begin
  SetLength(Gen,Ngen+1);
  for i:=1 to Ngen do
  begin
    Gen[i]:=TPembangkit.Create;
    Gen[i].GenNama:=dGen[i].GenNama;
    Gen[i].GenPmax:=dGen[i].GenPmax;
    Gen[i].GenPmin:=dGen[i].GenPmin;
    Gen[i].Gena2:=dGen[i].Gena2;
    Gen[i].Gena1:=dGen[i].Gena1;
    Gen[i].Gena0:=dGen[i].Gena0;
    Gen[i].GenSh:=dGen[i].GenSh;
    Gen[i].GenSc:=dGen[i].GenSc;
    Gen[i].GenTup:=dGen[i].GenTup;
    Gen[i].GenTdown:=dGen[i].GenTdown;
    Gen[i].GenTcold:=dGen[i].GenTcold;
    Gen[i].GenIniSt:=dGen[i].GenIniSt;
  end;
end;

procedure TtotalCost.SetBeban(const dBeban:Arr1);
var i:integer;
begin
```

```
  SetLength(Beban,NJam+1);
  for i:=1 to NJam do
  begin
    Beban[i]:=dBeban[i];
  end;
end;

procedure TtotalCost.SetRes(const dRes:Arr1);
var i:integer;
begin
  SetLength(Res,NJam+1);
  for i:=1 to NJam do
  begin
    Res[i]:=dRes[i];
  end;
end;

procedure TtotalCost.SetPL(const dPL:Arr2);
var i,j:integer;
begin
  SetLength(PL,Ngen+1,NJam+1);
  for i:=1 to Ngen do
  begin
    for j:=1 to NJam do
    begin
      PL[i,j]:=dPL[i,j];
    end;
  end;
end;

//data processing
procedure TtotalCost.doHitung;
var i,j,Xs,tcold:integer;
    sLoad,sPmin,sPmax:double;
    costGen:Arr2;
    costSUC:Arr2;
begin
  SetLength(costGen,Ngen+1,NJam+1);
  SetLength(costSUC,Ngen+1,NJam+1);
  SetLength(CostPerJam,NJam+1);
  pinTime:=0;
  for i:=1 to Ngen do
  begin
    Xs:=Gen[i].GenIniSt;
    tcold:=Gen[i].GenTdown+Gen[i].GenTcold;
    for j:=1 to NJam do
    begin
      costGen[i,j]:=0;
      costSUC[i,j]:=0;
      if PL[i,j]<>0 then
```

```
  begin
    costGen[i,j]:=Gen[i].GetBiaya(PL[i,j]);
    if Xs>0 then
    begin
      Xs:=Xs+1;
    end
    else if Xs<0 then
    begin
      if abs(Xs)<Gen[i].GenTdown then
      begin
        pinTime:=pinTime+1;
      end;
      if abs(Xs)>tcold then
      begin
        costSUC[i,j]:=Gen[i].GenSc;
      end
      else
      begin
        costSUC[i,j]:=Gen[i].GenSh;
      end;
      Xs:=1;
    end;
  end
  else if PL[i,j]=0 then
  begin
    if Xs>0 then
    begin
      if Xs<Gen[i].GenTup then
      begin
        PinTime:=PinTime+1;
      end;
      Xs:=-1;
    end
    else if Xs<0 then
    begin
      Xs:=Xs-1;
    end;
  end;
 end;
end;
FUC:=0;
for i:=1 to NJam do
begin
 CostPerJam[i]:=0;
 sLoad:=0;
 for j:=1 to Ngen do
 begin
  CostPerJam[i]:=CostPerJam[i]+costGen[j,i]+costSUC[j,i];
  sLoad:=sLoad+PL[j,i];
 end;
```

```
    FUC:=FUC+CostPerJam[i];
  end;
  pinDaya:=0;
  for i:=1 to NJam do
  begin
    sPmin:=0;
    sPmax:=0;
    for j:=1 to Ngen do
    begin
      if PL[j,i]<>0 then
      begin
        sPmin:=sPmin+Gen[j].GenPmin;
        sPmax:=sPmax+Gen[j].GenPmax;
      end;
    end;
    if (Beban[i]+Res[i])<sPmin then
    begin
      pinDaya:=pinDaya+abs(sPmin-Beban[i]+Res[i]);
    end
    else if (Beban[i]+Res[i])>sPmax then
    begin
      pinDaya:=pinDaya+abs(Beban[i]+Res[i]-sPmax);
    end;
  end;
end;

function TtotalCost.GetCostPerJam:Arr1;
var i:integer;
begin
  SetLength(result,NJam+1);
  for i:=1 to NJam do
  begin
    result[i]:=CostPerJam[i];
  end;
end;

function TtotalCost.GetFUC:double;
begin
  result:=FUC;
end;

function TtotalCost.GetPinDaya:double;
begin
  result:=pinDaya;
end;

function TtotalCost.GetPinTime:integer;
begin
  result:=pinTime;
end;
```

```
//destructor object
destructor TtotalCost.Destroy;
var i:integer;
begin
  try
    for i:=1 to Ngen do
    begin
      Gen[i].Free;
    end;
  finally
    inherited Destroy;
  end;
end;

end.


unit Hasil;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart, Grids,
  ComCtrls;

type
  TfrmHasil = class(TForm)
    btnKurva: TButton;
    brnUjiGA: TButton;
    btnClose: TButton;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    fgStatus: TStringGrid;
    TabSheet2: TTabSheet;
    fgLoad: TStringGrid;
    TabSheet4: TTabSheet;
    fgCostJam: TStringGrid;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    lblWatt: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    edtPinLoad: TEdit;
    edtPinTime: TEdit;
```

```
    edtTotalCost: TEdit;
    edtTime: TEdit;
    edtBiayaPLN: TEdit;
    TabSheet5: TTabSheet;
    fgPLN: TStringGrid;
    TabSheet3: TTabSheet;
    Chart1: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    TabSheet6: TTabSheet;
    fgHasilGA: TStringGrid;
    Label8: TLabel;
    edtSelisihBiaya: TEdit;
    rbtGA: TRadioButton;
    rbtGASA: TRadioButton;
    pbIterasi: TProgressBar;
    procedure btnCloseClick(Sender: TObject);
    procedure brnUjiGAClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmHasil: TfrmHasil;

implementation

uses IGA, Fitness, ChromCost, TypDatGA, Komplex;

{$R *.dfm}

procedure TfrmHasil.btnCloseClick(Sender: TObject);
begin
  Close;
end;

procedure TfrmHasil.brnUjiGAClick(Sender: TObject);
var i,j,PinTime:integer;
    FUC,FUCPLN,PinDaya:double;
    chromHasil:TChromBin2;
    FCost:TChromCost;
    PL:Arr2;
    min,avg,max,CostPerJam,CostPerJamPLN:Arr1;
    mulai,selesai,selang:TDateTime;
    jam,menit,detik,mdetik:word;
begin
  if rbtGA.Checked=true then
```

```
begin
  gasa.gaTypeGA:=2;
end;
if rbtGASA.Checked=true then
begin
  gasa.gaTypeGA:=1;
end;
mulai:=Time;
pbIterasi.Max:=gasa.gaMaxgen;
chromHasil:=gasa.gaChromHasil;
selesai:=Time;
min:=gasa.gaMin;
avg:=gasa.gaAvg;
max:=gasa.gaMax;
FCost:=TChromCost.Create;
FCost.toNgen:=FucFitness.fitNgen;
FCost.toNJam:=FucFitness.fitNjam;
FCost.toGen:=FucFitness.fitGen;
FCost.toBeban:=FucFitness.fitBeban;
FCost.toRes:=FucFitness.fitRes;
FCost.InitData;
FCost.toChrom:=chromHasil;
FCost.hitungPL;
FCost.doHitung;
PL:=FCost.toPL;
CostPerJam:=FCost.toCostPerJam;
FUC:=FCost.toFUC;
PinTime:=FCost.toPinTime;
PinDaya:=FCost.toPinDaya;
FCost.toPL:=PLN;
FCost.doHitung;
FUCPLN:=FCost.toFUC;
CostPerJamPLN:=FCost.toCostPerJam;
for i:=1 to FucFitness.fitNgen do
begin
  for j:=1 to FucFitness.fitNjam do
  begin
    fgLoad.Cells[j,i]:=RealToStr(PL[i,j],0);
    fgPLN.Cells[j,i]:=RealToStr(PLN[i,j],0);
    if chromHasil[i,j]=true then
    begin
      fgStatus.Cells[j,i]:='1';
    end
    else
    begin
      fgStatus.Cells[j,i]:='0';
    end;
  end;
end;
for i:=1 to FucFitness.fitNjam do
```

```
  begin
    fgCostJam.Cells[1,i]:=FormatFloat('#,##0',CostPerJam[i]);
    fgCostJam.Cells[2,i]:=FormatFloat('#,##0',CostPerJamPLN[i]);
    fgCostJam.Cells[3,i]:=FormatFloat('#,##0',(CostPerJamPLN[i]-CostPerJam[i]));
  end;
  edtPinLoad.Text:=RealToStr(PinDaya,0);
  edtPinTime.Text:=IntToStr(PinTime);
  edtTotalCost.Text:=FormatFloat('#,##0',FUC);
  edtBiayaPLN.Text:=FormatFloat('#,##0',FUCPLN);
  edtSelisihBiaya.Text:=FormatFloat('#,##0',(FUCPLN-FUC));
  Series1.Clear;
  Series2.Clear;
  for i:=1 to FucFitness.fitNjam do
  begin
    Series1.Add(CostPerJamPLN[i],IntToStr(i));
    Series2.Add(CostPerJam[i],IntToStr(i));
  end;
  for i:=1 to gasa.gaMaxgen do
  begin
    fgHasilGA.Cells[0,i]:=IntToStr(i);
    fgHasilGA.Cells[1,i]:=RealToStr(min[i],4);
    fgHasilGA.Cells[2,i]:=RealToStr(avg[i],4);
    fgHasilGA.Cells[3,i]:=RealToStr(max[i],4);
  end;
  selang:=selesai-mulai;
  DecodeTime(selang,jam,menit,detik,mdetik);
  edtTime.Text:=IntToStr(jam) + ':' + IntToStr(menit) +
          ':' + IntToStr(detik) + ':' + IntToStr(mdetik);
  FCost.Free;
end;

procedure TfrmHasil.FormActivate(Sender: TObject);
var i:integer;
begin
  fgStatus.RowCount:=FucFitness.fitNgen+1;
  fgStatus.ColCount:=FucFitness.fitNjam+1;
  fgLoad.RowCount:=FucFitness.fitNgen+1;
  fgLoad.ColCount:=Fucfitness.fitNjam+1;
  fgCostJam.RowCount:=FucFitness.fitNjam+1;
  fgPLN.RowCount:=FucFitness.fitNgen+1;
  fgPLN.ColCount:=Fucfitness.fitNjam+1;
  fgCostJam.Cells[1,0]:='Biaya UC';
  fgCostJam.Cells[2,0]:='Biaya PLN';
  fgCostJam.Cells[3,0]:='Selisih Biaya';
  fgHasilGA.RowCount:=gasa.gaMaxgen+1;
  fgHasilGA.Cells[0,0]:='Gen';
  fgHasilGA.Cells[1,0]:='Min';
  fgHasilGA.Cells[2,0]:='Avg';
  fgHasilGA.Cells[3,0]:='Max';
  for i:=1 to FucFitness.fitNgen do
```

```
  begin
    fgStatus.Cells[0,i]:='Gen '+IntToStr(i);
    fgLoad.Cells[0,i]:='Gen '+IntToStr(i);
    fgPLN.Cells[0,i]:='Gen '+IntToStr(i);
  end;
  for i:=1 to FucFitness.fitNjam do
  begin
    fgStatus.Cells[i,0]:='Jam '+IntToStr(i);
    fgLoad.Cells[i,0]:='Jam '+IntToStr(i);
    fgCostJam.Cells[0,i]:='Jam '+IntToStr(i);
    fgPLN.Cells[i,0]:='Jam '+IntToStr(i);
  end;
end;

end.


unit Tampil;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, AxCtrls, OleCtrls, VCF1, ComCtrls;

type
  TfrmTampil = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    lblSumUnit: TLabel;
    lblSumJam: TLabel;
    edtUnit: TEdit;
    edtJam: TEdit;
    TabSheet2: TTabSheet;
    fgUnit: TF1Book;
    TabSheet3: TTabSheet;
    fgBeban: TF1Book;
    TabSheet4: TTabSheet;
    fgPLN: TF1Book;
    Panel1: TPanel;
    btnClose: TButton;
    btnNext: TButton;
    procedure FormActivate(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
    procedure btnNextClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  frmTampil: TfrmTampil;

implementation

uses UnitGenerator, Fitness, Param;

{$R *.dfm}

procedure TfrmTampil.FormActivate(Sender: TObject);
var i,j:integer;
    gen:TGenArr;
begin
  edtUnit.Text:=IntToStr(FucFitness.fitNgen);
  edtJam.Text:=IntToStr(FucFitness.fitNjam);
  fgUnit.MaxRow:=FucFitness.fitNgen+2;
  fgBeban.MaxCol:=FucFitness.fitNjam+1;
  fgPLN.MaxCol:=FucFitness.fitNjam+1;
  fgPLN.MaxRow:=FucFitness.fitNgen+1;
  SetLength(gen,FucFitness.fitNgen+1);
  for i:=1 to FucFitness.fitNgen do
  begin
    gen[i]:=TPembangkit.Create;
  end;
  gen:=FucFitness.fitGen;
  for i:=1 to FucFitness.fitNgen do
  begin
    fgUnit.NumberRC[i+2,1]:=i;
    fgPLN.NumberRC[i+1,1]:=i;
    fgUnit.TextRC[i+2,2]:=gen[i].GenNama;
    fgUnit.NumberRC[i+2,3]:=gen[i].GenPmax;
    fgUnit.NumberRC[i+2,4]:=gen[i].GenPmin;
    fgUnit.NumberRC[i+2,5]:=gen[i].Gena0;
    fgUnit.NumberRC[i+2,6]:=gen[i].Gena1;
    fgUnit.NumberRC[i+2,7]:=gen[i].Gena2;
    fgUnit.NumberRC[i+2,8]:=gen[i].GenTup;
    fgUnit.NumberRC[i+2,9]:=gen[i].GenTdown;
    fgUnit.NumberRC[i+2,10]:=gen[i].GenSh;
    fgUnit.NumberRC[i+2,11]:=gen[i].GenSc;
    fgUnit.NumberRC[i+2,12]:=gen[i].GenTcold;
    fgUnit.NumberRC[i+2,13]:=gen[i].GenIniSt;
  end;
  for i:=1 to FucFitness.fitNjam do
  begin
    fgPLN.NumberRC[1,i+1]:=i;
    fgBeban.TextRC[1,i+1]:='Jam '+IntToStr(i);
    fgBeban.NumberRC[2,i+1]:=FucFitness.fitBeban[i];
    fgBeban.NumberRC[3,i+1]:=FucFitness.fitRes[i];
  end;
```

```
  for i:=1 to FucFitness.fitNgen do
  begin
    gen[i].Free;
    for j:=1 to FucFitness.fitNjam do
    begin
      fgPLN.NumberRC[i+1,j+1]:=PLN[i,j];
    end;
  end;
end;

procedure TfrmTampil.btnCloseClick(Sender: TObject);
begin
  Close;
end;

procedure TfrmTampil.btnNextClick(Sender: TObject);
begin
  frmParam.Show;
end;

end.
```