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

SKRIPSI

ANALISIS KOMITMENT UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI EVOLUTIONARY PROGRAMMING DAN TABU SEARCH PADA PT. PEMBANGKITAN JAWA-BALI

Disusun oleh;

ANTONI ANDI CHAROLI
99.12.143

DESEMBER 2005

LEMBAR PERSETUJUAN

ANALISIS KOMITMENT UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI EVOLUTIONARY PROGRAMMING DAN TABU SEARCH PADA PT. PEMBANGKITAN JAWA-BALI

SKRIPSI

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

Disusun Oleh :

ANTONI ANDI CHAROLI
NIM. 99.12.143

Mengetahui,
Ketua Jurusan Teknik Elektro

Ir.F Yudi Limpraptono, MT
NIP.103 950 0274

Menyetujui,
Dosen Pembimbing

Ir.H.Almizan Abdullah, MSEE
NIP.130 900 0208

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

ABSTRAKSI

ANALISIS KOMITMENT UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI EVOLUTIONARY PROGRAMMING DAN TABU SEARCH PADA PT. PEMBANGKITAN JAWA-BALI

(Antoni Andi Charoli, Nim. 99.12.143, Teknik Elektro Energi Listrik S-1)
(Dosen Pembimbing : Ir.H Almizan Abdullah,MSEE)

Kata Kunci: *Evolutionary Programming, Tabu Search,* Komitmen Unit, Optimasi Biaya Pembangkitan.

Skripsi ini merepresentasikan sebuah penyelesaian masalah komitmen unit menngunakan metode kombinasi *Evolutionary Programming, Tabu Search.* Objek dari skripsi ini adalah untuk mencari penjadwalan yang optimal dari unit-unit pembangkit yang terinterkoneksi dalam system sehingga diinginkan total biaya operasi dapat diminimalisasi dan mencari penjadwalan komitmen unit yang optima untuk jam berikutnya. Metode *Evolutionary Programming* merupakan sebuah teknik optimasi global untuk menyelesaikan masalah komitmen unit, operasi dalam sebuah system, dimana desainnya dengan melakukan pengkodean pada penjadwalan operasi dari unit-unit pembangkit dengan memperhatikan kendala *minimum up/down time*. Penjadwalan komitmen unit dikodekan sebagai sebuah symbol *string* yang membentuk sebuah *chromosome*. Sebuah populasi awal dari solusi *parent* dibangkitkan secara acak, dimana penjadwalan dibentuk dengan meng-*commit* semua unit semua unit menurut *initial status*. disini, *parent* yang dihasilkan dari sebuah set awal penentuan solusi telah disesuaikan dengan kebutuhan. Kemudian dilakukan penjadwalan kembali secara acak yang dilakukan dengan tetap memperhatikan kendala *minimum up/down time*, dan metode *Tabu Search* akan memperbaiki status yang diberikan yang digunakan untuk menghindari jebakan optimum lokal dengan menggunakan struktur memori yang membolehkan kembali ke solusi sebelumnya(bachtracking). Solusi yang terbaik akan diseleksi dengan menggunakan *Evolutionary Strategy*. Analisa dilakukan dengan bantuan program komputer dengan menggunakan bahasa pemrograman Delphi versi 7.0 dan diterapkan pada PT. PJB dengan jumlah unit pembangkit 37 unit pembangkit termal. Hasil perhitungan ditunjukkan dengan melakukan perbandingan besarnya penghematan total biaya pembangkitan antara PT. Pembangkitan Jawa Bali dengan metode kombinasi *Evolutionary Programming, Tabu Search* pada tanggal 27 juli 2005 sebesar 18 %, pada tanggal 30 Juli 2005 sebesar 20 %, pada tanggal 31 juli 2005 sebesar 21 %. Besarnya waktu eksekusi kurang dari 7 menit sehingga metode kombinasi *Evolutionary Programming, Tabu Search* dapat diterapkan untuk menyelesaikan masalah komitmen unit.

KATA PENGANTAR

Dengan rahmat Allah SWT, penulis mengucapkan syukur kehadirat-Nya atas karunia yang dilimpahkan serta menghaturkan sholawat serta salam selalu dilimpahkan kepada nabi besar Muhammad SAW sehingga dapat menyelesaikan skripsi yang berjudul "**ANALISIS KOMITMENT UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI EVOLUTIONARY PROGRAMMING DAN TABU SEARCH PADA PT. PEMBANGKITAN JAWA-BALI**".

Skripsi ini bertujuan untuk memenuhi kurikulum akademik yang harus ditempuh oleh setiap mahasiswa ITN Malang dalam menempuh sekaligus mengakhiri pendidikan pada jenjang S-1 pada Jurusan Teknik Elektro Konsentrasi Teknik Energi Listrik Saya menyadari bahwa skripsi ini masih jauh dari kesempurnaan, karena itu saran dan kritik membangun sangat saya harapkan.

Atas segala bimbingan, pengarahan dan bantuan yang diberikan, sehingga tersusun skripsi ini, maka penulis menyampaikan terima kasih kepada;

1. Bapak Dr. Ir.Abraham Lomi, MSEE, selaku Rektor ITN Malang.
2. Bapak Ir. F Yudi Limpraptono, MT, selaku Kepala Jurusan Teknik Elektro ITN Malang.
3. Bapak Ir. H. Almizan Abdullah, MSEE, selaku Dosen Pembimbing.
4. Bapak – bapak karyawan bagian personalia dan lapangan di PT. Pembangkitan Jawa-Bali.

5. Semua pihak yang telah memberikan bantuan kepada penulis dalam penyusunan skripsi ini yang tidak dapat penulis sebutkan satu per satu

Akhirnya saya mengharapkan skripsi ini berguna dan bermanfaat bagi rekan mahasiswa khususnya Jurusan Teknik Elektro.

Malang, September 2005

Penyusun

بسم الله الرحمن الرحيم

Fisrt of all I would like to say Alhamdulillah Robbil' alamiin to Alloh S.W.T. for the opportunity given to me to finish this thesis. Besides, I would like to dedicate this thesis to my beloved father and mother.....

Second I always mention prophet of Muhammad in each;every my prayer because he have guided our to step go to truth and respect endless heaven him

Third, this Skripsi dedicate to my Mother and Father which always pray me, giving its affection for the to go to of my aspiration.

All my family, my wife, and all side which have supported good me of items and also prayer.

بسم الله الرحمن الرحيم

SPECIAL THANKS FOR.....

Bapak Ir. H. Almizan Abdullah, MSEE makasih atas nasehat-nasehat, masukan-masukan, dan bimbingannya baik sebagai dosen Wali maupun sebagai Dosen Pembimbing. Makasih juga atas ilmu pengetahuan yang bapak berikan pada saya mudah-mudahan selalu diberikan kesehatan jasmani maupun rohani oleh Tuhan Y.M.E.

Bapak dan ibu dosen JTN MALANG khususnya Teknik Elektro Energi Listrik mudah-mudahan selalu diberikan kesehatan jasmani maupun rohani oleh Tuhan Y.M.E.

Buat Mas Ugro, makasih ya....mes atas bantuannya bualin program Delphi...friendship for you...
All fiesy angkatan 99 yang gak bisa aku sebutin atu-atu makasih banyak atas bantuannya...

Damel rewang rencang kos-kosan sumbersari 229 B yang gak bias disebut disini jang marah ya...atu mas ucapan THANKS FOR A LOT....

بسم الله الرحمن الرحيم

SPECIAL THANKS FOR

Tuk temen temen kontrakan JL Gajayana gang I 739 jj ono Nanang alias gondrng ending lolos lee...sakne mbok pakmu nek ngopeni ojo mung turaturu wae, gawe Jipang, Gentho, Mas Mbah sing sregep nek garap skripsi ojo putus asa..semangat!!!!, diaturaken kagem pak lek Kuennnthiiiiltttt...gek endang ajokne jurnal teko ieee, nyonya thit..wis ngenteni tho.. kagem mas cueeepoxxx, mengong, karo gundol wulong nek sregep nek kuliah ojo tilip absent alias TA wae...buat Risky jangan lupa sepuluh menit lagi.. n Kipunx jo mbok tiru sing elek elek teko kakanganmu yo...

Dik Neni, Dik Reni n Farida nek sekolah sing sregep ojo mung miderrrr wae...n ojo manja wis gede wong wis ngerti gendha'an jek manja wae..

Tuk YAYANGKOE tersayang tunggulah diriku ini, jangan kau lepaskan diriku dari pelukanmu yang hangat, kan.. kubawa dirimu menuju singgahsana sehari yang penuh keindahan, kebahagiaan, dan semerbak bunga musim semi...

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR GRAFIK

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan system tenaga listrik dari waktu ke waktu semakin meningkat seiring dengan kemajuan teknologi dan tingkat kesejahteraan hidup masyarakat yang membutuhkan pasokan energi listrik yang akan terus meningkat. Namun, mengingat penyediaan tenaga listrik membutuhkan teknologi yang relative canggih dan waktu pembangunan pusat tenaga listrik yang relative lama maka dapat difahami bila kebutuhan biaya pembangunan ketenagalistrikan sangat besar. Dari hal-hal tersebut diatas, maka PT. Pembangkitan Jawa Bali(PT. PJB) sebagai produsen tenaga listrik harus semakin efisien dan andal dalam melakukan operasional penyediaan tenaga listrik yang merupakan satu kesatuan yang terintegrasi antara pembangkit listrik, gardu induk(pusat beban) yang satu dengan lainnya dihubungkan dengan menggunakan jaringan tranmisi sehingga penyediaan tenaga listrik dapat berjalan lancar secara kontinyu sesuai dengan permintaan beban.

Penentuan komitmen unit yang efisien berperan penting dalam pengoperasian system tenaga listrik yang optimal, andal, dan ekonomis. Untuk mencapai hasil yang optimal, handal, dan ekonomis maka, perlu melakukan perencanaan penjadwalan unit pembangkit termal untuk memenuhi kebutuhan beban yang selalu berfluktuasi setiap waktu, yaitu dengan menentukan unit pembangkit mana yang harus dioperasikan untuk memenuhi kebutuhan beban dan unit mana yang tidak beroperasi/dimatikan. Hal ini dikarenakan biaya bahan baker

pada umumnya merupakan biaya terbesar dalam pengoperasian unit pembangkit termal. Oleh karenanya diperlukan suatu cara untuk menghitung biaya bahan baker yang diperlukan dalam pengoperasian system tenaga listrik untuk kurun waktu tertentu.

Beberapa hal yang perlu dicatat dalam melakukan perhitungan tersebut diatas adalah:

1. Perlu diingat adanya kendala-kendala atau syarat batas pembebanan minimum dan pembebanan maksimum untuk setiap unit pembangkit.
2. Perhtungan dilakukan mulai dari unit pembangkit terkecil sampai unit pembangkit terbesar dengan tetap memperhatikan *minimum up time* dan *minimum down time*.
3. Biaya *start up* dan biaya *shut down* dari unit pembangkit termal dapat ditambahkan setelah perhitungan biaya bahan bakar yang minimum sehingga didapatkan biaya total pembangkitan berdasarkan penjadwalan operasi unit pembangkit.

Metode dasar yang digunakan untuk menyelesaikan masalah komitmen unit antara lain adalah *metode skala prioritas, branch and bound, dynamic programming, genetic algorithm, dan lagrangian relaxation*. Sehingga dalam melakukan operasi komitmen unit melibatkan seleksi biaya *start up* yang paling ekonomis dan waktu *shut down* untuk penjadwalan unit pembangkit termal dengan menggunakan metode kombinasi evolutionary programming dan tabu search. Implementasi dari *evolutionary programming* digunakan sebagai masukan atau inputan dari metode tabu search. Metode tabu search berupaya untuk menghadapi bahaya pada optimum lokal dengan menggunakan struktur memori

yang melarang atau menghentikan gerakan tertentu yang akan kembali ke solusi yang telah digunakan atau solusi sebelumnya, karena algoritma ini dapat menyimpan informasi mengenai pergerakan berupa solusi-solusi terbaik untuk dimasukkan dalam daftar tabu(*tabu list*) yang digunakan untuk mencegah *hacktracking* atau kondisi dimana pergerakan menjauhi titik optimum global dan membimbing langkah agar selalu menuju kesuatu titik optimum global.

Metode ini mempunyai banyak keunggulan yang dapat diperoleh dalam penerapannya, diantaranya adalah untuk menghitung tabel penjadwalan operasi unit pembangkit, tidak perlu menyelesaikan persamaan koordinasi, sementara pada waktu yang bersamaan jumlah uji coba terhadap kombinasi terhadap unit pembangkit dapat berkurang, selain itu hasil yang didapatkan merupakan nilai dengan kondisi menjauhi titik optimum lokal dan menuju kesuatu titik optimum global.

Pada skripsi ini menyajikan metode kombinasi evolutionary programming dan tabu search untuk menyelesaikan masalah komitmen unit yang algoritma penyelesaiannya ditujukan pada system operasi yang didesain untuk menentukan penjadwalan operasi dari unit pembangkit.

1.2 Rumusan Masalah

Dalam mengoperasikan pusat-pusat pembangkit listrik, diharapkan adanya suatu optimalisasi dari biaya operasional pembangkit. Untuk itu diperlukan penentuan penjadwalan unit pembangkit yang ekonomis untuk dioperasikan. Salah satu metode yang akan dipakai adalah Metode kombinasi *evolutionary programming dan tabu search.*

Berdasarkan gambaran permasalahan tersebut maka skripsi ini diberi judul **"ANALISIS KOMITMEN UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH* PADA PT. PEMBANGKITAN JAWA BALI"**

1.3 Tujuan

Berdasarkan permasalahan yang telah dikemukakan diatas, maka skripsi ini bertujuan:

1. Untuk menentukan penjadwalan unit pembangkit termal yang akan melayani kebutuhan beban yang berubah tiap jamnya metode kombinasi *evolutionary programming dan tabu search.*
2. Untuk mengevaluasi dan mengoptimalkan biaya operasional system pada nilai minimum dengan menggunakan metode kombinasi *evolutionary programming dan tabu search.*
3. Untuk mendapatkan suatu perencanaan dan penjadwalan pembangkitan yang optimal, efisien, efektif, dan ekonomis baik pada komitmen unit skala kecil maupun komitmen unit skala besar.

1.4 Batasan Masalah

Skripsi ini akan dilakukan analisa tentang perencanaan komitmen unit mengunakan metode kombinasi *evolutionary programming dan tabu search* dengan mengambil sistem pembangkit tenaga listrik termal yang dimiliki oleh PT. Pembangkitan Jawa-Bali (PT.PJB) sebagai objek utama dalam penelitian. Pembahasan masalah dibatasi sebagai berikut :

- Tidak membahas masalah rugi-rugi saluran transmisi.
- Perhitungan dan analisa dilakukan dalam jangka waktu 3 hari pada PT. pembangkitan jawa bali(PT. PJB)
- Penjadwalan dilakukan dalam satu hari dalam jangka waktu 24 jam dengan range tiap jam pada periode studi tanggal 27, 30, dan 31Juli 2005
- Tidak membahas *combined cycle* pada PLTGU.
- Tidak membahas masalah biaya cadangan berputar (*spinning reverse*) dan hanya memperhatikan kendala batasan cadangan berputar.
- Tidak membahas unit hydro, karena memerlukan perhitungan yang terpisah dan menggunakan metode yang lain.
- Tidak membahas kendala tingkat kenaikan daya
- Tidak membahas kendala status harus operasi
- Pembahasan dititik beratkan pada segi ekonomis daripada segi teknis
- Untuk ST (*steam turbin)* pada *combined cycle,* diambil data parameter dari pola PLTGU CC-3.3.1 yang beroperasi.
- Tidak ada biaya *shutdown* unit.
- Data penawaran diambil data penawaran dari PT. PLN Pembangkitan Jawa Bali pada periode Agustus 2002.

1.5 Metode Pemecahan Masalah

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

1. Studi literatur: berupa pengumpulan referensi metode evolutionary programming dan metode tabu search

2. Pengambilan data lapangan yang dipakai sebagai objek penelitian yaitu data bahan bakar, data pembebanan harian, data pembangkitan yang meliputi *minimum up time dan minimum down time, maksimum generation*(P_{mak}) dan *minimum generation*(P_{min}), dan kapasitas daya terpasang dari unit pembangkit termal.
3. Pembuatan program menggunakan program Delphi 8.0
4. Validasi program menggunakan jurnal IEEE Transaction On Power System, yaitu **an evolutionary programming based tabu search method for Solving Unit Commitment Problem**, IEEE Trans. On Power System, Vol. 19, No. 1, Februari 2004.
5. Melakukan analisa data dan evaluasi sehingga dapat disimpulkan apakah metode ini efektif untuk menekan biaya operasi pembangkitan dan diterapkan dalam operasi pembangkitan PT. Pembangkitan Jawa Bali(PT. PJB)

1.6 Sistematika Pembahasan

Bab I : Pendahuluan

Dalam bab ini diuraikan latar belakang, rumusan masalah, tujuan, batasan masalah, metode pemecahan masalah, sistematika pembahasan, dan kontribusi.

Bab II : Landasan Teori

Dalam bab ini diuraikan tentang dasar komitmen unit, pembebanan ekonomis(*economic dispatch*), dan teori tentang metode Evolutionary Programming dan metode Tabu search

Bab III : Aplikasi Metode Kombinasi *Evolutionary Programming* dan metode *Tabu search* pada komitmen unit di PT. Pembangkitan Jawa Bali(PT. PJB)

Dalam bab ini diuraikan tentang penggunaan metode kombinasi *Evolutionary Programming dan metode Tabu search* untuk menyelesaikan komitmen unit di PT. Pembangkitan Jawa Bali(PT. PJB)

Bab IV : Analisa Data dan Hasil Simulasi Program

Dalam bab ini diuraikan tentang seluruh hasil simulasi program untuk menyelesaikan masalah komitmen unit serta hasil perhitungan, analisa, dan evaluasi.

Bab V : Kesimpulan dan Saran

Diambil dari hasil analisa komitmen unit dalam sistem tenaga listrik

1.7 Kontribusi

Dengan adanya analisa ini nantinya diharapkan dapat memberikan alternatif terbaik dalam pemecahan permasalahan komitmen unit pembangkit thermal yang lebih mudah dengan waktu yang lebih singkat, sehingga kemungkinan dapat diaplikasikan dalam PT. Pembangkitan Jawa Bali(PT. PJB).

BAB II

TEORI KOMITMEN UNIT, ECONOMIC DISPATCH, DAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING DAN TABU SEARCH*

2.1 Sistem Tenaga Listrik[2][3][4][5]

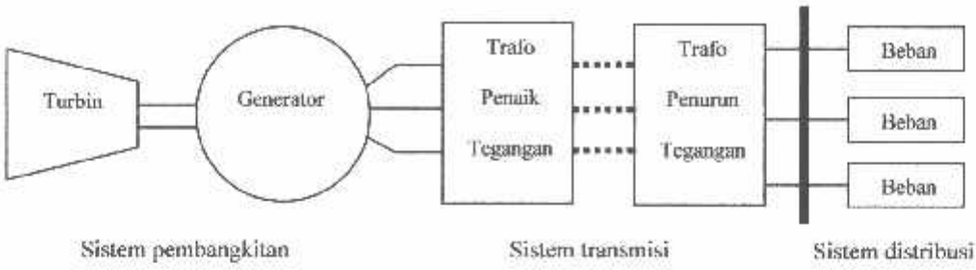
Karena berbagai persoalan teknis, tenaga listrik hanya dapat dibangkitkan pada lokasi tertentu. Mengingat pemakai tenaga listrik atau pelanggan listrik tersebar diberbagai tempat, maka penyaluran tenaga listrik dari pusat pembangkit tenaga listrik sampai ke pemakai atau pelanggan tenaga listrik memerlukan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

Tenaga listrik dibangkitkan di pusat-pusat listrik seperti PLTA, PLTD, PLTU, PLTG, dan PLTGU, kemudian disalurkan melalui saluran transmisi setelah tegangan dinaikkan terlebih dahulu oleh Transformator penaik tegangan(Transformator *Step-up*) yang terdapat di pusat listrik.

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

Jaringan setelah melalui atau keluar gardu induk umumnya disebut jaringan distribusi, sedangkan jaringan dari pusat pembangkit tenaga listrik dengan gardu induk disebut jaringan transmisi. Setelah melalui saluran distribusi primer, maka tenaga listrik kemudian diturunkan tegangannya oleh transformator distribusi

menjadi tegangan 380/220 Volt atau 220/127 Volt, dan baru kemudian disalurkan kepada konsumen.



GAMBAR 2.1 ELEMEN POKOK SISTEM TENAGA LISTRIK[4]

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

Untuk memenuhi kebutuhan tenaga listrik bagi para konsumen, diperlukan berbagai peralatan listrik. Berbagai peralatan listrik tersebut dihubungkan satu dengan yang lainnya secara keseluruhan membentuk suatu sistem tenaga listrik, sehingga sistem tenaga listrik yang dimaksud disini adalah sekumpulan pusat pembangkit tenaga listrik dan gardu induk yang satu dengan yang lainnya dihubungkan oleh jaringan transmisi sehingga merupakan sebuah kesatuan interkoneksi.

Karena daya listrik yang dibangkitkan harus sama dengan tenaga listrik yang dibutuhkan konsumen, maka dalam mengoperasikan sistem tenaga listrik dengan baik perlu adanya hal-hal sebagai berikut:

a. Perencanaan operasi
b. Pelaksanaan operasi
c. Pengendaliaan operasi
d. Analisa operasi

Operasi sistem tenaga listrik ini menyangkut biaya opersi yang besar, maka manajemen operasi sistem tenaga listrik harus memikirkan bagaimana menyediakan tenaga listrik yang seekonomis mungkin dengan tetap memperhatikan mutu dan keandalan. Sehingga hal-hal yang perlu diperhatikan dalam manajemen operasi sistem tenaga listrik adalah sebagai berikut:

a. Perkiraan beban
b. Syarat-syarat pemeliharaan peralatan
c. Keandalan yang diinginkan
d. Pengaturan dan penyaluran beban
e. Proses produksi tenaga listrik yang ekonomis

Kelima hal diatas masih harus seringkali dikaji ulang terhadap berbagai kendala seperti:

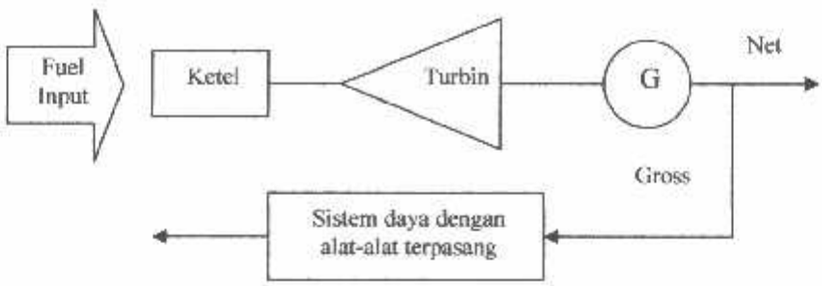
a. Aliran beban dalam jaringan
b. Daya hubung singkat
c. Gangguan yang sering kali menimpa peralatan
d. Stabilitas sistem
e. Penyediaan suku cadang dan dana

Dengan memperhatikan kendala-kendala diatas, maka seringkali harus dilakukan pengaturan kembali terhadap rencana pemeliharaan dan alokasi beban. Maka besar suatu sistem tenaga listrik, makin banyak unsure yang harus dikoordinasikan serta yang harus diamati. Sehingga perlu perencanaan, pelaksanaan, pengendalian, dan analisa operasi system tenaga listrik yang cermat.

2.2 Karateristik Pembangkit Tenaga Listrik[2]

Hal yang paling mendasar dalam optimasi ekonomi dari sebuah pembangkit listrik termal adalah dengan ditentukannya karateristik masukan - keluaran (*input-output characteristic*) dari suatu pusat pembangkit tenaga listrik tersebut. Dalam mendefinisikan karateristik masukan-keluaran, akan dibicarakan tentang masukan kotor (*gross input*) dan keluaran bersih (*net output*). Masukan kotor (*gross input*) pembangkit listrik termal menyatakan jumlah keseluruhan bahan bakar yang diperlukan, sedangkan keluaran bersih (*net output*) merupakan daya nyata (*real power*) yang dihasilkan generator.

Tipe sebuah pembangkit listrik termal dapat dilihat pada gambar 2.2 dibawah ini. Bagan tersebut terdiri atas sebuah boiler/ketel yang menghasilkan uap untuk menggerakkan turbin uap yang dikopel dengan sebuah generator listrik. Daya listrik yang dihasilkan tidak seluruhnya disalurkan ke sistem tetapi sebagian kecil dipakai untuk mengoperasikan peralatan-peralatan listrik yang terdapat pada puast pembangkit tenaga listrik tersebut, seperti boiler/ketel, pompa, kompresor, serta mencatu peralatan kontrol, komunikasi, penerangan, komputer, dan sebagainya.



GAMBAR 2.2 UNIT BOILER – TURBIN – GENERATOR[3][4]

2.2.1 Karateristik Masukan-Keluaran[2]

Masukan sebuah pembengkit listrik termal umumnya dinyatakan sebagai banyaknya energi per satuan waktu dari bahan bakar yang diberikan ke boiler/ketel untuk menghasilkan daya listrik yang merupakan keluaran dari pusat pembangkit tenaga listrik tersebut. Hal tersebut diatas dapat dinyatakan sebagai:

$F = H \times \$US/Btu$ 2.1

Dimana : H : jumlah energi per satuan waktu[MBtu/hour]

F : jumlah harga bahan bakar per satuan waktu[$US/hour]

$US/Btu : harga bahan bakar per satuan energi yang dikandung bahan bakar

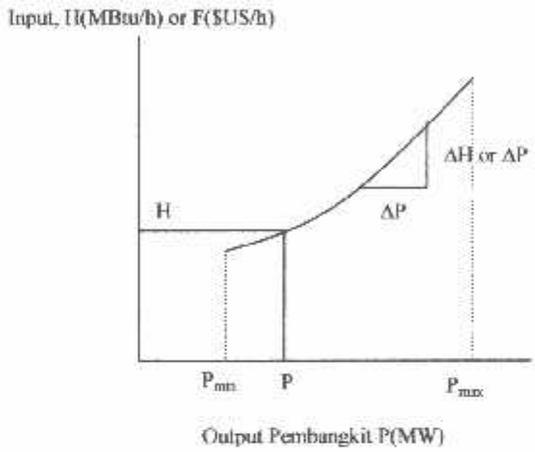
Sedangkan keluaran dari pembangkit listrik termal adalah daya nyata yang dihasilkan oleh generator listrik dikurangi dengan daya nyata yang dipakai oleh pusat listrik tersebut. Notasi yang umum dipakai adalah:

P dengan satuan MW

Jadi dapat disimpulkan bahwa masukan pusat listrik merupakan fungsi terhadap keluarannya, sehingga hubungan tersebut diatas dapat ditulis sebagai berikut:

$H = f(P)$[MBtu/hour] atau $F = f(P)$ [$US/hour] 2.2

Kurva dari karateristik masukan-keluaran dari sebuah pembangkit listrik termal yang telah diidealkan dengan berpedoman atas dasar fungsi biaya bahan bakar(F = f(P) [$US/hour]) ditunjukkan pada gambar 2.3. Masukan adalah sebagai ordinat, yang dinyatakan banyaknya energi yang diperlukan per satuan waktu[MBtu/h] atau besarnya biaya bahan bakar yang dikonsumsi per satuan waktu[$US/h]. Keluaran adalah sebagai absisnya, yang menyatakan daya listrik[MW] yang dihasilkan untuk melayani beban sistem.



GAMBAR 2.3 KURVA KARATERISTIK INPUT-OUTPUT PEMBANGKIT TERMAL[2]

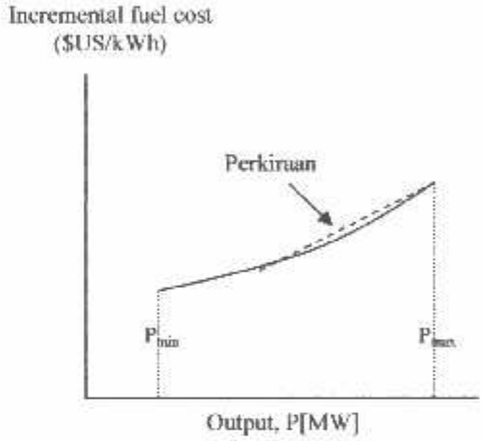
Data yang diperlukan untuk menggambarkan kurva karateristik input-output pembangkit termal dapat diperoleh dari perhitungan pada saat perencanaan atau tes yang telah dilakukan terhadap unit pembangkit termal yang bersangkutan.

2.2.2 Karateristik Laju Pertambahan Biaya Bahan Bakar[2]

Karateristik laju pertambahan biaya bahan bakar atau *Incremental Fuel Cost Characteristic* adalah turunan pertama dari fungsi biaya bahan bakar F[$US/h] terhadap tingkat pembebanan P [MW] dari pusat pembangkit listrik yang

bersangkutan. Fungsi ini menunjukkan tingkat besarnya kenaikan atau penurunan biaya bahan bakar untuk setiap tingkat perubahan beban.

Fungsi biaya bahan bakar secara luas digunakan untuk menentukan pembebanan ekonomis(*economic dispatch*) dari setiap pembangkit listrik termal. Kurva laju pertambahan biaya bahan bakar yang telah diidealkan dari sebuah pembangkit listrik termal dapat dilihat pada gambar dibawah ini:



GAMBAR 2.4 KURVA KARATERISTIK LAJU PERTAMBAHAN BIAYA BAHAN BAKAR[2]

2.3 Komitmen Unit[2][6]

Dalam suatu siklus waktu, misalnya siklus harian yang terbagi dalam interval waktu 1 jam selama 24 jam, beban listrik dalam sistem tenaga selalu berubah. Operasi pusat-pusat pembangkit di dalam sistem tenaga harus selalu dikoordinasikan dalam pembagian pembebanan secara optimal atau ekonomis pada setiap perubahan beban dalam setiap inteval waktu untuk siklus waktu tertentu. Oleh karena itu diperlukan penjadwalan operasi unit-unit pembangkit dalam melayani beban sistem selama periode waktu tertentu, atau dikenal dengan istilah komitmen unit.

Pada masalah komitmen unit diasumsikan ada sejumlah N unit pembangkit yang tersedia dan diopersikan untuk memenuhi permintaan beban, sehingga dapat disimpulkan bahwa bila tersedia N unit pembangkit yang diharapkan mencukupi permintaan beban, masalahnya adalah unit-unit pembangkit mana yang seharusnya dioperasikan dengan hasil seekonomis mungkin.

Beberapa metode yang digunakan dalam menyelesaikan masalah komitmen unit, antara lain:

- **Metode Lagrange Relaxation**

  Pada penjadwalan unit pembangkit yang akan beroperasi tiap periode waktu dalam jam, pembagian bebannya harus dalam keadaan optimal atau seekonomis mungkin. Dari jumlah unit pembangkit yang banyak, maka untuk menentukan unit-unit mana yang harus beroperasi dan tidak beropersi pada jam-jam tertentu. Penentuan unit-unit mana yang harus beroperasi dan tidak beropersi dapat diperhitungkan dengan membuat kombinasi dari unit-unit yang ada dan berada pada keadaan operasi yang optimal, baru kemudian dipilih kombinasi unit-unit pembangkit mana yang termurah biaya operasinya.

  Lagrange relaxation adalah Suatu metode yang didasarkan pada pendekatan optimasi ganda(dual optimization) yang berhubungan dengan pengali lagrange(lagrange multiplier), yang dapat dipakai untuk menentukan kombinasi yang paling optimal sehingga didapatkan biaya operasi suatu unit pembangkit yang paling ekonomis. Pengali lagrange digunakan pada persamaan batasan pembebanan dengan melalui proses subgradien sehingga dapat memaksimumkan nilai ganda(dual value).

Dalam penyelesaian atau solusi lagrange relaxation pada masalah komitmen unit, digunakan bentuk penguraian yang berlainan untuk memudahkan dalam memutuskan penjadwalan unit-unit pembangkit dalam horizon waktu perencanaan, kemudian tiap-tiap unit pembangkit ini dapat dikombinasikan apakah sudah memenuhi batasan yang diinginkan

- **Metode Priority list**

  Merupakan metode yang membuat urutan prioritas dari unit pembangkit dari unit pembangkit yang mempunyai biaya pembangkitan terkecil sampai dengan unit pembangkit yang mempunyai biaya pembangkitan terbesar. Urutan prioritas tersebut dapat diperoleh dengan melakukan perhitungan berdasarkan besarnya biaya rata-rata beban penuh yang didasarkan pada pemakaian bahan bakar pada beban penuh dari tiap-tiap unit pembangkit yang mempunyai karateristik sama.

- **Metode Dynamic Programming**

  Merupakan teknik pencarian solusi yang optimal secara bertahap. Hasil dari tahap sebelumnya merupakan dasar untuk mencari solusi yang optimal pada tahap berikutnya. Implementasi dari metode *dynamic programming* sudah terbukti efektif dapat dilakukan untuk sistem dengan ukuran sedang, karena untuk sistem yang besar jumlah perhitungannya akan menjadi besar sekali.

2.3.1 Kendala-kendala komitmen unit[1][2][6]

Dalam penyelesaian komitmen unit sangat tergantung dari kendala-kendala yang dimasukkan untuk memperoleh fungsi tujuan. Beberapa kendala

dapat ditempatkan pada permasalahan komitmen unit, karena setiap karateristik sistem daya, pengumpulan daya, keandalan sistem, dan sebagainya dapat menimbulkan aturan yang berbeda dalam penjadwalan unit-unit pembangkit listrik. Penjadwalan unit-unit pembangkit sangat dipengaruhi oleh kemampuan pembangkit dan karateristik beban. Pengoperasian unit-unit pembangkit untuk memenuhi kebutuhan beban sistem terdapat berbagai kendala yang merupakan syarat-syarat batas. Syarat-syarat batas(constraints) tersebut antara lain:

- Cadangan berputar(spinning reverse)

  Cadangan berputar merupakan cadangan daya yang harus diperhitungkan dari setiap unit-unit pembangkit yang beroperasi. Tujuan atau kegunaan dari cadangan berputar adalah apabila ada salah satu unit pembangkit mengalami kegagalan operasi, maka harus ada cukup cadangan daya untuk mengganti atau menutupi berkurangnya suplai daya pada periode waktu tertentu. Cadangan berputar dialokasikan menurut aturan yang telah ditetapkan besarnya, biasanya sekian persen(%) dari kebutuhan beban puncak. Umumnya cadangan berputar diperhitungkan untuk mampu mengganti apabila ada unit terbesar mengalami kegagalan opersi.

- Kendala-kendala unit termal

  Unit-unit pembangkit termal dapat mengalami perubahan terperatur kerjanyasetahap demi setahap, dan ini diterjemahkan dalam periode waktu dalam jam, dimana dibutuhkan energi untuk membawa unit tersebut on line atau beroperasi. Hal ini menyebabkan beberapa kendala-kendala antara lain:

1. Minimum up time

   Minimum up time adalah interval waktu minimum, dimana suatu unit yang dihidupkan tidak boleh segera dimatikan kembali sebelum melewati batas waktu yang diijinkan.

2. Minimum down time

   Minimum down time adalah interval waktu minimum, dimana suatu unit dimatiakan tidak boleh segera dihidupkan kembali sebelum melewati batas waktu minimum yang diijinkan.

3. Status harus beroperasi

   Beberapa unit pembengkit yang diberi status harus beropersi beberapa kali dalam setahun yang tujuannya adalah untuk menstabilkan tegangan pada jaringan transmisi atau sebagai suplai daya diluar pemakaian sendiri.

4. Kendala pembangkit hidro

   Biaya operasi pembangkit hidro lebih kecil dibandingkan pembangkitan termal karena pembangkit hidro tidak ada proses pembakaran. Dilain pihak pembangkitan hidro tergantung perubahan musim, sehingga merupakan kendala. Kendala hidro pada masalah komitmen unit tidak dapat dipisahkan dari unit hidro, tapi koordinasi hidrotermal dapat dipisahkan dari masalah komitmen unit

2.3.2 Biaya start up[2][6]

Biaya start up adalah biaya yang diperlukan oleh pembangkit untuk start dari keadaan tidak beropersi menjadi pembangkit beroperasi dan terhubung pada sistem tenaga listrik. Ada dua macam biaya start antara lain:

a. Biaya start saat kondisi dingin

Biaya start saat kondisi dingin terjadi pada saat pembangkit lepas dari sistem atau tidak beroperasi, sedangkan temperatur boiler dibiyarkan turun dari temperatur kerjanya. Sehingga pada saat unit tersebut ingin kembali beroperasi perlu dilakukan pemanasan ulang atau kembali sampai temperatur operasi pada waktunya untuk penyalaan kembali yang telah terjadwalkan .

b. Biaya start saat kondisi panas

Biaya start saat kondisi panas terjadi bila suatu unit pembangkit dilepas dari sistem atau tidak beroperasi, temperatur boiler tetap dipertahankan pada temperatur kerja.

2.3.3 Kendala bahan bakar[6]

Kendala bahan bakar terjadi, apabila dalam sistem terdapat unit-unit pembangkit yang memiliki bahan bakar dalam jumlah terbatas atau memerlukan bahan bakar dalam jumlah besar, sehingga perlu diperhatiakn dalam komitmen unit.

2.3.4 Biaya bahan bakar[6]

Biaya bahan bakar merupakan unsur biaya yang paling besar dan terpenting dalam operasi pembangkitan termal. Fungsi biaya bahan bakar untuk tiap unit pembangkit terhadap daya keluaran dinyatakan dalam bentuk fungsi kuadrat sebagai berikut:

$$F_i(P_{it}) = a_i + b_iP_{it} + c_iP_{it}^2 \quad \text{.....2.3}$$

Dimana : a_i, b_i, c_i = konstanta persamaan dari unit ke-i

P_{it} = daya keluaran dari unit ke-i pada jam t

2.3.5 Fungsi Objektif dan Kendala Komitmen Unit[1][2]

Sasaran dari masalah komitmen unit adalah meminimalkan biaya total operasi dalam penjadwalan unit pembangkit. Oleh karena itu, fungsi obyektif dinyatakan sebagai jumlah dari fungsi biaya bahan bakar dan biaya start up dari unit pembangkit, dapat dituliskan dengan :

$$F(P_{it},U_{it}) = \sum_{t=1}^{T}\sum_{i=1}^{N}[F_i(P_{it}) + ST(1-U_{i,t-1})]U_{it} \quad \text{.....2.4}$$

Dengan $F_i(P_{it}) = a_i + b_iP_{it} + c_iP_{it}^2$

Dimana :

$F_i(P_{it})$ = fungsi biaya bahan bakar unit ke-i dengan a_i, b_i dan c_i adalah konstanta persamaan dari unit ke-i

P_{it} = Daya keluaran dari unit ke-i pada jam t

S_{Ti} = Biaya start up dari unit ke-i

U_{it} = Status on atau off dari unit ke-i pada jam t, $U_{it} = 0$ ketika unit off dan $U_{it} = 1$ ketika unit on

N = Jumlah unit

T = Jumlah jam

Sehubungan dengan minimalisasi dari total biaya operasi, terdapat berbagai kendala atau syarat batas sebagai berikut :

a) Batas pembebanan

$$\sum_{i=1}^{N} P_{it}U_{it} = D_t \quad \text{untuk t = 1,...T} \text{.....2.5}$$

Dimana : D_t = Kebutuhan beban pada jam t

b) Batasan cadangan berputar

$$\sum_{i=1}^{N} U_{it} P_i^{max} \geq D_t + R_t \quad \text{..... 2.6}$$

Dimana : R_t = cadangan berputar pada jam t

c) Batasan pembangkitan

$$U_{it} P_i^{min} \leq P_i \leq U_{it} P_i^{max} \quad \text{..... 2.7}$$

untuk i = 1,...N dan t = 1,...T

d) Minimum up time dan minimum down time

$$T_{on\,i} \geq T_{up\,i} \quad \text{..... 2.8}$$

$$T_{off\,i} \geq T_{down\,i}$$

Untuk I = 1,...N dan t = 1,...T

Dimana : t_{up} = *Minimum up time*

t_{down} = *Minimum down time*

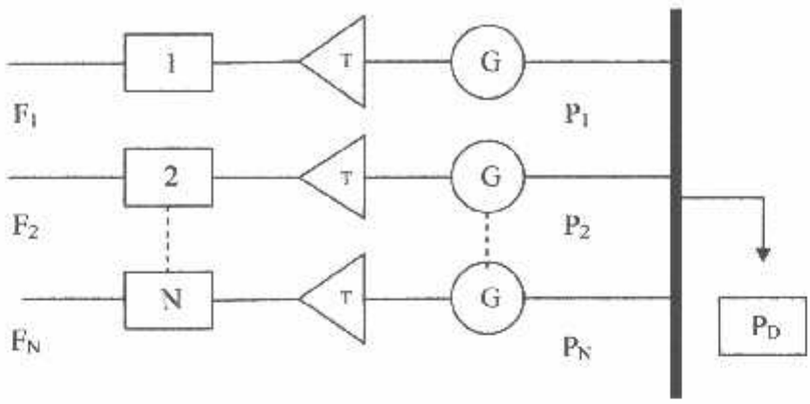
T_{on} = durasi waktu untuk unit i secara kontinyu beroperasi (on)

T_{off} = durasi waktu untuk unit i secara kontinyu tidak beroperasi (off)

2.4 Pembebanan Ekonomis Pembangkit Listrik[2][3][6]

Pembebanan ekonomis atau *economic dispatch* adalah pembebanan pada pembangkit-pembangkit yang ada dalam sistem tenaga listrik, secara optimal dan ekonomis pada beban tertentu. Karena komponen terbesar dari biaya pembangkitan adalah biaya bahan bakar, maka pembebanan ekonomis dibatasi

pada biaya bahan bakar. Oleh karena itu dengan dilakukannya *economic dispatch* berarti pula didapat biaya bahan bakar pembangkitan yang paling murah.



GAMBAR. 2.5 N UNIT PEMBANGKIT THERMAL MELAYANI BEBAN P_D

Sistem tenaga listrik dengan mengabaikan rugi-rugi transmisi dapat dilihat pada gambar 2.5. Sistem ini memperlihatkan pembangkitan termal yang terdiri atas N buah unit yang dihubungkan pada sebuah bus bar untuk melayani total beban sebesar P_D. Masukan untuk setiap unit ke-i adalah F_i yang menyatakan tingkat biaya bahan bakar dari masing-masing unit, dan daya keluaran dari masing-masing unit P_i adalah daya listrik yang dibangkitkan oleh tiap-tiap unit. Biaya total F_T yang ditanggung sistem adalah jumlah biaya dari tiap-tiap unit pembangkit. Dan batasan yang paling penting dari pengoperasiaan pembangkit termal tersebut adalah daya listrik yang dihasilkan harus sama dengan besarnya beban konsumen. Oleh karena setiap pembangkit tenaga listrik memikul beban yang selalu berubah setiap periode waktu tertentu, maka perhitungan pembebanan ekonomis dilakukan pada setiap harga beban tertentu. Jika ditinjau dari selang waktu yang cukup kecil dan dalam selang waktu tersebut beban sistem, rugi-rugi dalam sistem, dan juga rugi-rugi beban unit-unit pembangkit termal dianggap

konstan, maka biaya bahan bakar dalam selang waktu tersebut Δt adalah sebagai berikut:

$$F_{\Delta t} = \sum_{j=1}^{j=n} F_j(P_{Tj}) * \Delta t \quad \text{.....2.9}$$

Dimana : $F_{\Delta t}$ = biaya bahan bakar dalam sistem selama selang waktu Δt

F_jP_{Tj} = biaya bahan bakar unit termis ke-j per jam

P_{Tj} = beban unit termis ke-j

N = jumlah unit termal

j = indeks nomor unit pembangkit

Δt = selang waktu

Sistem tenaga listrik dengan mengabaikan rugi-rugi transmisi dapat dilihat pada gambar 2.5. Sistem ini memperlihatkan pembangkitan termal yang terdiri atas N buah unit yang dihubungkan pada sebuah bus bar untuk melayani total beban sebesar P_D. Masukan untuk setiap unit ke-i adalah F_i yang menyatakan tingkat biaya bahan bakar dari masing-masing unit, dan daya keluaran dari masing-masing unit P_i adalah daya listik yang dibangkitkan oleh tiap-tiap unit. Biaya total F_T yang ditanggung sistem adalah jumlah biaya dari tiap-tiap unit pembangkit. Dan batasan yang paling penting dari pengoperasiaan pembangkit termal tersebut adalah daya listrik yang dihasilkan harus sama dengan besarnya beban konsumen.

Secara matematis pernyataan diatas dapat dinyatakan dengan persamaan berikut:

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

$$= \sum_{i=1}^{N} F_i(P_i) \quad \text{.....2.10}$$

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

$$P_D = \sum_{i=1}^{N} P_i \quad \text{.....2.11}$$

$$P_D - \sum_{i=1}^{N} P_i = \phi = 0 \quad \text{.....2.12}$$

Dimana: P_D = kebutuhan beban

P_i = jumlah daya yang dihasilkan.

Penyelesaian masalah optimasi ini dapat diselesaikan dengan metode yang menggunakan fungsi pengali lagrange:

$$\pounds = F_T + \lambda\phi$$

$$\pounds = \sum_{i=1}^{N} F_i(P_i) + \lambda(P_D - \sum_{i=1}^{N} P_i) \quad \text{.....2.13}$$

Dimana : λ : pengali lagrange

Untuk mencari harga optimal dari fungsi lagrange terhadap P_i, dapat diperoleh dengan operasi gradien dari persamaan lagrange sama dengan nol.

$$\nabla\pounds = 0$$

$$\frac{\partial L}{\partial P_i} = \frac{\partial F_i}{\partial P_i} + \lambda\left(\frac{\partial P_D}{\partial P_i} - \frac{\partial P_i}{\partial P_i}\right) = 0$$

$$\frac{\partial F_i}{\partial P_i} + \lambda(0-1) = 0$$

$$\frac{\partial F_i}{\partial P_i} = \lambda \quad \text{.....2.14}$$

Persamaan diatas menunjukkan bahwa distribusi atau pembagian beban yang optimal terjadi apabila semua unit pembangkit beroperasi pada laju pertambahan

bahan bakar atau *Incremental Fuel Cost Characteristic* yang sama, ternyata nilainya sama dengan nilai λ. Kondisi optimal ini memerlukan persamaan-persamaan pembatas(constraints) agar keluaran(output) setiap unit pembangkit listrik harus lebih besar atau sama dengan keluaran minimum, lebih kecil atau sama dengan keluaran maksimum yang diijinkan.

Dari banyaknya N buah unit pembangkit listrik yang telah dibahas, maka dapat diambil kesimpulan sebagai berikut:

$\frac{\partial F_i}{\partial P_i} = \lambda$ ada N buah persamaan

$P_{i\,min} \leq P_i \leq P_{i\,mak}$ ada 2N buah pertidaksamaan

$P_D = \sum_{i=1}^{N} P_i$ ada 1 buah pembatas

Batasan-batasan kesimpulan diatas dapat diperluas menjadi:

$\frac{\partial F_i}{\partial P_i} = \lambda$ untuk $P_{i\,min} \leq P_i \leq P_{i\,mak}$

$\frac{\partial F_i}{\partial P_i} \leq \lambda$ untuk $P_i = P_{i\,mak}$

$\frac{\partial F_i}{\partial P_i} \geq \lambda$ untuk $P_i = P_{i\,min}$

2.5 Metode Tabu Search[1][8][10]

Tabu search adalah suatu metode optimasi yang didasarkan pada prinsip yang berasal dari proses biologi, yang menghindari jebakan optimum lokal dengan menggunakan struktur memori jangka pendek, solusi-solusi paling baik yang baru terbentuk dimasukkan dalam daftar tabu yang membolehkan kembali ke solusi sebelumnya (backtracking). Tabu search menggunakan sebuah prosedur

peningkatan iteratif yang dimulai dari sebuah solusi awal dan mencoba-coba beberapa solusi yang dilakukan secara acak untuk menentukan solusi yang paling baik.

Dalam algoritma *tabu search*, untuk memecahkan masalah optimasi kombinasi dari unit-unit pembangkit pada sistem tenaga listrik, ide dasarnya adalah untuk memilih solusi pada tingkat random atau acak dan kemudian memperoleh pendekatan terhadap solusi. Selama prosedur pencarian, solusi terbaik selalu disesuaikan dan disimpan sampai kriteria penghentian bisa terpenuhi.

2.5.1 Komponen Utama Tabu Search[1][8][10]

Tabu search menggunakan sebuah prosedur peningkatan iteratif yang dimulai dari sebuah solusi awal dan mencoba-coba beberapa solusi yang dilakukan secara acak untuk menentukan solusi yang paling baik. *Tabu search* merupakan suatu prosedur optimasi yang kuat yang telah sukses dijalankan pada sebuah masalah optimasi kombinatioral.

Komponen utama dari algoritma tabu search adalah daftar tabu (*tabu list)* dan level aspirasi (*aspiration level)* yang dapat dijabarkan sebagai berikut:

1. Daftar tabu (*tabu list)*

   Merupakan suatu daftar yang digunakan untuk menghindari kembalinya solusi ke optimum lokal yang baru saja dilakukan, perpindahan selanjutnya yang sudah ada seperti yang ada dalam daftar tabu yang telah ditetapkan untuk mencapai solusi yang optimal harus tidak diperbolehkan. Hal ini dapat dilakukan dengan melakukan penyimpanan terhadap perpindahan pada struktur data seperti panjang

struktur pertama yang disebut daftar tabu. Elemen-elemen daftar tabu ditentukan oleh besarnya solusi trial(coba-coba) yang telah dilakukan yang mana solusi baru yang terbentuk dimasukkan atau ditambahkan dibagian bawah dari daftar tabu dan solusi yang telah lama terbentuk dimasukkan atau ditambahkan ke bagian atas dari daftar tabu, sehingga dalam daftar tabu berisi tentang solusi-solusi terbaik yang dihasilkan dari solusi trial. Setelah solusi trial (coba-coba) dibangkitkan, maka solusi trial(coba-coba) dicek apakah ada dalam daftar tabu atau tidak ada. Jika tidak ada, maka ruang pencarian yang berhubungan akan dikurangi, begitu seterusnya. Pendekatan ini dimaksudkan untuk meningkatkan optimalitas lokal dengan strategi pelanggaran gerakan tertentu (atau secara luas dapat diartikan menghentikan). Tujuan dari klasifikasi tabu adalah untuk mencegah adanya siklus atau pengulangan pencarian solusi dengan cara atau jalan yang sama atau pernah digunakan.

Elemen-elemen *tabu list* ditentukan oleh sebuah fungsi yang menggunakan informasi historis dari proses pencarian, yaitu peningkatan iterasi sampai iterasi Z dimasa lalu, dimana Z adalah ukuran dari daftar tabu(*tabu list*), yang dapat ditetapkan atau bervariasi tergantung pada aplikasi pada tahap pencarian. Daftar tabu (*tabu list*) diperoleh dengan mencatat gerakan dalam urutan pembuatannya. Setiap waktu elemen baru ditambahkan pada bagian bawah daftar, elemen terlama dalam daftar akan dihilangkan di bagian atasnya.

Adapun filosofi dari pencarian dengan menggunakan daftar tabu (*tabu list*) bertujuan agar langkah pencarian solusi terbaik tidak terjebak dalam optimum lokal namun langkah pencarian dibimbing untuk selalu menuju ketitik optimum global dan mencegah terjadinya *backtracking* dimana suatu kondisi kembali kepada solusi yang pernah dilalui atau keadaan yang menjauhi titik optimum global. Hal ini dikarenakan pada sistem tabu search menggunakan memori jangka pendek yang mencatat atau merekam solusi-solusi terbaik yang pernah didapatkan untuk dimasukkan dalam daftar tabu (*tabu list*). Struktur memori melarang atau menghentikan gerakan tertentu yang akan kembali ke solusi yang sebelumnya atau pernah digunakan.

Dimensi daftar tabu (*tabu list*) disebut dengan ukuran daftar tabu (*tabu list*). Jika ukurannya terlalu besar maka solusi dengan kualitas tinggi tidak dapat tercapai, sedangkan bila ukurannya terlalu kecil maka perputaran kembali bisa terjadi dalam proses pencarian dan proses ini sering kembali ke solusi yang baru saja dilakukan atau dikunjungi, sehinggan pilihan dalam daftar tabu(*tabu list*) sangat penting. Secara empiris, ukuran daftar tabu(*tabu list*) harus tumbuh sesuai dengan ukuran masalah yang akan diselesaikan. Dalam skripsi ini, menggunakan daftar perpaduan yang dinamis yang berperan sebagai daftar tabu(*tabu list*), yang ukurannya adalah 10 untuk mencatat dan merekam solusi yang baru saja dilakukan. Hal ini berarti perpindahan akan disimpan dalam tabu untuk durasi waktu selama 10 perpindahan. Untuk perpindahan selanjutnya dicek apakah lebih baik dari yang ada

dalam daftar tabu(*tabu list*), dan jika lebih baik diambil dan dipilih dari daftar tabu(*tabu list*) yang terjelek dibuang dan apabila tidak proses kelanjutan.

2. Kriteria Aspirasi(*aspiration level)*

   Merupakan masukan dalam daftar tabu yang berupa nilai tertentu sebagai fungsi evaluasi, oleh karena itu kriteria aspirasi digunakan untuk memungkinkan perpindahan tabu jika perpindahan tersebut dinilai memberikan solusi yang lebih baik. Dengan kata lain, criteria aspirasi adalah untuk memungkinkan perpindahan tabu yang lebih baik dari perpindahan sebelumnya untuk diseleksi jika tingkat aspirasinya diperoleh dengan baik. Status tabu dari sebuah gerakan bukanlah absolut, tetapi dapat dilanggar jukan kondisi tetentu bisa dipenuhi, yang dinyatakan dengan level aspirasi(*aspiration level*). Jika kriteria aspirasi yang tepat terpenuhi maka gerakan dapat diterima meskipun berada dalam klasifikasi tabu. Intinya, kriteria aspirasi dimasudkan untuk meniadakan status tabu jika suatu gerakan "cukup baik" dengan pengertian mengasilkan solusi dengan funsi objektif yang lebih baik dari pada solusi sebelumnya dengan gerakan yang sama. Akibatnya, level aspirasi(*aspiration level*) yang dihubungkan dengan setiap gerakan dalam daftar tabu adalah sama dengan nilai fungsi obyektif yang diperoleh ketika menjalankan gerakan. Efektifitas dalam penggunaan kriteria aspirasi adalah menambah sedikit fleksibilitas dalam *tabu search* dengan mengarahkan pencarian kepada gerakan yang atraktif.

2.5.2 Kriteria penghentian[1]

Ada beberapa kondisi penghentian dari pencarian. Dalam hal ini, pencarian dihentikan jika dua kondisi berikut terpenuhi:

- Kendala batasan pembebanan telah tercapai.
- Kendala cadangan berputar bisa tercapai atau terpenuhi.

2.5.3 Algoritma Tabu Search Untuk Komitmen Unit[1][8][10]

Dalam menggunakan algoritma *tabu search*, untuk memecahkan masalah optimasi kombinasi unit-unit pembangkit pada sistem tenaga listrik, ide dasarnya adalah untuk memilih solusi pada tingkat random atau acak dan kemudian memperoleh pendekatan terhadap solusi. Selama prosedur pencarian, solusi terbaik selalu disesuaikan dan disimpan sampai kriteria penghentian bisa dipenuhi.

Pada masalah komitmen unit, dua tipe variabel perlu ditentukan. Adapun variabel-variabel tersebut adalah status unit U_{it} menjadi variabel integer (0 atau 1) dan variabel daya keluaran P_{it} menjadi variabel berkelanjutan. Masalahnya dapat diuraikan menjadi dua submasalah, yaitu masalah optimasi kombinatioral dalam U_{it} dan masalah optimasi non linier dalam P_{it}. *Tabu search* digunakan untuk memecahkan masalah optimasi kombinatioral dan optimasi non linier dipecahkan dengan program kuadrat.

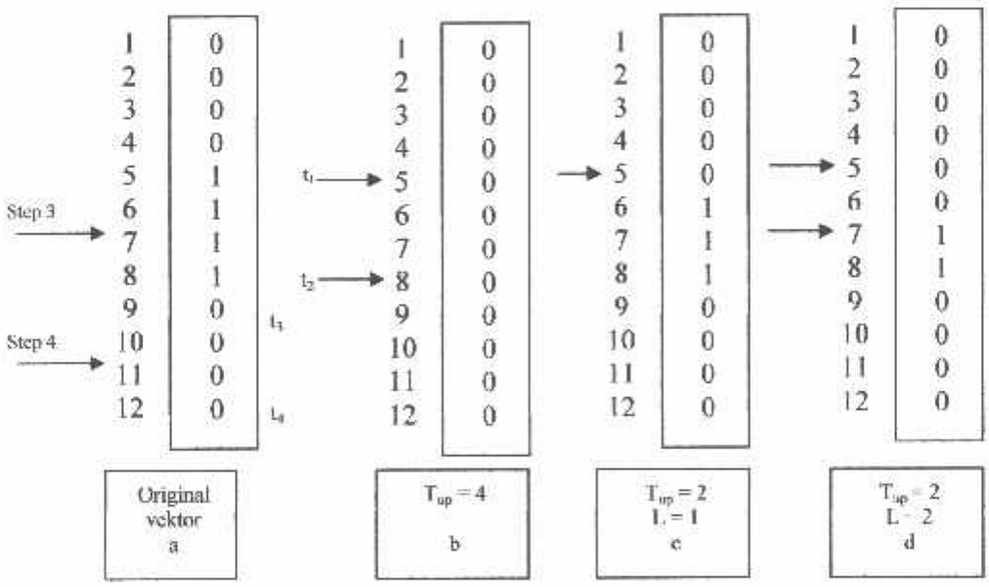
Algoritma tabu search yang telah ditentukan berisi tiga langkah:

1. menentukan solusi trial secara random
2. menghitung fungsi objektif dari solusi dengan memecahkan masalah economi dispatch

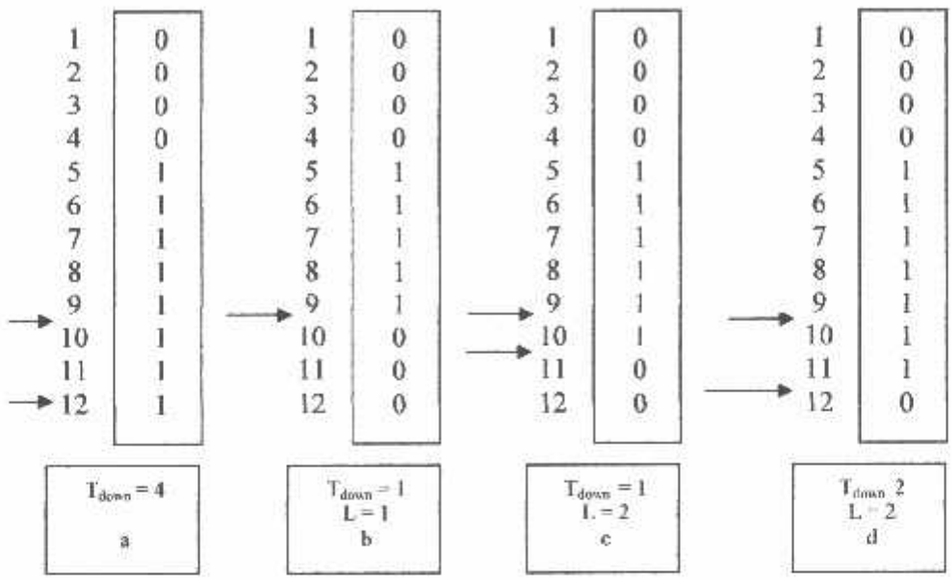
3. menggunakan prosedur tabu search untuk menerima atau menolak solusi yang telah ada.

2.5.4 Pembangkitan Solusi Trial[8][10]

Tetangga harus dicari secara random atau acak, layak, dan memenuhi ruang solusi sebanyak mungkin. Yang dimaksud tetangga disini adalah solusi coba-coba yang dilakukan disekitar atau didaerah dekat solusi utama untuk mendapatkan solusi terbaik, dimana solusi tersebut tetap mengacu pada solusi utama yang telah dihasilkan sebelumnya dari solusi awal. Karena kendala-kendala dalam masalah komitmen unit bukan masalah yang mungkin. Kendala-kendala yang paling sulit adalah memenuhi batasan minimum up time dan minimum down time.



GAMBAR 2.6 ATURAN-ATURAN YANG DIBERLAKUKAN UNTUK MEMPEROLEH SUATU SOLUSI TRIAL SEBAGAI TETANGGA DARI SUATU SOLUSI YANG LAYAK



GAMBAR 2.7 ATURAN-ATURAN YANG DIBERLAKUKAN UNTUK MEMPEROLEH SUATU SOLUSI TRIAL SEBAGAI TETANGGA ADARI SUATU SOLUSI YANG LAYAK

Kontribusi utama dalam proses tersebut adalah implementasi aturan-aturan baru untuk memperoleh solusi-solusi yang layak secara random lebih tepat. Aturan-aturan yang digunakan untuk memperoleh solusi trial dari tetangga dari solusi yang layak yang ada akan digambarkan dengan menggunakan sebuah contoh. Dalam contoh pada gambar diatas, diasumsikan nilai T = 12, $MUT_i = 2$ atau 4 dan $MDT_i = 1$ atau 4.

Langkah 1: Tentukan secara acak unit i, i ~ (1, N); dan satu jam t, t ~ (1, T). Gambar 2.6a menunjukkan status beberapa unit i dalam periode 12 jam. Unit ON antara periode 5 dan periode 8.

Langkah 2: Jika unit *i* pada jam t adalah ON (seperti pada 5,6,7, dan 8 pada gambar 2.6a), maka lanjutkan ke langkah 3 untuk mempelajari kondisi OFF pada jam *t* ini. Sebaliknya, jika unit *i* pada jam *t* adalah OFF (seperti pada 1, 2, 3, 4, 9, 10, 11, dan 12), maka lanjutkan ke langkah 4 untuk mempelajari kondisi ON pada waktu *t* ini.

Langkah 3: Perubahan unit *i* dari ON ke OFF

a. Bergerak dari jam *t* mundur dan maju dalam waktunya, untuk menentukan panjang periode ON. Dalam contoh ini, jika $t = 6$, maka $T_{on} = 8 - 5 + 1 = 4$, dan unit adalah ON selama jam 5, 6, 7, 8.
b. Jika T_{on} = MUT, maka unit ditetapkan dalam kondisi OFF di semua jam yang melibatkan T_{on}. Dalam contoh, jikan MUT = 4 maka matikan unit ke OFF pada *t* = 5, 6, 7, 8(Gambar 2.6b)
c. Jika T_{on} > MUT, maka tetapkan $L \sim (1, T_{on} - MUT)$
d. Matikan unit dalam kondisi OFF pada jam t_1, t_1+1, ..., t_1+L-1. dimana t_1 adalah jam pertama saat unit dalam kondisi ON. Dalam contoh, jika MUT_i adalah 2, maka $L \sim (1, 2)$. Ada dua solusi yang memungkinkan disini yaitu, jika L = 1, maka unit menjadi OFF pada *t* = 5 (Gambar 2.6c) sedangkan jika L = 2 maka unit menjadi OFF pada *t* 5, 6(Gambar 2.6d).

Langkah 4: Perubahan unit dari OFF ke ON

a. Bergerak dari jam *t* mundur dan maju dalam waktunya, untuk menentukan panjang periode OFF. Dalam contoh ini, jika $t = 10$, maka $T_{off} = 12 - 9 + 1 = 4$, dan unit adalah OFF selama jam 9,10, 11, dan 12.

b. Jika T_{off} = MDT, maka unit ditetapkan dalam kondisi ON di semua jam yang melibatkan T_{off}. Dalam contoh, jikan MDT = 4 maka matikan unit ke ON pada t = 9,10, 11, dan 12 (Gambar 2.7a)

c. Jika T_{off} > MDT, maka tetapkan L ~ (1, T_{off} – MDT)

d. Tetapkan unit dalam kondisi ON pada jam t_3, t_3+1, ..., t_3+L – 1. dimana t_3 adalah jam pertama saat unit dalam kondisi OFF. Dalam contoh, jika MDT, adalah 2, maka L ~ (1, 3). Ada tiga solusi yang memungkinkan disini yaitu, jika L = 1, maka unit menjadi ON pada t = 9 (Gambar 2.7b) sedangkan jika L = 2 maka unit menjadi ON pada t 9,10 (Gambar 2.7c); jika L = 3, unit berada dalam kondisi On pada t – 9, 10, 11 (Gambar 2.7d)

Langkah 5: Periksa cadangan berputar; periksa pemenuhan cadangan berputar untuk periode waktu unit berubah dalam langkah 3 dan langkah 4. Jika ini bisa memenuhi, maka solusi trialnya bisa digunakan, jika tidak memenuhi, maka kembali ke langkah 1.

2.5.5 Membangkitkan solusi awal[1][8][10]

Metode tabu search membutuhkan sebuah penjadwalan awal yang mungkin, yang mana memenuhi semua batasan-batasan dari sistem dan unit-unit pembangkit. Penjadwalan ini dibangkitkan secara acak. Algoritma untuk menentukan solusi awal (*initial solution*) dapat digambarkan sebagai berikut:

Langkah 1: Tetapkan $U_{it} = V_{it} = P_{it} = 0$. tentukan t = 1

Langkah 2: Lakukan Proses berikut:

a. Bangkitkan secara random sebuah unit i, i ~ (1, N)

b. Jika unit i pada jam t adalah OFF (U_{it} = 0) maka lanjutkan ke langkah 3. jika tidak, lanjutkan kc langkah 2b untuk memilih unit yang lainnya.

Langkah 3: Mengikuti prosedur dalam langkah 4, pada paragraf/subbab 2.5.4, untuk melihat kemungkinan menjalankan unit ON mulai jam t

Langkah 4: Jika t = T, lanjutkan ke langkah 5, dan jika tidak, tetapkan $t = t + 1$ dan kembali ke langkah 2.

Langkah 5: Periksa batasan cadangan berputar pada semua jam. Ulangi langkah 2 dan langkah 3 untuk jam yang mana batasan tidak bisa dipenuhi.

2.5.6. Tipe Daftar Tabu Untuk UCP[8][10]

Daftar tabu (*tabu list*) mencakup salah satu fungsi jangka pendek primer dari prosedur tabu search, yang mana hanya mencatat hanya *Z* gerakan terbaru, diman Z adalah ukuran dari daftar tabu (*tabu list*). Sebuah gerakan dalam masalah komitmen unit dikatakan sebagai sebuah perubahan unit ON ke OFF atau sebaliknya pada jam tertentudalam penjadwalan waktu horison. Karena matrik solusi dari masalah komitmen unit adalah U dan P mempunyai ukuran yang besar(T x N) diperlukan percobaan dan pengujian yang berbeda. Untuk menjelaskan atribut gerakan daftar tabu (*tabu list*) darpada mencatat matrik solusi secara keseluruhan. Dalam pelaksanaannya, daftar tabu (*tabu list*) dibuat terpisah untuk setiap unit pembangkit. daftar tabu (*tabu list*) unit pembangkit (GUTL) mempunyai dimensi Z X L, dimana L adalah panjang atribut gerakan tercatat. Ada empat pendekatan untuk tipe daftar tabu (*tabu list*) untuk masalah komitmen unit.

2.5.7. Pendekatan Yang Berbeda Dari Daftar Tabu Untuk UCP[8][10]

Dalam pelaksanaan algoritma tabu search untuk masalah komitmen unit, digunakan pendekatan daftar tabu (*tabu list*) berbeda untuk memperoleh yang lebih tepat (dalam ukuran fungsi biaya, kecepatan solusi, dan memori yang

dialokasikan). Dibawah ini akan dijelaskan setiap pendekatan dengan metode pelaksanaan dari pendekatan berbeda dari daftar tabu (*tabu list*).

Pendekatan 1

Pada pendekatan 1 ini, setiap GUTL adalah suatu rangkaian dimensi dengan panjang Z. Setiap data masukan berisi sebuah waktu yang telah ditetapkan sebelumnya untuk menghasilakan solusi coba-coba(trial) dari unit tanpa memperhatikan statusnya pada jam tersebut.

Pendekatan 2

GUTL dalam pendekatan ini adalah yang berdimensi Z x 2. setiap data masukan berisi waktu yang bertepatan sebelumnya untuk menghasilkan solusi coba-coba(trial) untuk unit ini bukan menjadi status pada jam ini.

Pendekatan 3

Dalam hal ini, setiap GUTL berisi satu rangkaian dimensi berisi beberapa entri Z. Setiap data masukan berisi jumlah periode ON dari unit. Dalam menggunakan pendekatan ini pada gambar 2.6 gerakan tercatat adalah 4(jumlah dalam vektor)

Pendekatan 4

Dalam pendekatan ini, dilakukan pencatatan pada kondisi yang mana unit ditetapkan sebagai ON dan OFF selam horison waktu penjadwalan. Kondisi ini bisa berada dalam pasangan. Jika pasangan ON – OFF ini terjadi dalam waktu m selama horison waktu jadwal, maka GUTL akan mempunyai ukuran Z x ($2m$). Dalam menggunakan pendekatan ini kepada solusi yang ditetapkan pada gambar 2.6 dan 2.7, daftar tabu(*tabu list*) dari unit ini dapat dilihat pada gambar 2.8. daftar

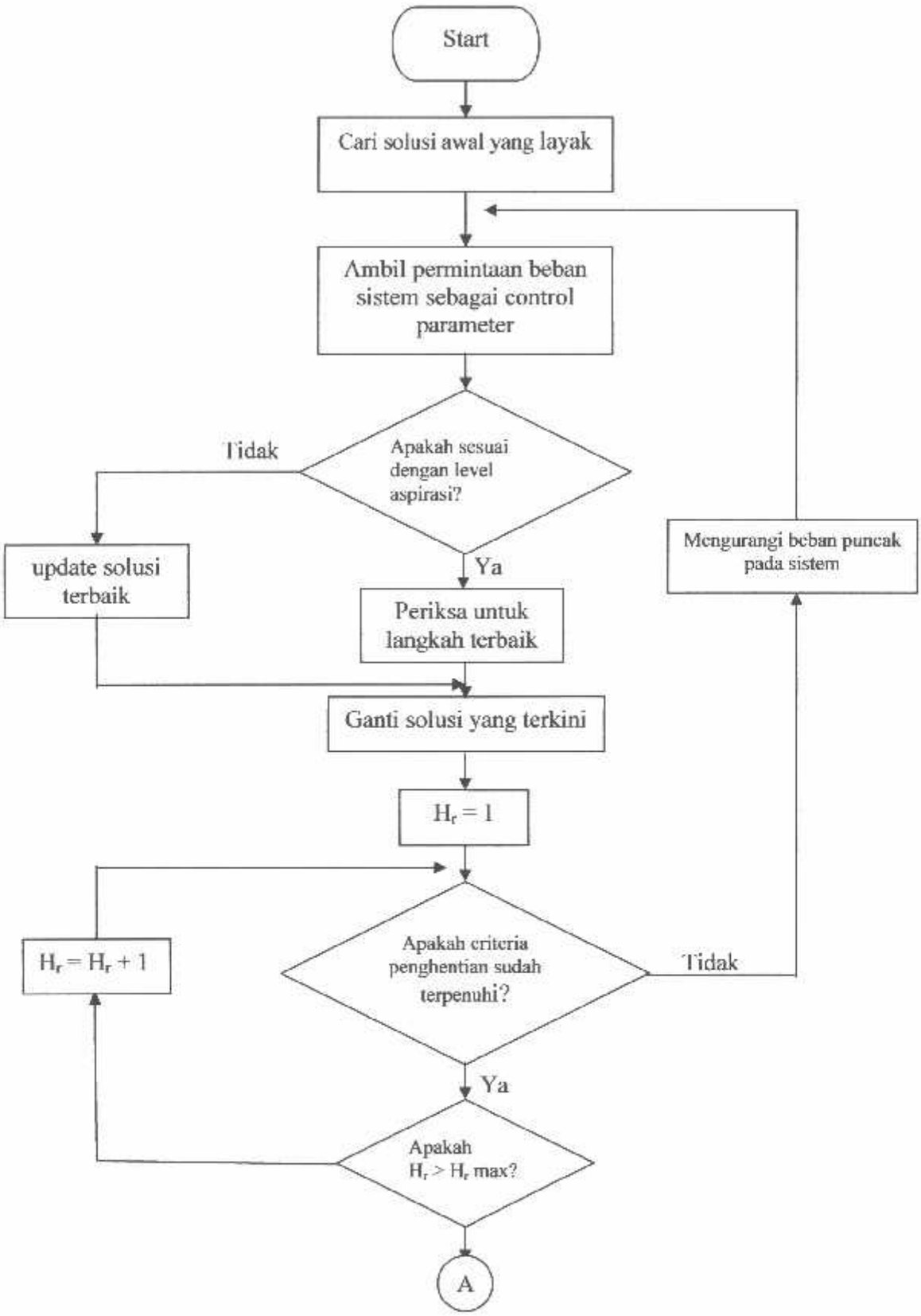
tabu(*tabu list*) dengan ukuran 8 (8 gerakan atau trial yang dicatat, dan $m = 1$) yang memperlihatkan start up dan shut down untuk setiap solusi trial.

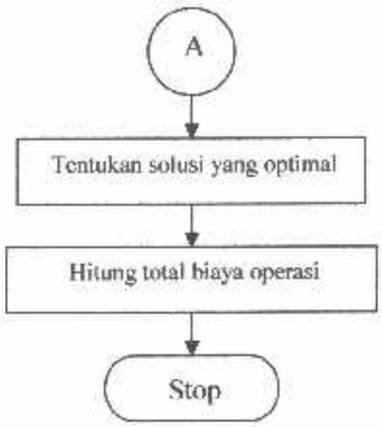
	1	2	3	4	5	6	7	8
Start	5	0	6	7	5	5	5	5
Shut	9	0	9	9	1	11	11	12
	.	.	.	.	.	.	.	.

GAMBAR 2.8 DAFTAR TABU(*TABU LIST*) UNTUK SATU UNIT YANG BERHUBUNGAN DENGAN GAMBAR 2.6 DAN GAMBAR 2.7 DENGAN Z = 8, M = 1

Langkah kerja dari metode tabu search dapat dilihat pada bagan berikut:

1. Inisialisasi parameter
2. mencari solusi awal yang layak
3. Mengambil masukan permintaan beban sistem sebagai kontrol parameter
4. Memeriksa apakah sesuai dengan level aspirasi
   - Jika 'Ya' update ke solusi terbaik
   - Jika 'tidak' memeriksa untuk langkah/pergerakan terbaik
5. Mengganti dan menempatkan kembali solusi terkini dan terbaik.
6. Apakah criteria penghentian menempatkan kembali solusi yang terbaru dan terbaik
7. set H_r = 1sudah terpenuhi. Jika "ya" lanjutkan ke langkah selanjutnya, bila "tidak" melakukan pengurangan permintaan atau masukan pada system dan kembali ke langkah 3
8. Apakah $H_r > H_r$ max. Jika "ya" lanjutkan ke langkah 10, jika "tidak" lanjutkan ke langkah 9
9. $H_r = H_r + 1$
10. Melakukan perhitungan untuk mendapatkan solusi yang optimal
11. Menghitung total biaya operasi
12. cetak hasil dan stop


Start
Cari solusi awal yang layak
Ambil permintaan beban sistem sebagai control parameter
Apakah sesuai dengan level aspirasi?
Tidak
Ya
update solusi terbaik
Periksa untuk langkah terbaik
Mengurangi beban puncak pada sistem
Ganti solusi yang terkini
$H_r = 1$
Apakah criteria penghentian sudah terpenuhi?
Tidak
Ya
$H_r = H_r + 1$
Apakah $H_r > H_r$ max?
A



GAMBAR 2.9 SKEMA *TABU SEARCH*

2.6 Teori *Evolutionary Programming*[1][7][11]

Metode ini diperkenalkan oleh Lawrence J. fogel pada tahun 1960. Metode ini merupakan strategi optimasi yang meniru prinsip evolusi alam sebagai metode untuk memecahkan masalah optimasi parameter. Pada metode ini individu yang lebih kuat mempunyai kemungkinan untuk menjadi pemenang dan mempunyai kesempatan hidup lebih besar.

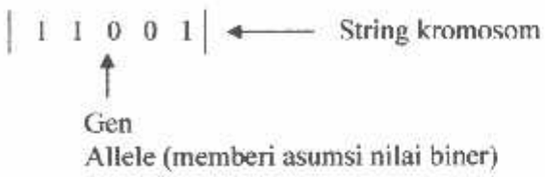
2.6.1 Prosedur *Evolutionary Programming*[1][7]

Evolutionary programming merupakan strategi optimasi yang meniru prinsip evolusi alam sebagai metode untuk memecahkan masalah optimasi parameter dengan cara penyusunan populasi dari kandidat penyelesaian dari jumlah generasi. Pada prosedur metode evolutionary programming terdapat beberapa urutan langkah langkah kerja atau operasi, yaitu:

1. Pengkodean

   Langkah pertama kali dilakukan dalam *evolutionary programming* adalah melakukan pengkodean atau representasi terhadap

permasalahan yang akan dilakukan. Secara umum evolutionary programming dibentuk oleh serangkaian kromosom yang ditandai dengan X_i dengan i = 1,2,...N Setiap elemen dalam kromosom ini adalah variable string yang disebut gen yang berisi nilai-nilai atau allele. Variable ini dapat dibentuk dengan bilangan interger.



GAMBAR 2.10 STRING KROMOSOM BIT BINER

Selanjutnya beberapa kromosom dibentuk dan berkumpul membentuk populasi. Populasi ini adalah populasi awal bagi evolutionary programming untuk awal melakukan pencarian.

2. *Fitness Function*

Dalam evolutionary programming sebuah fungsi fitness F(x) harus dirancang untuk masing-masing permasalahan yang akan diselesaikan. Dengan menggunakan kromosom tertentu fungsi objektif atau fungsi evaluasi akan mengevaluasi status masing-masing kromosom. Setiap gen X_i dipergunakan untuk menghitung $F_{k(x)}$ dengan k = 1,2, popsize.

3. Mutasi

Operator mutasi dipergunakan untuk memodifikasi satu atau lebih gen dalam satu individu. Cara kerjanya dengan membangkitkan sebuah nilai random r_k dimana k = 1,2,...panjang kromosom. Proses mutasi dalam evolutionary programming menggunakan operator *gaussian mutation*, dimana setiap individu terpilih secara acak untuk mengalami

mutasi berdasarkan nomor acak Gaussian untuk menciptakan individu baru(offspring)

4. Seleksi

   Masalah yang paling mendasar pada proses ini adalah bagaimana proses penyeleksian kromosom pada setiap populasi. Menurut teori Darwin, proses seleksi individu adalah: "individu terbaik akan tetap hidup dan akan menghasilkan keturunan".

5. Kompetisi

   Pada tahap kompetisi mekanisme seleksi dipakai untuk menghasilkan populasi yang layak. Setiap individu dalam populasi baik orang tua (parent) maupun anak (offspring) akan dikompetisi satu dengan yang lainnya. Kompetisi individu dengan lawannya didasarkan pada nilai fitness dari setiap individu tersebut. Agar proses kompetisi mencapai solusi yang optimal, maka terlebih dahulu ditetapkan peluang seleksi yang lebih besar dengan cara dirangking. Individu yang memenangkan kompetisi akan digunakan sebagai individu baru bagi pembangkitan selanjutnya

2.6.2 Skema *Evolutionary Programming*[7][11]

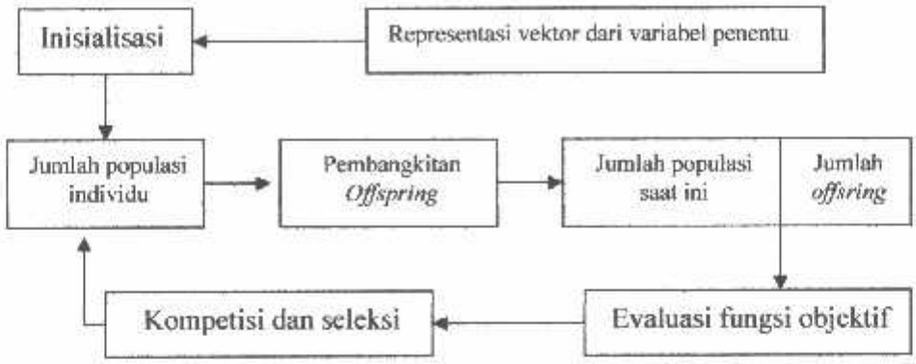
Evolutionary Programming mencari solusi optimal dari proses optimasi dengan melakukan penyusunan kromosom – kromosom untuk membentuk sebuah individu kemudian berkumpul membentuk populasi dari kandidat penyelesaian dari jumlah generasi atau proses iterasi. Populasi yang baru dibentuk dari seluruh populasi yang ada menggunakan operator mutasi. Operator ini memberikan

gangguan pada masing-masing komponen dari tiap solusi dalam populasi secara random atau acak untuk memperoleh solusi baru.

Tingkat optimal dari masing-masing kandidat solusi yang baru terbentuk dihitung dengan menggunakan fungsi fitness, dimana dapat didefinisikan sebagai suatu fungsi biaya atau fungsi objektif dari persoalan yang akan diselesaikan.

Keseluruhan proses untuk mencari solusi yang optimal menggunakan skema kompetisi, masing-masing individu baik orang tua (parent) maupun anak (offspring) dalam populasi saling berkompetisi yang didasarkan pada besarnya nilai fitnes pada masing-masing individu. Individu yang memenangkan kompetisi akan dijadikan hasil dari populasi yang akan digunakan untuk generasi selanjutnya. Pada tahap kompetisi berlaku, solusi yang lebih kuat dan lebih sehat menggantikan solusi yang lebih lemah sehingga solusi dari seluruh populasi ini disusun penyelesaian optimal secara global.

Teknik *Evolutionary Programming* seperti proses iterasi, dimana proses berhenti setelah mencapai kriteria penghentian yang telah ditetapkan atau ditentukan sebelumnya. Kriteria ini bekerja jika penyelesaian yang spesifik sudah tercapai dalam proses iterasi atau berhenti setelah solusi terbaik tidak berubah dalam beberapa generasi yang telah ditentukan. Skema dari *Evolutionary Programming* dapat dilihat pada gambar dibawah ini:



GAMBAR 2.11 SKEMA *EVOLUTIONARY PROGRAMMING*[11]

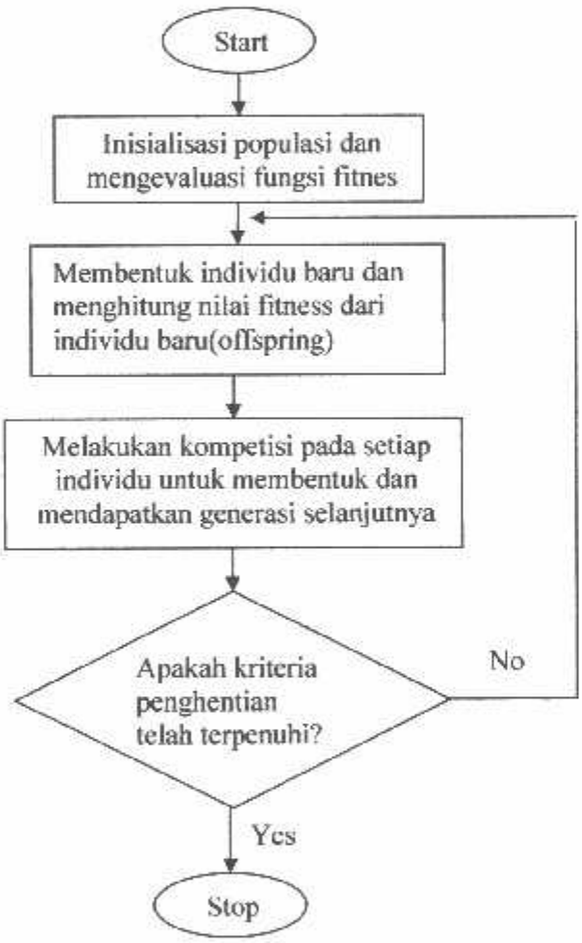
Skema diatas dapat diterangkan sebagai berikut:

1. Representasi vektor, nilai rel dari variabel yang diselesaikan dalam *Evolutionary Programming* direpresentasikan sebagai vektor p n-dimensi yang berhubungan dengan fungsi objektif. Masing-masing vektor p disusun sebagai individu dalam populasi.
2. Inisialisasi populasi dari individu asal dalam artian orang tua p_i , i = 1,2,...,N_p untuk diseleksi secara acak dari range yang layak pada masing-masing dimensi yang terdistribusi uniform.
3. Membangkitkan anak(offspring), jumlah dari offspring p_i , i= 1,2,...,N_p dibangkitkan dengan menggunakan operator gaussian mutation secara random variabel dengan mean nol atau standar deviasi untuk masing-masing komponen p_i, dimana setiap individu baik parent maupun offspring berada dalam kelompok seleksi.
4. kompetisi dan seleksi, masing-masing individu dalam kelompok seleksi harus berdasarkan $f(p_i)$ dan $f(p_i')$. N_p individu dengan nilai

fungsi terbaik yang mempunyai nilai minimum untuk proses minimasi diseleksi untuk membentuk *survivor set* untuk pengambilan keputusan dan digunakan sebagai individu asal untuk generasi selanjutnya.

5. Stopping rules, proses pembangkitan dan seleksi dengan nilai fungsi terbaik terus berlangsung sampai nilai fungsi itu tidak berubah sampai beberapa generasi atau nilai yang ditentukan sampai tercapai.

Dari skema *Evolutionari programming* diatas dapat dibuat langkah kerja dari metode *Evolutionari programming* sebagai berikut:



GAMBAR 2.12 FLOWCHART *EVOLUTIONARI PROGRAMMING*

Langkah-langkah utama algoritma penyelesaian metode evolutionary programming dapat diuraikan sebagai berikut:

1. Inisialisasi besarnya populasi dengan membentuk $s_i = S_i \sim U(a_k,b_k)^k$ dengan i = 1,2,...,m, dimana S_i adalah sebuah vektor acak, s_i adalah hasil dari vektor acak, $U(a_k,b_k)^k$ adalah sebuah distribusi seragam yang berkisar antara (a_k,b_k) dalam bentuk dimensi k, dan m adalah jumlah orang tua(parent)
2. Untuk s_i, i = 1,2,...,m, adalah membentuk besarnya nilai fitnes dengan $\theta(s_i) = G(F(s_i), v_i)$, dimana v_i direpresentasikan sebagai perubahan penggantian secara acak dalam lingkup dari s_i, variasi acak yang dikenakan dalam evaluasi dari $F(s_i)$, atau memenuhi hubungan s_i lainnya. F adalah peta atau daerah fungsi fitnes sesungguhnya dari s_i. $G(F(s_i), v_i)$ dideskripsikan sebagai besarnya fitnes yang telah dijalankan.
3. untuk s_i, i = 1,...,m, diubah dan dipasang pada s_{i+m} yaitu

   $s_{i+m} = s_{ij} + N(0, \beta j\ \theta(s_i) + z_j)$, j = 1,...,k

   $N(0, \beta j\ \theta(s_i) + z_j)$ adalah representasi sebuah variabel acak Gaussian dengan merata μ dan varian σ^2, βj adalah konstanta kesebandingan dari besarnya $\theta(s_i)$ dengan besarnya $\beta x p_i = 0{,}2$ dan z_j adalah simpangan untuk menjamin jumlah minimum dari varian.
4. Untuk s_{i-m}, i = 1,...,m. Dibutuhkan untuk besarnya nilai fitnes, yaitu:

   $$\theta(s_{i+m}) = G(F(s_{i+m}), v_{i+m})$$

5. Untuk s_i, i = 1,...,2m, adalah sebuah nilai w_i yang dihasilkan dengan menggunakan persamaan sebagai berikut:

$$W_i = \sum_{i=1}^{c} w_i *$$

$$W_i* = \begin{cases} 1 \text{ jika } \theta(s_i) \le \theta(s_{i+m}) \\ 0 \text{ cara lainnya} \end{cases}$$

Dimana: c adalah jumlah kompetisi

6. Solusi dari s_i dengan i = 1, ...,2m adalah turunan yang bersesuaian dengan nilai w_i dengan acuan nilai aktual dari $\theta(s_i)$, Jika nilainya lebih besar dari solusi *m* mencapai nilai dari *c*. Nilai pertama solusi *m* adalah panjang salinan atau tiruan yang besesuaian dengan nilai $0(s_i)$ untuk dasar dari generasi selanjutnya.
7. Proses kembali ke tahap 3 jika eksekusi waktu yang tersedia telah terlampaui.

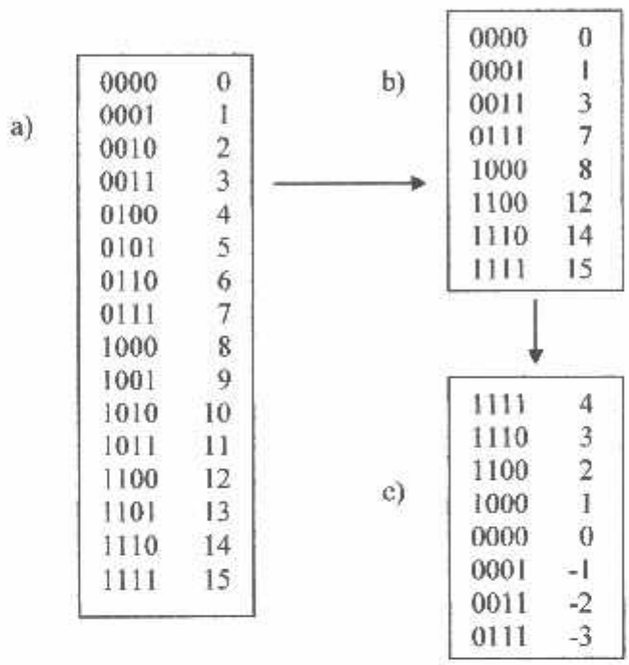
2.6.3. Evolutionary Programming untuk Masalah Komitmen Unit[1][7][11]

Pada dasarnya algoritma *Evolutionari programming* menggunakan pengkodean yang dilukiskan di bawah ini. Untuk mudahnya kita ambil contoh untuk N- unit pembangkit, dengan 24 jam waktu penjadwalan horison, dimana MUT sama dengan MDT, yaitu 4 jam untuk semua unit pembangkit. Untuk penjadwalan operasi suatu unit i seperti berikut, dimana "1" menunjukkan bahwa unit i beroperasi atau ON dan "0" menunjukkan bahwa unit i tidak beroperasi atau OFF.

Unit i [0000] [0001] [1111] [1111] [1110] [0000]

Unit i $[a_{i1}]$ $[a_{i2}]$ $[a_{i3}]$ $[a_{i4}]$ $[a_{i5}]$ $[a_{i6}]$

Peralihan antara penjadwalan 0 – 1 ke penjadwalan yang dikodekan ditunjukkan sebagai variabel state $[a_{ij}]$ (j = 1,2,3,...,6, dimana j adalah indeks untuk periode diperpendek). Elemen-elemen ini menggunakan nilai integer seperti yang ditunjukkan pada gambar berikut:



GAMBAR 2.13 PENGKODEAN OPERASI UNIT I[7]

Pengkodean dimulai dari sistem bit biner yang sederhana (gambar 2.13a). kemudian dilanjutkan dengan menghapus atau mengeliminasi kombinasi yang melanggar kendala *minimum down time* dan *minimum up time* untuk p = 4 jam waktu pembatasan (gambar 2.13b). Contohnya sebagai berikut, 0010 melanggar MUT = 4 dan 1001 melanggar MDT = 4. kombinasi selebihnya hanya disusun kembali menurut urutan angka jam-jam unit-unit i beroperasi atau ON yang dicakup dan tipe peralihan yang terlibat. Jadi walaupun "-2" dan "2" sebenarnya

menunjukkan bahwa unit beroperasi selama 2 jam, pada kasus pertama unit i dipasang dalam kondisi ON (0011) sedangkan pada kasus kedua tindakan yang diperintahkan justru sebaliknya (1100) jadi dalam kondisi OFF(Gambar 2.13c)

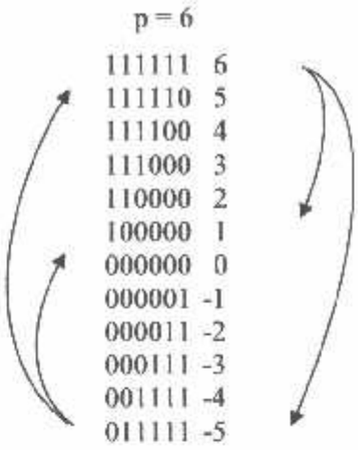
Untuk sebuah unit pembangkit i atau sebuah grup dari unit pembangkit, dipilih sedemikian rupa sehingga $p \leq$ *min (minimum up time, minimum down time)*; dengan $p \in \{2,3,4,6,8,12,24\}$ (untuk 24 jam)

Misalnya, jika besarnya *minimum up time* = 14 jam dan besarnya *minimum down time* = 7 jam, dan p harus dipilih pada $p \leq 7$. pada masalah penjadwalan suatu unit pembangkit i selama 24 jam dimana p harus merupakan pembagi dari 24, maka p bisa dipilih dari subset{2,3,4,6}. Misalkan dipilih p = 6, maka operasi penjadwalan dapat ditulis ulang sebagai berikut:

Unit i [000000] [011111] [111111] [100000]

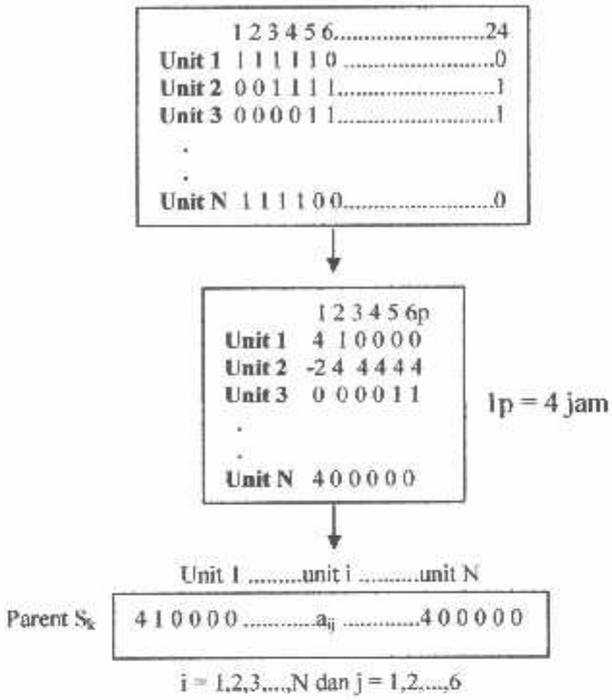
Unit i [0] [-5] [6] [1]

Dimana kode [0], [-5], [6], dan [1] adalah pengkodean yang menunjukkan bahwa unit beroperasi ON atau tidak beroperasiOFF. Selama pencarian, ketika $[a_{ij}]$ lebih besar dari [6] atau kurang dari [5], maka dapat diset ulang pada [0],[-5], atau [6] jika tes kemungkinan terpenuhi maka bagan dapat dilihat pada gambar 2.14



GAMBAR 2.14 PENCARIAN MEMUTAR

Ketika kode-kode semacam itu telah menandai semua unit i, atau grup dalam unit i, penjadwalan disusun sebagai salah satu deretan bilangan bulat dan dipandang, dilakukan dan dipandang sebagai kandidat reproduksi parent S_k (orang tua S_k pada gambar 2.15). Sebuah populasi kandidat yang demikian menghasilkan bentuk dasar dari generasi-generasi berikutnya pada proses *Evolutionary Programming.*



Gambar 2.15 Representasi String Dari Kandidat[7]

Dari gambar diatas dapat diimplementasikan langkah kerja dari *Evolutionary Programming* untuk menyelesaikan masalah komitmen unit mengikuti opersi-opersi sebagai berikut:

1. Masukkan populasi awal $p = (p_1, p_2,...,p_n)$ dengan $i = 1,2,...,N_p$ seperti pada elemen dalam vektor yang telah ditentukan oleh $p_j \sim$ acak ($p_{j\,min}$, $p_{j\,max}$) dengan $j = 1,2,...,N$.
2. Menghitung semua fungsi okjektif dan mencari nilai minimum dari fungsi objektif dengan fungsi fitnes
3. Menciptakan atau membentuk anak (*offspring*) menggunakan beberapa tahap, yaitu:

a. menghitung standar deviasi

$$\sigma_j = \beta\left(\frac{F_{Tij}}{\min(F_{Ti})}\right)(P_{j\,max} - P_{j\,min}),\ \text{dengan besar}\ \beta = 0{,}2$$

b. menambah variabel acak *gaussian* $N(0, \sigma_j^2)$ untuk semua tahapan dari variabel p_i untuk mendapatkan p_i'

4. Menghitung nilai N_p tiap individu dari $2N_p$ total tiap individu dari kedua p_i dan p_i' menggunakan tahap tahap untuk iterasi berikutnya, yaitu:
   a. mengevaluasi $r = (2N_p\ \text{acak}(0,1) + 1)$
   b. Mengevaluasi vektor acak dengan W_{pi} sum(W_x), dimana $x = 1,2, \ldots, N_p$, $i = 1,2,\ldots,N_p$ sehingga $W_x = 1$ jika $F_{Tij} / (F_{Tij} + F_{Tir}) <$ acak (0,1) atau lainnya $W_x = 0$
5. W_{pi} diturunkan dengan setiap N_p pertama atau awal akan dirangking dan panjang salinan atau tiruan yang besesuaian dengan nilai $\theta(s_i)$ untuk dasar dari generasi selanjutnya.
6. Prosedur di atas akan diulang dari tahap 2 sampai jumlah seluruh unit-unit pembangkit N_m telah dicari seluruhnya.
7. Proses seleksi dilakukan dengan menggunakan *Evolutionary Strategy*.

2.7 Metode *Evolutionary Strategy*[7][9]

Evolutionary strategy merupakan metode yang biasa digunakan untuk memecahkan suatau pencarian nilai dalam sebuah masalah optimasi. Metode ini didasarkan pada proses evolusi yang ada dalam makhluk hidup yaitu perkembangan generasi dalam sebuah populasi yang alami, secara lambat laun

mengikuti mekanisme alam, dimana individu yang lebih kuat memiliki kemungkinan untuk menjadi pemenang dan mempunyai kesempatan hidup yang lebih besar di dalam lingkungan yang kompetitif. *Evolutionary strategy* dapat digunakan untuk mencari solusi permasalahan-permasalahan dalam dunia nyata.

Evolutionary strategy ditemukan oleh Rechenberg pada tahun 1973 dengan seleksi (*selection*), mutasi (*mutation*), populasi (*population*) kemudian dikembangkan oleh Schwefel tahun 1981 yang memperkenalkan *recombination* dan populasi (*population*) individu lebih dari satu. Mereka kemudian menyajikan suatu perbandingan *Evolutionary Strategies* (ES) dengan teknik optimasi yang lebih tradisional.

Sebelum *Evolutionary strategy* dijalankan, maka sebuah kode yang sesuai (representasi) untuk persoalan harus dirancang. Titik solusi dalam ruang permasalahan dikodekan dalam bentuk kromosom/string yang terdiri dari komponen genetik terkecil yaitu gen. Pemakaian bilangan real (*floating point*) sebagai *allele* (nilai gen) memungkinkan penerapan operator *Evolutionary strategy* yaitu proses seleksi *(selection)*, *recombination*, mutasi *(mutation)*, dan kompetisi (*competition*) untuk menciptakan himpunan titik-titik solusi. Untuk memeriksa hasil optimasi, kita membutuhkan fungsi *fitness* yang menandakan gambaran hasil *(solution)* yang sudah dikodekan. Selama proses, induk harus digunakan untuk reperoduksi, mutasi dan kompetisi untuk menciptakan keturunan *(offspring)*.

2.7.1 Parameter *Evolutionary Strategy*[7][9]

Terdapat beberapa parameter yang digunakan dalam *Evolutionary Strategy*. Parameter tersebut digunakan untuk melihat kompleksitas dari *Evolutionary Strategy*. Parameter yang digunakan tersebut adalah :

a. Jumlah Generasi (MAXGEN)

Merupakan jumlah perulangan (iterasi) dilakukannya rekombinasi dan seleksi. Jumlah generasi ini mempengaruhi kestabilan output dan lama iterasi (waktu proses ES). Jumlah generasi yang besar dapat mengarahkan kearah solusi yang optimal, namun akan membutuhkan waktu yang lama. Sedangkan jika jumlah generasinya terlalu sedikit maka solusi akan terjebak pada *local optimum solution.*

b. Ukuran Populasi (POPSIZE)

Ukuran populasi mempengaruhi kinerja dan efektifitas dari *Evolutionary Strategy*. Jika ukuran populasi kecil maka populasi tidak menyediakan cukup materi untuk mencakup ruang permasalah, sehingga pada umumnya kinerja ES menjadi buruk. Dalam hal ini dibutuhkan ruang yang lebih besar untuk mempersentasikan keseluruhan ruang permasalahan. Selain itu penggunaan populasi yang besar dapat mencegah terjadinya konvergensi pada wilayah lokal.

c. Probabilitas Mutasi *(Pm)*

Mutasi digunakan untuk meningkatkan variasi populasi digunakan untuk menentukan tingkat mutasi yang terjadi, karena frekuensi terjadinya mutasi tersebut menjadi *Pm x POPSIZE x N*, dimana N adalah panjang struktur/gen dalam satu individu. Probabilitas mutasi yang rendah akan menyebabkan gen-gen

yang berpotensi tidak dicoba. Dan sebaliknya, tingkat mutasi yang tinggi akan menyebabkan keturunan akan semakin mirip dengan induknya. Dalam *Evolutionary Strategy* mutasi menjalankan aturan penting yaitu :

1. Mengganti gen-gen yang hilang sama proses seleksi.
2. Menyediakan gen-gen yang tidak muncul pada saat inisialisasi awal populasi.

d. Panjang Kromosom *(NVAR)*

Panjang kromosom berbeda-beda sesuai dengan model permasalahan. Titik solusi dalam ruang permasalahan dikodekan dalam bentuk kromosom/string yang terdiri dari komponen genetik terkecil yaitu gen. Pengkodean memakai *string* bilangan *real*.

2.7.2 Mekanisme *Evolutionary Strategy*[7][9]

a. Pengkodean atau Representasi

Langkah pertama kali yang dilakukan dalam penggunaan *Evolutionary Strategy* adalah melakukan pengkodean atau representasi terhadap permasalahan yang akan dilakukan. Secara umum *Evolutionary Strategy* dibentuk oleh serangkaian kromosom yang ditandai dengan xi $(i = 1,2...N)$. Setiap elemen dalam kromosom ini adalah variabel string yang disebut gen, berisi nilai-nilai atau allele. Variabel-variabel ini dapat dinyatakan dalam bentuk bilangan real *(floating point)*. Selanjutnya beberapa kromosom dibentuk dan berkumpul membentuk populasi. Populasi inilah populasi awal bagi *Evolutionary Strategy* untuk awal melakukan pencarian.

b. Fungsi *Fitness* (Fungsi Evaluasi)

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

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

c. Seleksi

Masalah yang paling mendasar pada proses ini adalah bagaimana proses penyeleksiannya. Menurut teori Darwin proses seleksi individu adalah : *"individu terbaik akan tetap hidup dan menghasilkan keturunan"*. Pada proses seleksi ini dapat menggunakan banyak metode seperti *rouletee wheel selection, rank selection, elitesm* dan lain sebagainya. Adapun metode-metode seleksi dapat dijabarkan sebagai berikut:

- *Roulette Wheel Selection*

  Dimana setiap individual memiliki harga fitness sehingga didapatkan probabilitas individual $\left(f(t)/\sum f(t)\right)$ tersebut dikopikan pada populasi yang baru. Untuk individual yang memiliki probabilitas 20% untuk jumlah populasi 10 maka kemungkinan individual tersebut dapat terpilih sebanyak dua kali.

Adapun algoritma dari *roulette-wheel* adalah sebagai berikut :

1. Menjumlahkan fitness dari seluruh anggota populasi.
2. Membangkitkan nilai k, suatu nilai random antara 0 dan total fitnessnya.
3. Menjumlahkan fitness dari kromosom-kromosom dari populasi mulai 0 hingga total fitness lebih besar atau sama dengan nilai k lalu ambil kromosom tersebut.

- *Rank Selection*

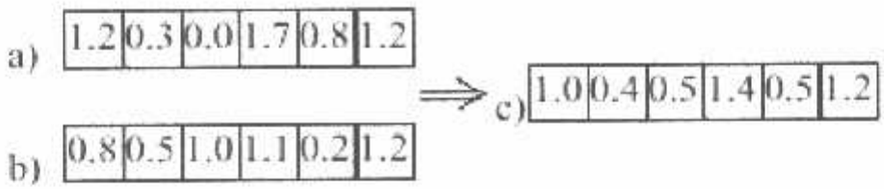
Apabila fitness yang dimiliki oleh suatu kromosom dalam populasi berbeda terlalu jauh dari kromosom lainnya maka hal ini dapat menjadi permasalahan. Misalnya bila kromosom terbaik mempunyai *fitness* yang menyebabkan besarnya tempat yang dimilikinya dalam *roulette wheel* sebesar 90% maka kromosom-kromosom yang lain akan mempunyai peluang yang terlalu kecil untuk diseleksi.

Rank selection pertama kali merangking populasi dan kemudian setiap kromosom diberi nilai fitness baru berdasarkan hasil rangking tersebut. Yang pertama akan mempunyai *fitness* 1, yang kedua akan mempunyai fitness 2 dan seterusnya sampai yang terakhir akan mempunyai *fitness* N. Dengan demikian semua kromosom akan mempunyai peluang untuk diseleksi..

d. *Recombination*

Evolutionary strategies setelah melakukan *initialization* dan *evaluation*, individual dipilih secara acak menjadi *parents*. Setelah melakukan seleksi *parent*

kemudian melakukan *Intermediate recombination.* dimana vector dari dua orang tua *(parent)* dirata-ratakan bersama-sama, unsur demi unsur, untuk membentuk suatu keturunan yang baru.



Gambar 2.15

INTERMEDIATE RECOMBINATION DARI INDUK (*PARENT*) A) & B) MENJADI ANAK (*OFFSPRING*) C)

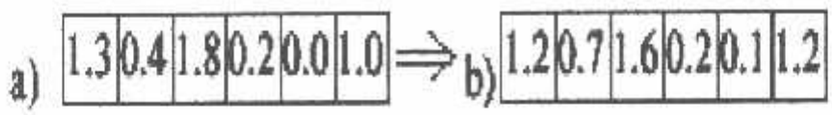
Seleksi orang tua (*parent*) menjadi turunan (*offspring*) menjadi lebih mudah, sebagai contoh dalam kaitannya dengan sifat alami penyajian, adalah sangat mudah merata-ratakan dari banyak individu menjadi turunan tunggal. Didalam *Evolutionary Strategy,* N *parent* yang terpilih secara acak (tidak didasarkan pada fitness terbaik

), lebih dari N *offspring* dihasilkan melelui penggunaan *intermediate recombination* dan kemudian N *parent* yang selamat terpilih secara deterministik

e. *Mutation* (Mutasi)

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

Evolutionary Programming yaitu menggunakan operator *Gaussian mutation*, dimana setiap individu akan terpilih secara acak untuk mengalami mutasi berdasarkan nomor acak Gaussian untuk menciptakan individu baru (*offspring*)



GAMBAR 2.16
MUTASI GAUSSIAN DARI INDUK (*PARENT*) A) MENGHASILKAN ANAK (*OFFSPRING*) B)

Fungsi dari operator mutasi adalah untuk menghindari agar solusi masalahan yang diperoleh bukan merupakan solusi optimum lokal. Tipe dan implementasi dari operator mutasi bergantung pada jenis pengkodean dan permasalahan yang dihadapi. Seberapa sering mutasi dilakukan dinyatakan dengan suatu probabilitas mutasi, Pm. Posisi elemen pada kromosom yang akan mutasi ditentukan secara random. Mutasi dikerjakan dengan cara melakukan perubahan pada elemen tersebut.

f. *Competition* (Kompetisi)

Dalam tahap kompetisi, mekanisme seleksi dipakai untuk menghasilkan populasi baru dari populasi yang ada. Melalui penggunaan skema kompetisi setiap individu dalam populasi rang tua (*parent*) maupun anak (*offspring*) akan dikompetisi/bersaing satu dengan yang lainnya. Kompetisi setiap individu dengan lawannya didasarkan pada nilai *fitness* dari setiap individu tersebut. Agar optimal, solusi yang lebih pas atau lebih optimal seharusnya memiliki peluang seleksi yang

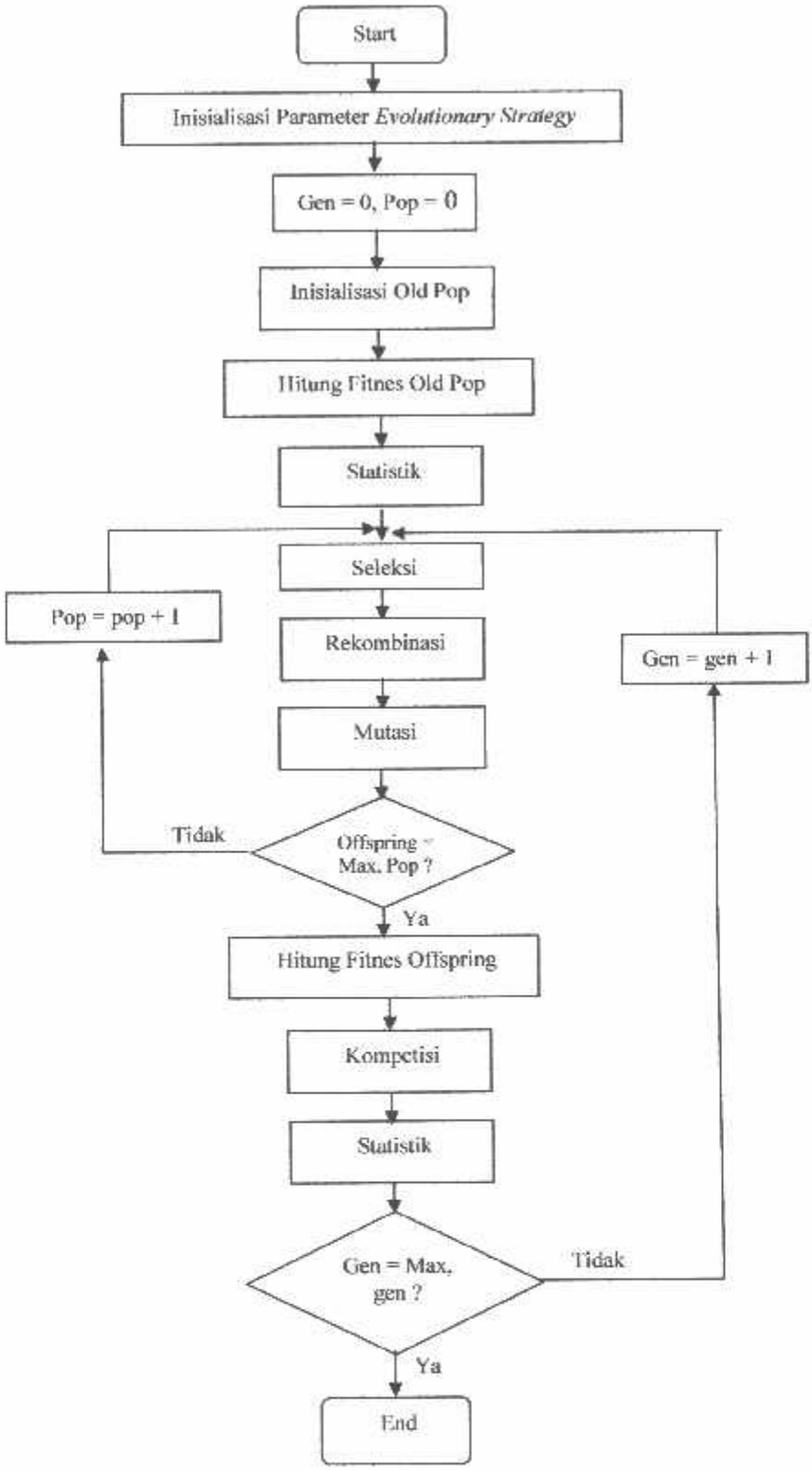
lebih besar. Individu yang memenangkan dari kompetisi akan digunakan sebagai individu yang baru bagi pembangkitan selanjutnya.

2.7.3 Algoritma *Evolutionary Strategy*

Langkah-langkah utama algoritma penyelesaian metode *Evolutionary Strategy* dapat diuraikan sebagai berikut:

1. Menentukan parameter inputan *evolutionary strategy* yang meliputi jumlah populasi, maksimum generasi, nilai kemungkinan mutasi, dan panjang kromosom tiap-tiap individu.
2. Generasi = 0, Populasi = 0.
3. Menghitung *fitness* dari kromosom tiap-tiap individu.
4. Melakukan proses statistik.
5. Melakukan rekombinasi terhadap *perent*
6. Melakukan proses mutasi mutasi *evolutionary strategy*.
7. Apakah *offspring* yang diinginkan sudah terpenuhi (max pop).
8. Jika " Tidak " maka populasi = pop + 1, kembali ke langkah 7.
9. Jika " Ya " maka perhitungan dilanjutkan ke langkah 11.
10. Menghitung *fitness* dari *offspring*.
11. Melakukan proses kompetisi.
12. Melakukan proses statistik
13. Apakah generasi yang diinginkan sudah terpenuhi (max Gen).
14. Jika " Tidak " maka generasi = gen + 1, kembali ke langkah 7.
    Jika " Ya " maka perhitungan berhenti.

Dari algoritma atau langkah kerja dari metode *Evolutionary Strategy* diatas dapat dibuat suatu flowchart sebagai berikut:

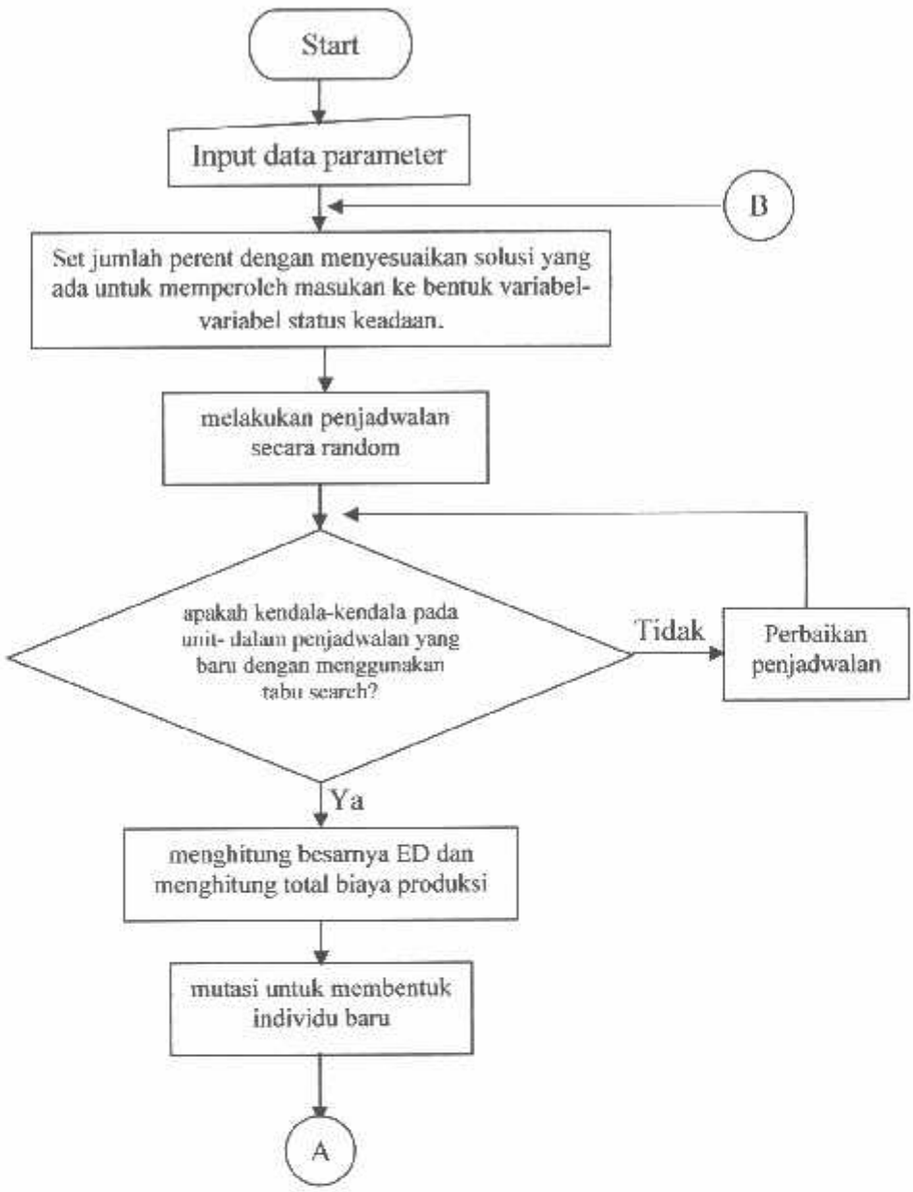


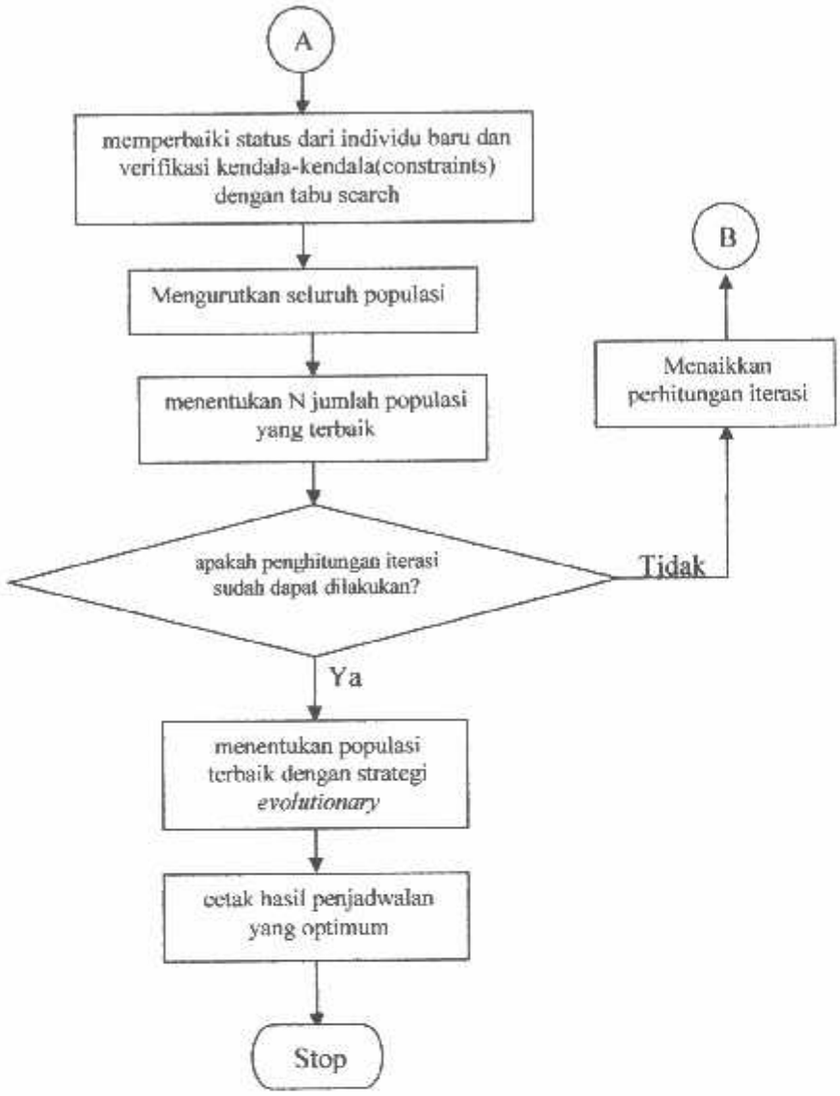
GAMBAR 2.17 FLOWCHART *EVOLUTIONARY STRATEGY*

2.8 Metode Kombinasi Evolutionary Programming dan Tabu Search[1]

Dalam teknik algoritma tabu search untuk menyelesaikan masalah komitmen unit, pengenalan (inisialisasi) status penjadwalan operasi suatu unit pembangkit dalam sistem dimasukkan dalam bentuk daya nyata yang dibangkitkan suatu unit-unit pembangkit yang beroperasi yang diberikan sebagai input. Sebagaimana diketahui bahwa tabu search digunakan untuk memperbaiki beberapa status yang diberikan yang digunakan untuk menghindari jebakan optimum lokal dengan menggunakan struktur memori yang membolehkan kembali ke solusi sebelumnya (backtracking). Individu baru (offspring) yang dihasilkan dari algoritma evolutionary programming yang dibentuk melalui tahap pengkodean, perhitungan fungsi *fitness*, mutasi, seleksi, dan kompetisi, dimasukkan sebagai input untuk algoritma tabu search. Melalui mekanisme perbaikan dan pembentukan individu baru yang layak maka status untuk input tabu search telah diperoleh. Melakukan seleksi dengan algoritma *Evolutionary Strategy* untuk mendapatkan status final atau hasil akhir.

Flowchart metode kombinasi *Evolutionary Programming – Tabu Search* sebagai berikut:


Start
Input data parameter
B
Set jumlah perent dengan menyesuaikan solusi yang ada untuk memperoleh masukan ke bentuk variabel-variabel status keadaan.
melakukan penjadwalan secara random
apakah kendala-kendala pada unit- dalam penjadwalan yang baru dengan menggunakan tabu search?
Tidak
Perbaikan penjadwalan
Ya
menghitung besarnya ED dan menghitung total biaya produksi
mutasi untuk membentuk individu baru
A



GAMBAR 2.16 SKEMA METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH*[1]

Langkah kerja dari metode kombinasi *evolutionary programming* dan *tabu search* untuk menyelesaikan masalah komitmen unit dapat diuraikan sebagai berikut:

1. Inisialisasi input parameter
2. Tentukan hasil jumlah populasi dari orang tua (parent) dengan menyesuaikan solusi yang ada untuk memperoleh masukan ke bentuk variabel-variabel status keadaan.
3. melakukan penjadwalan secara random atau acak pada unit-unit yang tidak dioperasikan/beroperasi (minimum up/down time)
4. melakukan pengecekan untuk kendala-kendala pada unit-unit yang tidak dioperasikan/beroperasi (minimum up/down time) dalam penjadwalan yang baru dengan menggunakan tabu search. Jika kendala tidak terpenuhi maka penjadwalan harus diperbaiki dengan menggunakan mekanisme perbaikan dan kembali ke langkah 3.
5. menghitung besarnya economi dispatch dan menghitung total biaya produksi untuk perent yang lainnya.
6. melakukan mutasi untuk membentuk individu baru (offspring) dengan menggunakan metode *Gaussian mutation*, dimana setiap individu akan mengalami beberapa operasi perbaikan dengan menggunakan mekanisme perbaikan. Kemudian penjadwalan baru di cek untuk verifikasi apakah semua kendala-kendala telah terpenuhi.
7. memperbaiki status dari individu baru dan verifikasi kendala-kendala (constraints) dengan *tabu search*.
8. merumuskan secara berurutan untuk seluruh populasi

9. memilih dan menentukan N jumlah populasi yang terbaik untuk iterasi selanjutnya
10. apakah penghitungan iterasi sudah dapat dilakukan. Jika "ya" lanjutkan ke langkah 11, jika "tidak" kembali ke langkah 2.
11. memilih dan menentukan populasi terbaik dengan strategi *evolutionary*.
12. cetak hasil penjadwalan yang optimum
13. stop

2.8.1 Mekanisme Perbaikan[1]

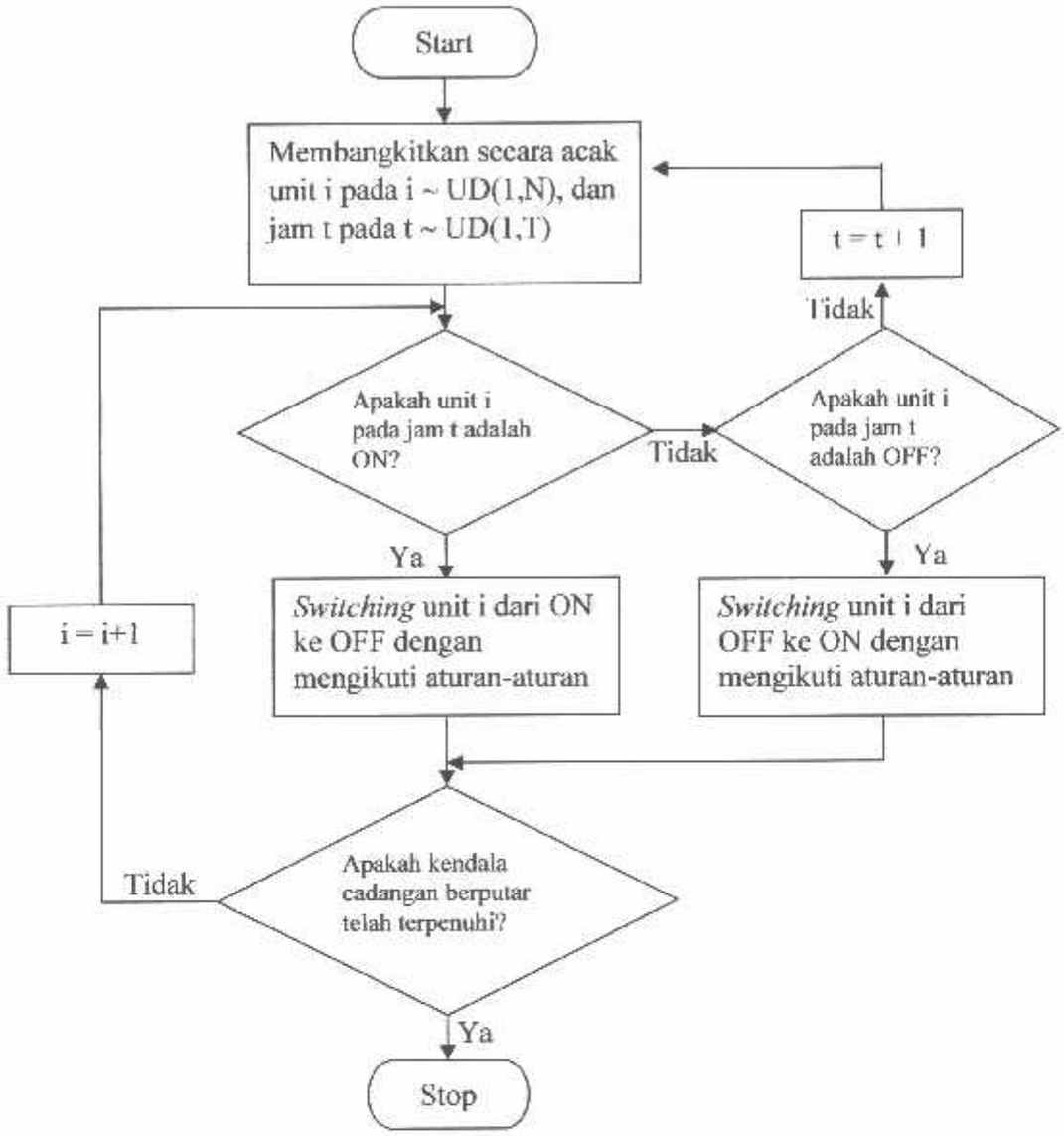
Mekanisme perbaikan dilakukan untuk memperbaiki kendala-kendala yang mungkin diterapkan dan diuraikan sebagai berikut:

a. Memilih dengan teliti secara acak pada satu dari unit yang tidak beroperasi atau dimatikan (off)
b. Membangkitkan solusi trial untuk menyeleksi unit i dari status OFF ke status ON dengan memperhatikan kemungkinan kendala minimum down time.
c. Melakukan pengecekan untuk kendala cadangan berputar pada jam tersebut, kemudian proses kembali pada jam yang sama untuk unit yang lainnya

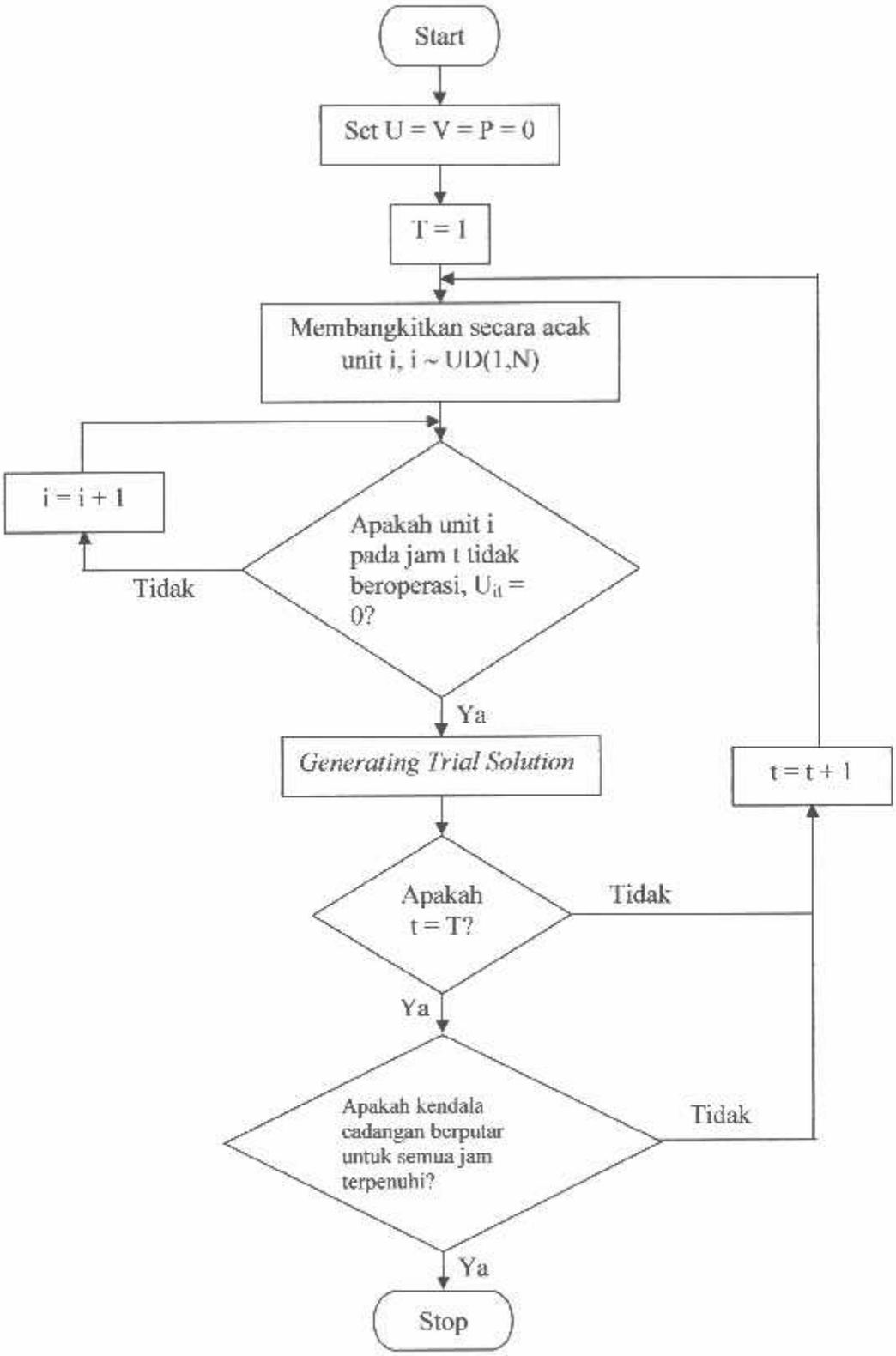
2.8.2 Pembentukan Individu Baru Yang Layak[1]

Untuk membentuk individu baru yang mungkin, seleksi dilakukan sebagai ""pemilihan kekuatan (*culling force*)" untuk mengeliminasi penjadwalan-penjadwalan yang mungkin. Sebelum solusi terbaik diseleksi dengan menggunakan *evolutionary strategy*, bagian kecil dari solusi terbaik telah dibentuk untuk mengoreksi mutasi-mutasi yang yang tidak dikehendaki.

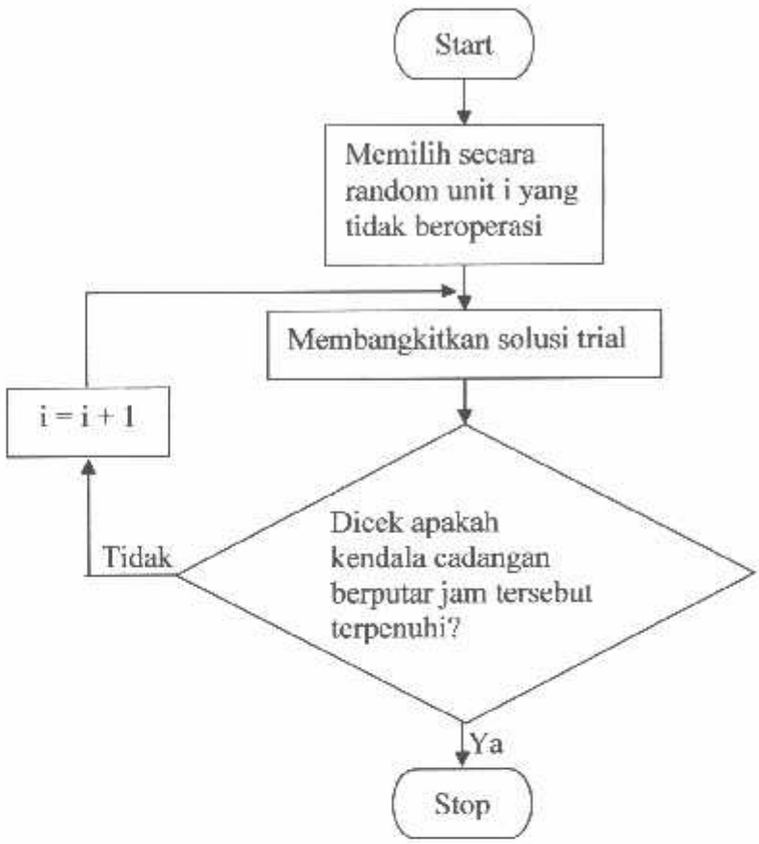
Flowchart Generating Trial Solution(Membangkitkan Solusi Trial)



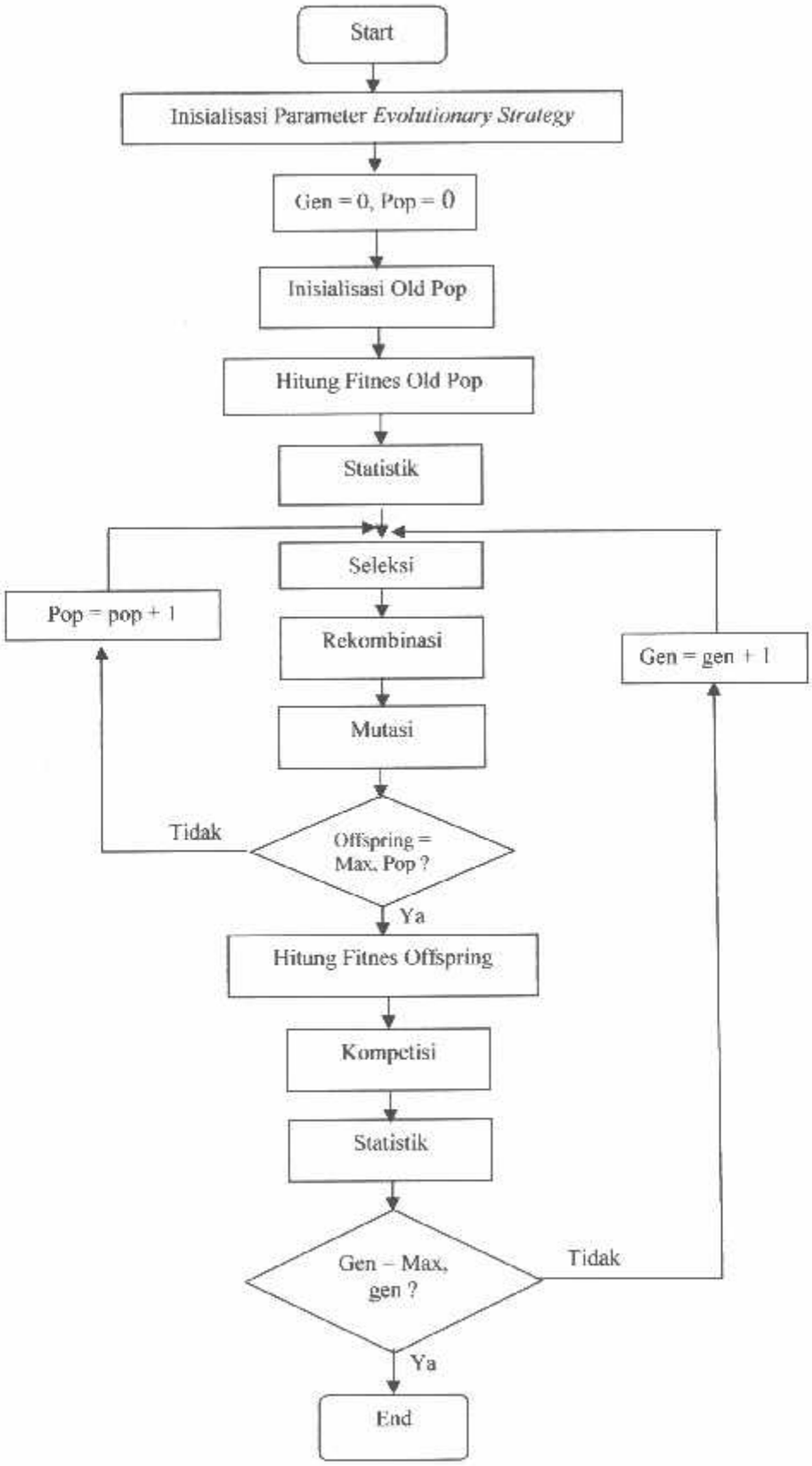
Flowchart Generating initial Solution(Membangkitkan solusi awal)



Flowchart Repair Mechanism(Mekanisme Perbaikan)



Flowchart Evolutionary Strategy



BAB III

PENERAPAN METODE *EVOLUTIONARY PROGRAMMING* DAN TABU SEARCH PADA PT. PEMBANGKITAN JAWA-BALI

3.1 Pendahuluan

PT. Pembangkitan Jawa Bali dalam menjalankan tugasnya sebagai penyedia kebutuhan energi listrik bagi masyarakat luas baik dalam masyarakat lapisan menengah atas sampai lapisan masyarakat menengah ke bawah, dituntut ubtuk memberikan pelayanan yang memuaskan disamping juga, harus memenuhi tujuan lainnya sebagai perusahaan yang bergerak dibidang jasa yaitu untuk mendapatkan keuntungan financial.

Dalam menyelenggarakan usaha ketenagalistrikan, berdasarkan prinsip industri dan perniagaan yang sehat, PT. Pembangkitan Jawa Bali dituntut mampu bersaing dan mampu memanfaatkan sebesar-besarnya peluang pasar dalam bidang tenaga listrik. Hal tersebut menuntut PT. Pembangkitan Jawa Bali harus menjaga efisiensi dan keandalan dalam operasional penyediaan tenaga listrik dari unit-unit pembangkit yang dimiliki.

Dengan demikian merupakan suatu keharusan bagi seluruh jajarannya untuk selalu berupaya untuk meningkatkan kondisi penyediaan tenaga listrik dari pembangkit agar lebih ekonomis, bermutu, dan didukung keandalan yang tinggi.

3.2 Data Unit-Unit Pembangkit Termal

Unit-unit pembangkit termal yang berada pada kawasan PT. Pembangkitan Jawa Bali berjumlah 37 unit pembangkit termal yang terdiri dari 5 blok pembangkit listrik tenaga gas dan uap, 11 unit pembangkit listrik tenaga uap, dan

5 unit pembangkit listrik tenaga gas. Adapun data-data lebih lengkapnya dapat dilihat pada table 3.1 dan 3.2, untuk harga bahan bakar berdasarkan data penawaran PLN untuk per Agustus 2002.

TABEL 3.1
DATA UNIT TERMAL PADA PT. PEMBANGKITAN JAWA BALI

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

Sumber: Data Penawaran PT. PJB, JL. Ketintang Baru 11 Surabaya 60231
Keterangan: MUT = minimum up time, MDT = minimum down time

TABEL 3.2
DATA BIAYA DAN PARAMETER UNIT-UNIT PEMABANGKIT TERMAL
PADA PT.PEMBANGKITAN JAWA BALI

No.	Nama Pembangkit	Biaya Start Up (Juta Rupiah)		Koefisien Biaya Bahan Bakar		
		Cold Start	Hot Start	a	b	c
1	PLTU Paiton 1	682.98	149.68	3244978	11172.15	10.2971
2	PLTU Paiton 2	682.98	149.68	3244978	11172.15	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	368.875
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	368.875
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	368.875
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 Gilitimur 1	6.13	0	687181.85	683240.965	1762.3893
22	PLTG Gilitimur 2	6.13	0	687181.85	683240.965	1762.3892
23	PLTGU M. Karang GT 1.1	7.35	0	5730795	202052.97	108.045
24	PLTGU M. Karang GT 1.2	7.35	0	5730795	202052.97	108.045
25	PLTGU M. Karang GT 1.3	7.35	0	5730795	202052.97	108.045
26	PLTGU M. Karang ST 1.0	54.52	29.67	11560815	53685.135	460.845
27	PLTGU M. Karang GT 1.1	0	0	14706521.25	433337.8	49.4605
28	PLTGU M. Karang GT 1.2	0	0	14706521.25	433337.8	49.4605
29	PLTGU M. Karang GT 1.3	0	0	14706521.25	433337.8	49.4605
30	PLTGU M. Karang GT 2.1	0	0	14706521.25	433337.8	49.4605
31	PLTGU M. Karang GT 2.2	0	0	14706521.25	433337.8	49.4605
32	PLTGU M. Karang ST 2.0	118.08	64.4	672630	144191.717	519.1757
33	PLTU M. Karang 1	122.58	31.08	2417820.7	473895.41	120.77935
34	PLTU M. Karang 2	122.58	31.08	2417820.7	473895.41	120.77935
35	PLTU M. Karang 3	122.58	31.08	2417820.7	473895.41	120.77935
36	PLTU M. Karang 4	215.34	89.29	2949187.5	205217.145	83.79
37	PLTU M. Karang 5	215.34	89.29	2949187.5	205217.145	83.79

Sumber: Data Penawaran PT. PJB, JL. Ketintang Baru 11 Surabaya 60231

Catatan:	Harga Batubara	253	Rp/Kg
	Harga MFO	1595,5	Rp/Liter
	Harga HSD	1595,5	Rp/Liter
	Harga gas UP. Gresik	2,53	US$/MMBTU
	Harga gas UP.M. Karang	2,45	US$/MMBTU
	Nilai Tukar	9000	Rp/USD

Data total biaya pembangkitan PT. Pembangkitan Jawa Bali yang digunKn sebagai data pembanding dari metode kombinasi *Evolutionary Programming* dan *Tabu Search* diperoleh dengan menggunakan persamaan pada subbab 2.3.5, dimana a, b, dan c merupakan konstanta dari fungsi biaya bahan bakar yang telah didapat dari data PT. Pembangkitan Jawa Bali pada table 3.2. Untuk tiap unit dan diselesaikan mulai dari unit pertama sampai pada unit terakhir yang beroperasi tiap periode atau jam.

3.3 Aplikasi Metode Kombinasi *Evolutionary Programming* dan *Tabu Search*

Perhitungan dan analisa dari metode kombinasi Evolutionary programming dan Tabu search dilakukan pada kebutuhan daya yang ditanggung PT. Pembangkitan Jawa Bali pada tanggal 27, 30, dan 31 Juli 2005. analisa data dilakukan untuk waktu tiga hari tersebut, karena dapat mewakili karateristik kurva yang berlainan dengan keterangan sebagai berikut:

- Tanggal 27 Juli 2005 adalah hari rabu, merupakan beban hari kerja
- Tanggal 30 Juli 2005 adalah hari sabtu, merupakan beban pada hari setengah penuh
- Tanggal 31 Juli 2005 adalah hari minggu, merupakan beban pada hari libur

Berdasarkan data unit termal yang terdapat pada PT. Pembangkitan Jawa Bali pada system tenaga listrik pada table 3.1, ternyata pada saat dilakukan

pengambilan data, semua unit pembangkit termal dalam kondisi siap beroperasi. Maka *input* data unit pembangkit termal dapat disusun yang siap beroperasi pada tanggal 27,30, dan 31 Juli 2005.

Data beban harian system yang diperoleh dari PT. Pembangkitan Jawa Bali, terdapat data hasil perhitungan mengenai jumlah total pembangkitan, beban total, dan cadangan berputar tiap jam dalam tiap-tiap area. Data-data ini tidak dipakai dalam skripsi ini, karena data tersebut menyangkut system secara keseluruhan dalam satu area. Dalam satu area terdapat lebih dari satu perusahaan penyedia energi listrik, seperti PT. Pembangkitan Jawa Bali dan PT Indonesia Power serta perusahaan milik swasta yang terdapat pada area IV.

Model perhitungan yang digunakan dalam melakukan perhitungan optimalisasi penjadwalan unit-unit pembangkit termal baik PLTU, PLTG, ataupun PLTGU dapat menggunakan karateristik tiap unit termal, meskipun karateristik tiap blok saling tergantung antara unit gas(gas turbin) atau GT dan unit uap(steam turbin) atau ST sehingga sering disebut *combined cycle*. Untuk memudahkan perhitungan total biaya operasi, dilakukan menggunakan pendekatan per unit termal, dimana data parameter tiap unit GT dapat diambil parameter kombinasi CC 3.3.1.

PT. Pembangkitan Jawa Bali tidak memiliki dasar yang pasti dalam menentukan besarnya cadangan berputartiap periode jam, tetapi PT. Pembangkitan Jawa Bali menggunakan asumsi bahwa nilai cadangan berputar diambil dari daya terpasang terbesar dari unit pembangkit yang mengalami gagal operasi. Dalam hal ini PT. Pembangkitan Jawa Bali daya terpasang terbesar

adalah unit pembangkit PLTU Paiton yang memiliki daya terpasang sebesar 400 MW sebagai nialai cadangan berputar tiap periode jamnya.

3.4 Beban Sistem

Unit-unit Pembangkit yang ada dalam wilayah Jawa – Bali dikoordinasi oleh PT. Pembangkitan Jawa Bali. Proses penjadwalan unit-unit pembangkit dengan menggunakan metode kombinasi *Evolutionary Programming* dan *Tabu Search* bertujuan untuk membuat perencanaan penjadwalan unit pembangkit dalam system tenaga listrik untuk dapat memenuhi kebutuhan beban system dengan biaya operasi yang seekonomis mungkin.

Untuk dapat mengetahui besarnya pengaruh, dan efisiensi dari metode kombinasi ini, maka dilakukan evaluasi dan analisa dengan mengambil data unit pembangkit termal dan data beban yang ditanggung PT. Pembangkitan Jawa Bali. Untuk kombinasi penjadwalan dan daya output pembangkit tenaga listrik dalam system PT. Pembangkitan Jawa Bali dapat diambil data pada tanggal 27, 30, 31 Juli 2005.

TABEL 3.3
DATA BEBAN UNIT-UNIT PEMABANGKIT TERMAL
PADA PT. PEMBANGKITAN JAWA BALI

JAM	Rabu 27 Juli 2005		Sabtu 30 Juli 2005		Minggu 31 Juli 2005	
	Beban Sistem	Cadangan Berputar	Beban Sistem	Cadangan Berputar	Beban Sistem	Cadangan Berputar
01.00	2300	400	2525	400	2275	400
02.00	2175	400	2300	400	1755	400
03.00	2090	400	2170	400	1755	400
04.00	2090	400	2170	400	1740	400
05.00	2240	400	2470	400	1895	400
06.00	2215	400	2250	400	1970	400
07.00	1990	400	1940	400	1642	400
08.00	2250	400	2065	400	1565	400
09.00	2540	400	2190	400	1615	400
10.00	2590	400	2190	400	1675	400
11.00	2590	400	2210	400	1625	400
12.00	2340	400	2165	400	1575	400
13.00	2575	400	2140	400	1575	400
14.00	2575	400	2190	400	1575	400
15.00	2575	400	2265	400	1575	400
16.00	2475	400	2130	400	1575	400
17.00	2457	400	2197	400	1689	400
18.00	2951	400	2849	400	2689	400
19.00	2981	400	2989	400	2929	400
20.00	2981	400	2934	400	2924	400
21.00	2951	400	2914	400	2904	400
22.00	2664	400	2582	400	2632	400
23.00	2430	400	2375	400	2330	400
24.00	2405	400	2300	400	2215	400

Sumber: Data Penawaran PT. PJB, JL. Ketintang Baru 11 Surabaya 60231

3.5 Data Validasi Program

Pengujian validasi program dilakukan dengan menggunakan jurnal C. Christoper Asir Rajan, M. R. Mohan, " **an evolutionary programming based tabu search method for Solving Unit Commitment Problem**", IEEE Trans. On Power System, Vol. 19, No. 1, Februari 2004, yang digunakan sebagai data validasi program. Karena pada referensi diatas tidak dicantumkan *minimum up*

BAB IV

ANALISA DATA DENGAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING DAN TABU SEARCH* PADA PT. PJB

4.1 Program Komputer Metode Kombinasi EP dan TS

Untuk pemecahan masalah komitmen unit diguanakan bantuan program komputer. Program computer ini digunakan untuk mempercepat proses perhitungan yang membutuhkan tingkat ketelitian dan akurasi tinggi dan sering melibatkan iterasi yang memerlukan waktu yang lama bila dikerjakan secara manual.

Program komputer ini menggunakan bahasa pemrograman Borland Delphi versi 8.0 yang dijalankan pada komputer AMD Athlon XP 2,4 GHz. Bahasa pemrograman Borland Delphi versi 8.0 merupakan bahasa pemrograman terstruktur yang relative lebih mudah untuk dipelajari dan dimengerti serta mudah penggunaannya.

4.2 Algoritma Program

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

1. Inisialisasi input parameter unit-unit pembangkit termal dan data pembebanan harian untuk periode 24 jam. Data parameter meliputi: jumlah unit pembangkit, daya maksimum dan daya minimum, konstanta persamaan biaya bahan bakar, harga bahan bakar, biaya start baik start pada kondisi dingin maupun dalam kondisi panas, *minimum up time* dan *minimum down time*.

2. Inisialisasi parameter-parameter *Evolutionary Programming* yang meliputi: konstanta swap, konstanta perbaikan(*repair*), populasi maksimum, *maximum generation*, konstanta β, konstanta α, konstanta M, probabilitas mutasi.
3. Inisialisasi parameter-parameter *Tabu Search* yaitu memasukkan suatu nilai awal dalam daftar tabu untuk setiap periode waktu jam.
4. Tentukan nilai awal *counter* yaitu sama dengan nol.
5. Tentukan hasil jumlah populasi dari orang tua(perent) dengan menyesuaikan solusi yang ada untuk memperoleh masukan ke bentuk variabel-variabel status keadaan.
6. Lakukan penjadwalan secara random atau acak pada unit-unit yang tidak dioperasikan/beroperasi dengan memperhatikan minimum up/down time
7. Hitung fungsi fitness pada *parent* dan melakukan proses statistic pada *parent*.
8. Periksa untuk kendala-kendala pada unit-unit yang tidak dioperasikan/beroperasi dengan memperhatikan minimum up/down time dalam penjadwalan yang baru dengan menggunakan tabu search yaitu dengan membangkitkan solusi coba coba yang merupakan suatu proses pengacakan pada tiap unit pembangkit dari status ON dan OFF. Jika kendala tidak terpenuhi maka penjadwalan harus diperbaiki dengan menggunakan mekanisme perbaikan dan kembali ke langkah 3.

   Adapun algoritma mekanisme perbaikan sebagai berikut:

   8.1 Pilih dengan teliti secara acak salah satu unit yang tidak beroperasi atau dimatikan(off)

8.2 Bangkitkan solusi trial untuk menyeleksi unit i dari status OFF ke status ON dengan memperhatikan kemungkinan kendala minimum down time.

Adapun langkah-langkah membangkitkan solusi trial sebagai berikut:

Langkah 8.2.1: Tentukan secara acak unit i, i ~ (1, N); dan satu jam t, t ~ (1, T). yang menunjukkan status beberapa unit i dalam suatu periode jam dalam keadaan Unit ON atau OFF.

Langkah 8.2.2: Jika unit *i* pada jam t adalah ON, maka lanjutkan ke langkah 8.2.3 untuk mempelajari kondisi OFF pada jam *t* ini. Sebaliknya, jika unit *i* pada jam *t* adalah OFF, maka lanjutkan ke langkah 8.2.4 untuk mempelajari kondisi ON pada waktu *t* ini.

Langkah 8.2.3: Perubahan unit *i* dari ON ke OFF

a. Bergerak dari jam *t* mundur dan maju dalam waktunya, untuk menentukan panjang periode ON..
b. Jika T_{on} = MUT, maka unit ditetapkan dalam kondisi OFF di semua jam yang melibatkan T_{on}
c. Jika T_{on} > MUT, maka tetapkan L ~ (1, T_{on} – MUT)
d. Matikan unit dalam kondisi OFF pada jam t_1, t_1+1, ..., t_1+L – 1. dimana t_1 adalah jam pertama saat unit dalam kondisi ON.

Langkah 8.2.4: Perubahan unit dari OFF ke ON

a. Bergerak dari jam *t* mundur dan maju dalam waktunya, untuk menentukan panjang periode OFF.

b. Jika T_{off} = MDT, maka unit ditetapkan dalam kondisi ON di semua jam yang melibatkan T_{off}.

c. Jika T_{off} > MDT, maka tetapkan L ~ (1, T_{off} – MDT)

d. Tetapkan unit dalam kondisi ON pada jam t_3, t_3+1, ..., t_3+L – 1. dimana t_3 adalah jam pertama saat unit dalam kondisi OFF.

Langkah 8.2.5: Periksa cadangan berputar; periksa pemenuhan cadangan berputar untuk periode waktu unit berubah dalam langkah 8.2.3 dan langkah 8.2.4. Jika ini bisa memenuhi, maka solusi trialnya bisa digunakan, jika tidak memenuhi, maka kembali ke langkah 8.2.1.

9. Hitung besarnya economic dispatch dan hitung total biaya produksi untuk *parent* yang lainnya.

10. Lakukan mutasi untuk membentuk individu baru(*offspring*)dengan menggunakan metode Gaussian mutation, dimana setiap individu akan mengalami beberapa operasi perbaikan dengan menggunakan mekanisme perbaikan, yaitu dengan melakukan pembentukan individu baru yang layak. Kemudian penjadwalan baru di cek untuk verifikasi apakah semua kendala-kendala telah terpenuhi.

    Adapun mekanisme pembentukan individu baru yang layak sebagai berikut:

    Langkah 10.1: Tetapkan $U_{it} = V_{it} = P_{it} = 0$. tentukan $t = 1$

    Langkah 10.2: Lakukan Proses berikut:

    a. Bangkitkan secara random sebuah unit i, i ~ (1, N)

b. Jika unit i pada jam t adalah OFF ($U_{it} = 0$) maka lanjutkan ke langkah 10.3. jika tidak, lanjutkan ke langkah 2b untuk memilih unit yang lainnya.

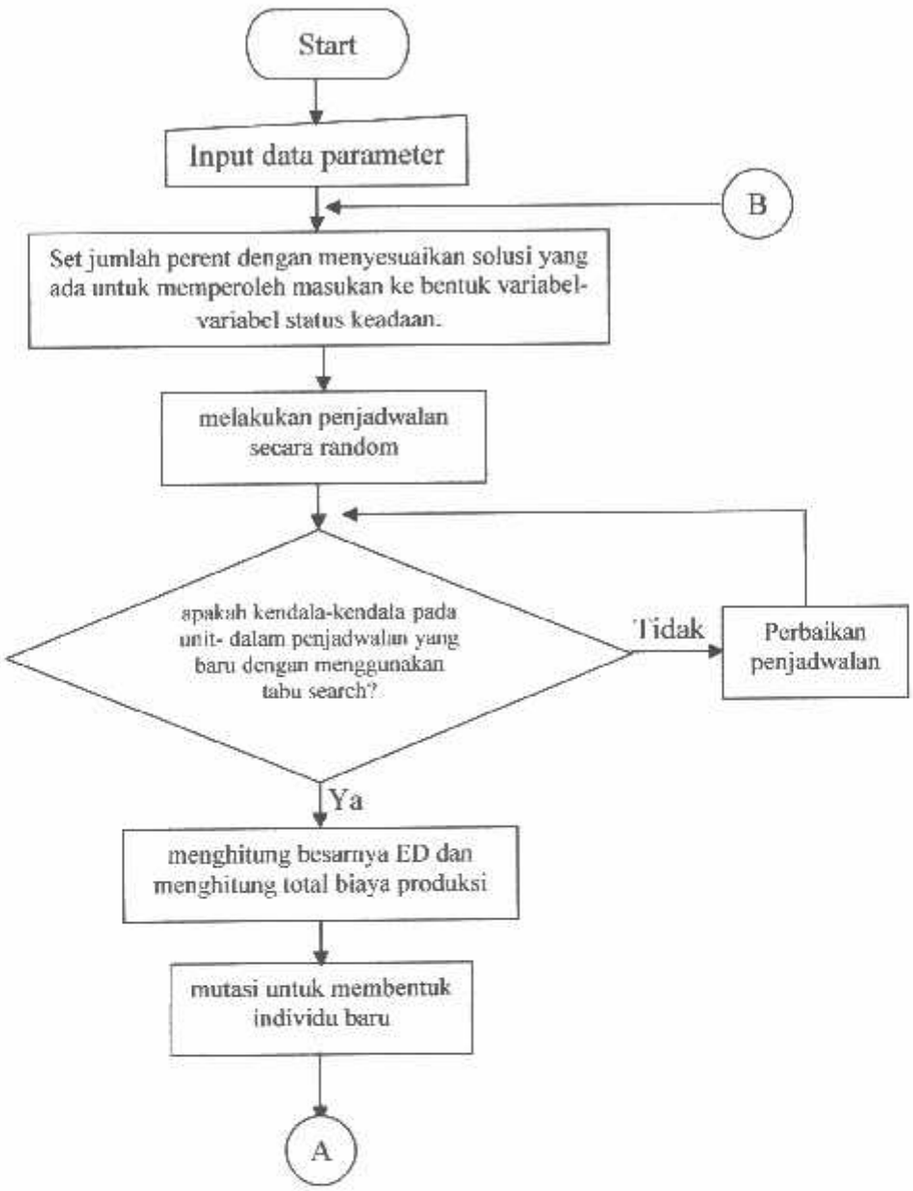
Langkah 10.3: Mengikuti prosedur dalam langkah 10.4, pada mekanisme perbaikan, untuk melihat kemungkinan menjalankan unit ON mulai jam t

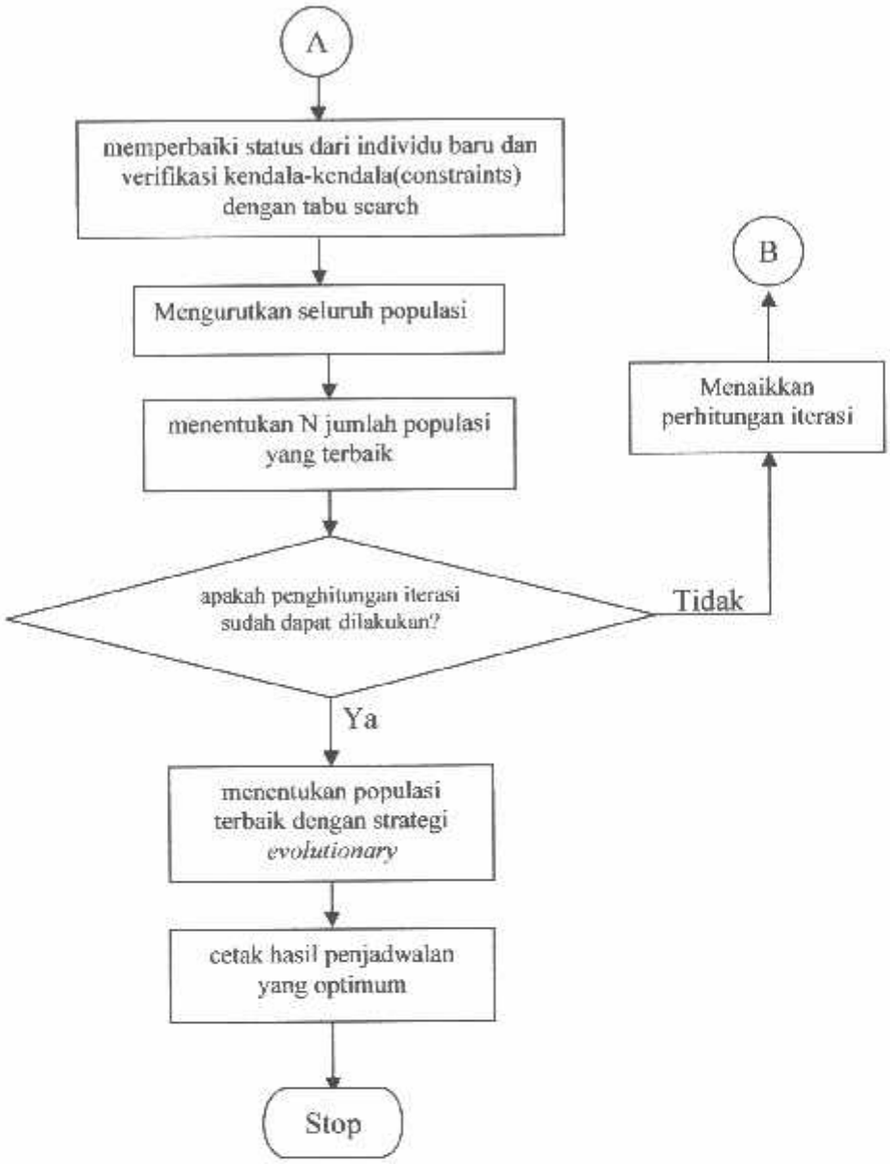
Langkah 10.4: Jika t = T, lanjutkan ke langkah 10.5, dan jika tidak, tetapkan $t = t - 1$ dan kembali ke langkah 10.2.

Langkah 10.5: Periksa batasan cadangan berputar pada semua jam. Ulangi langkah 10.2 dan langkah 10.3 untuk jam yang mana batasan tidak bisa dipenuhi.

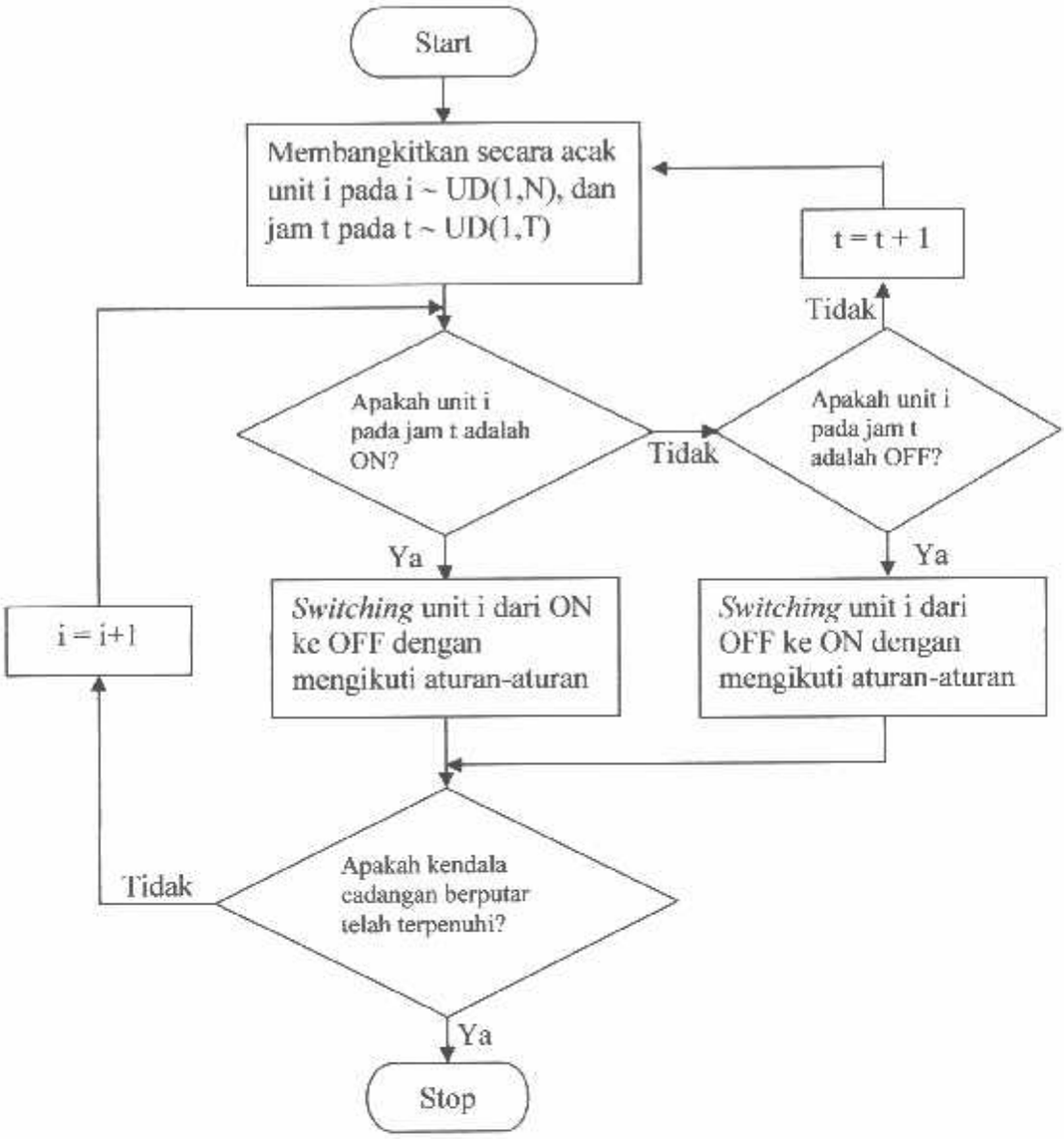
11. Perbaiki status dari individu baru dan verifikasi kendala-kendala(constraints) dengan tabu search dengan menggunakan mekanisme perbaikan .
12. Rumuskan secara berurutan untuk seluruh populasi berdasarkan nilai fitnesnya.
13. Pilih dan tentukan N jumlah populasi yang terbaik untuk iterasi selanjutnya
14. apakah penghitungan iterasi sudah dapat dilakukan. Jika "ya" lanjutkan ke langkah 11, jika "tidak" kembali ke langkah 2.
15. Pilih dan tentukan populasi terbaik dengan strategi *evolutionary*.
16. cetak hasil penjadwalan yang optimum
17. stop

4.3 Flowchart Metode Kombinasi Evolutionary Programming – Tabu Search

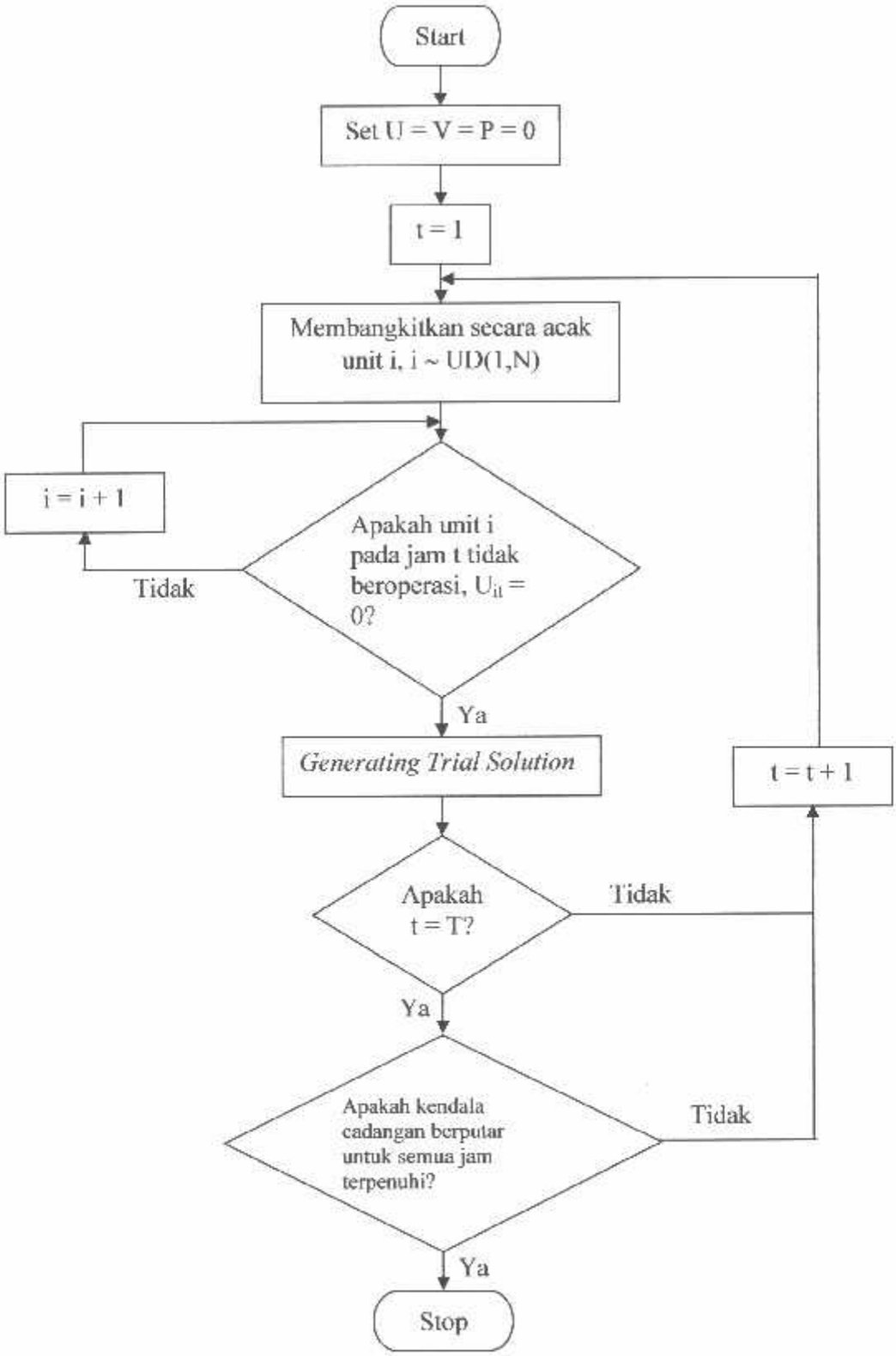



A
memperbaiki status dari individu baru dan verifikasi kendala-kendala(constraints) dengan tabu search
Mengurutkan seluruh populasi
menentukan N jumlah populasi yang terbaik
apakah penghitungan iterasi sudah dapat dilakukan?
Tidak
B
Menaikkan perhitungan iterasi
Ya
menentukan populasi terbaik dengan strategi evolutionary
cetak hasil penjadwalan yang optimum
Stop

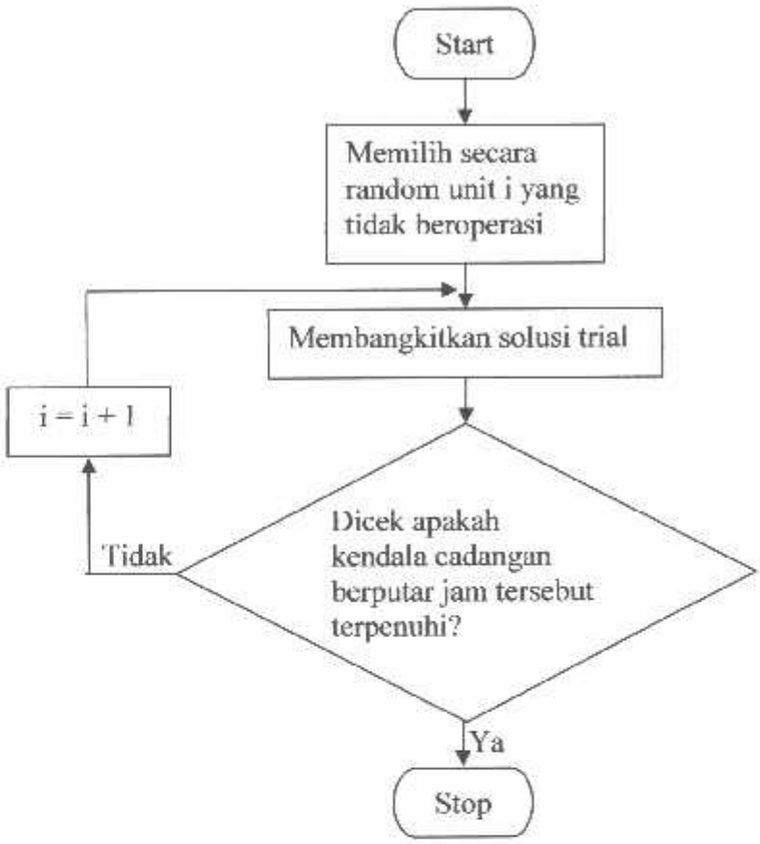
Flowchart Generating Trial Solution(Membangkitkan Solusi Trial)



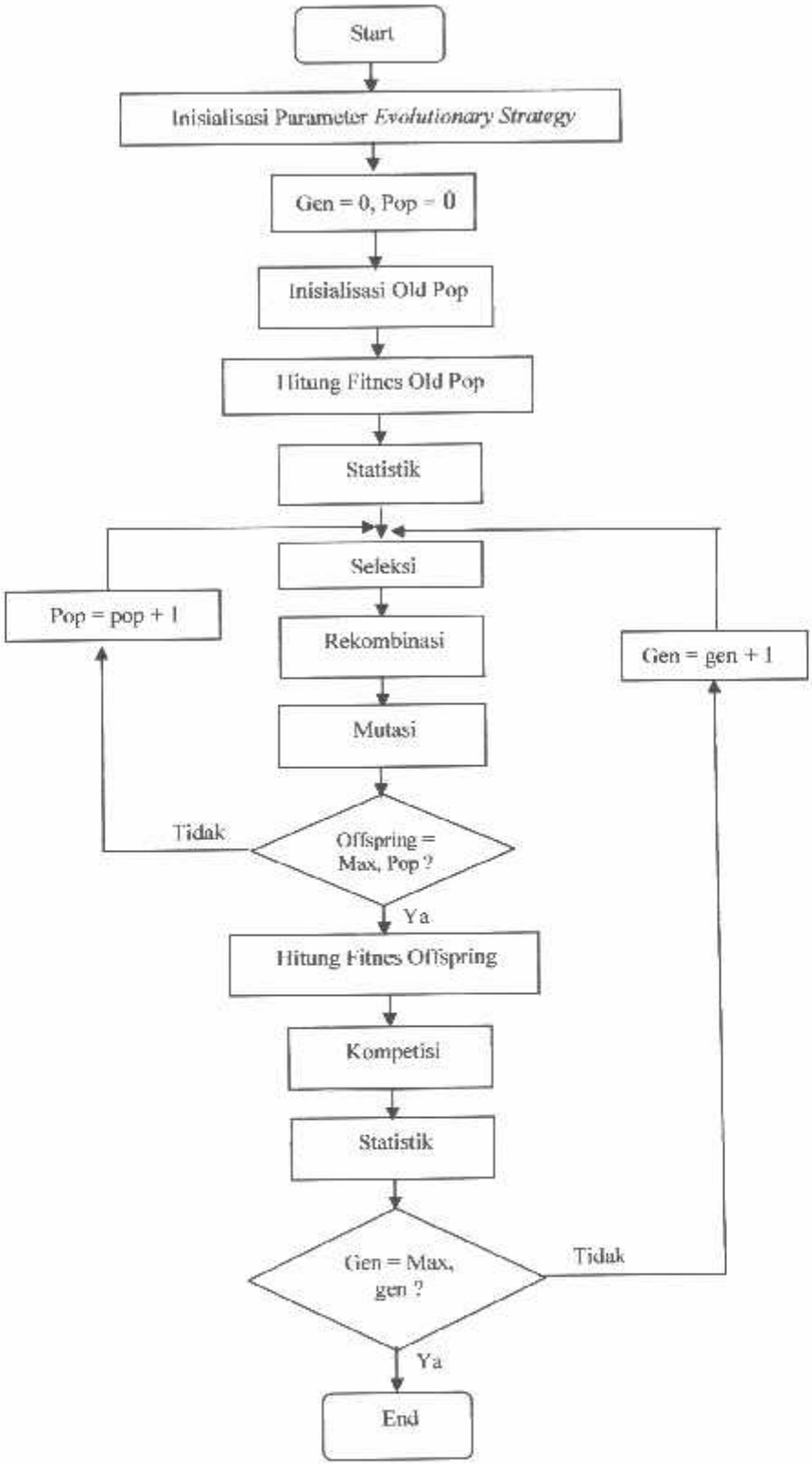
Flowchart Generating initial Solution(Membangkitkan solusi awal)



Flowchart Repair Mechanism(Mekanisme Perbaikan)



Flowchart Evolutionary Strategy



4.4 Uji Validasi

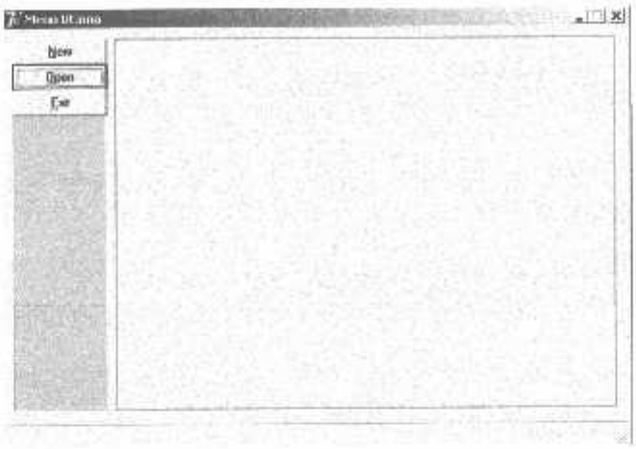
Sebelum dilakukan optimasi perhitungan biaya total, terlebih dahulu dilakukan uji validasi program untuk melihat kelayakan dari program tersebut. Dalam program ini dilakukan pengujian yang berpedoman pada jurnal C. Christoper Asir Rajan, M. R. Mohan, " **An evolutionary programming based tabu search method for Solving Unit Commitment Problem**", IEEE Trans. On Power System, Vol. 19, No. 1, Februari 2004. hasil pengujian dapat dilihat pada gambar 4.2. Kemudian dilakukan perhitungan biaya total pembangkitan untuk 10 unit dalam tiap jamnya.

Table 4.1
Data Parameter 10 Unit Untuk Validasi

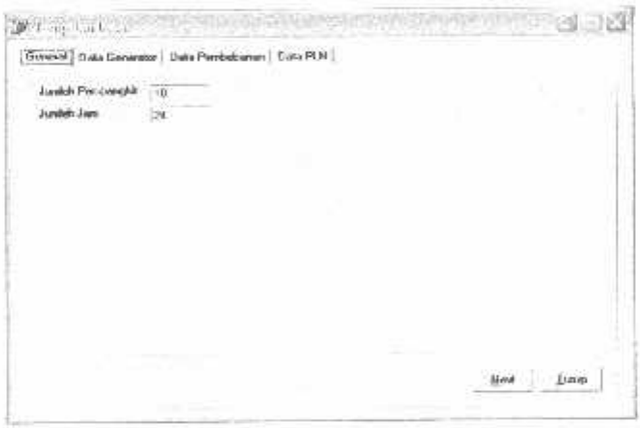
No. Unit	P_{max} (MW)	P_{min} (MW)	a	b	c	MUT h	MDT h	Hot Start Cost $	Cold Start cost $	Hot Start Hours h	Initial Status h
1	455	150	1000	16.19	0.00048	8	8	4500	9000	5	8
2	455	150	970	17.26	0.00031	8	8	5000	10000	5	8
3	130	20	700	16.60	0.00200	5	5	550	1100	4	-5
4	130	20	680	16.50	0.00211	5	5	560	1120	4	-5
5	162	25	450	19.70	0.00398	6	6	900	1800	4	-6
6	80	20	370	22.26	0.00712	3	3	170	340	2	-3
7	85	25	480	27.74	0.00079	3	3	260	520	2	-3
8	55	10	660	25.92	0.00413	1	1	30	60	0	-1
9	55	10	665	27.27	0.00222	1	1	30	60	0	-1
10	55	10	670	27.79	0.00173	1	1	30	60	0	-1

Tabel 4.2
Data Beban Sistem Untuk Validasi

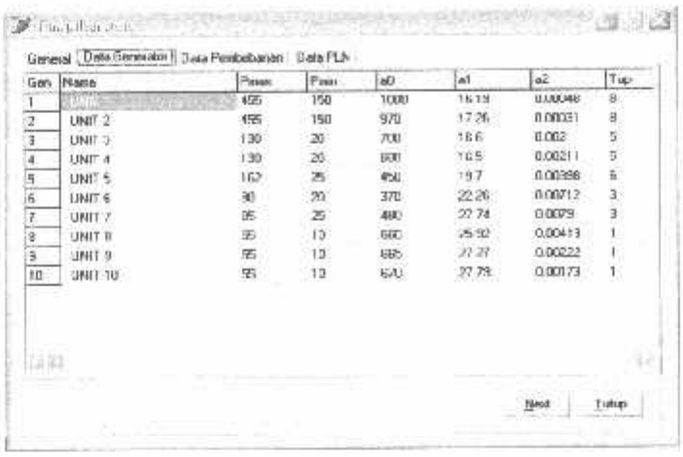
JAM	BEBAN SISTEM (MW)	Cadangan Berputar (MW)
01.00	700	70
02.00	750	75
03.00	850	85
04.00	950	95
05.00	1000	100
06.00	1100	110
07.00	1150	115
08.00	1200	120
09.00	1300	130
10.00	1400	140
11.00	1450	145
12.00	1500	150
13.00	1400	140
14.00	1300	130
15.00	1200	120
16.00	1050	105
17.00	1000	100
18.00	1100	110
19.00	1200	120
20.00	1400	140
21.00	1300	130
22.00	1100	110
23.00	900	90
24.00	800	80



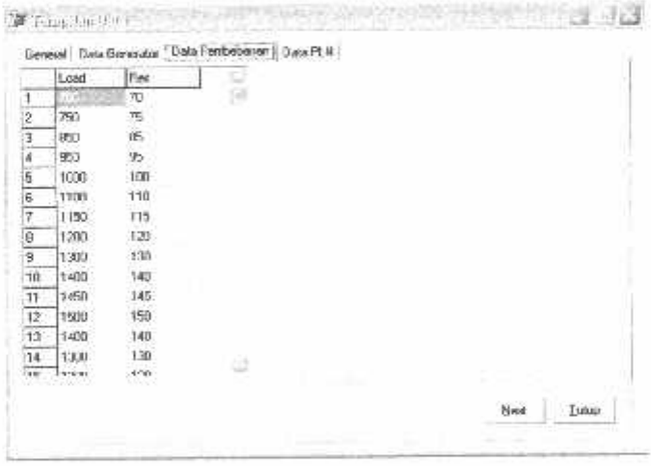
GAMBAR 4.1
TAMPILAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH*



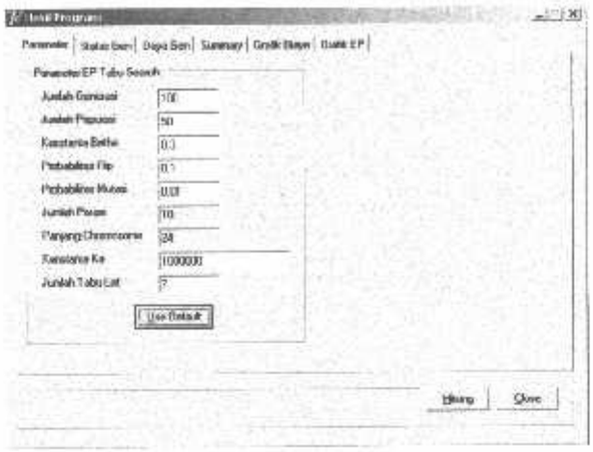
GAMBAR 4.2. TAMPILAN DATA UNTUK VALIDASI



GAMBAR 4.3. DATA GENERATOR UNTUK VALIDASI



GAMBAR 4.4. DATA PEMBEBANAN UNTUK VALIDASI



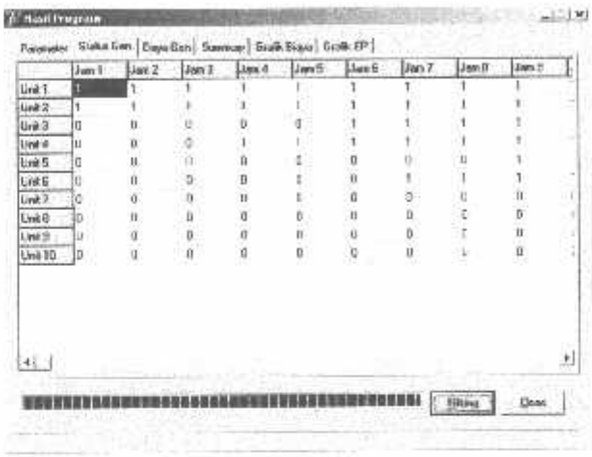
GAMBAR 4.5 TAMPILAN PARAMETER VALIDASI

Dimana:

- Jumlah tabu list menunjukkan banyaknya daftar yang digunakan dalam menyimpan hasil perhitungan dari solusi-solusi terbaik yang pernah ditemukan untuk disimpan sementara kemudian dengan ditemukan solusi yang baru dilakukan seleksi apakah hasil solusi terkini lebih baik dari solusi yang pernah ada dalam daftar. Apabila solusi terkini lebih baik dari solusi yang ada dalam daftar, maka solusi yang lama akan diganti dengan solusi baru yang lebih baik.
- Jumlah generasi Merupakan jumlah perulangan (iterasi) dilakukannya rekombinasi dan seleksi. Jumlah generasi ini mempengaruhi kestabilan output dan lama iterasi. Jumlah generasi yang besar dapat mengarahkan kearah solusi yang optimal, namun akan membutuhkan waktu yang lama. Sedangkan jika jumlah generasinya terlalu sedikit maka solusi akan terjebak pada *local optimum solution*
- Jumlah Populasi merupakan banyaknya populasi yang digunakan dalam melakukan pencarian solusi. Jumlah populasi dalam program dapat ditentukan, yaitu populasi awal dari solusi *parent M* yang dibangkitkan secara acak yang masing-masing populasi dari *parent* dihitung fungsi objektifnya sesuai besarnya populasi. Kemudian membentuk *offspring* dari *parent* yang layak dengan melakukan mutasi dan menambah variable acak gausian.

Sehingga sebelum dilakukannya seleksi dan kompetisi jumlah populasi adalah $(2M + 1)$. Pada jurnal jumlah populasi ditentukan besarnya 50 populasi.

Perbandingan hasil perhitungan antara komputasi program dan data hasil perhitungan adalah sebagai berikut:



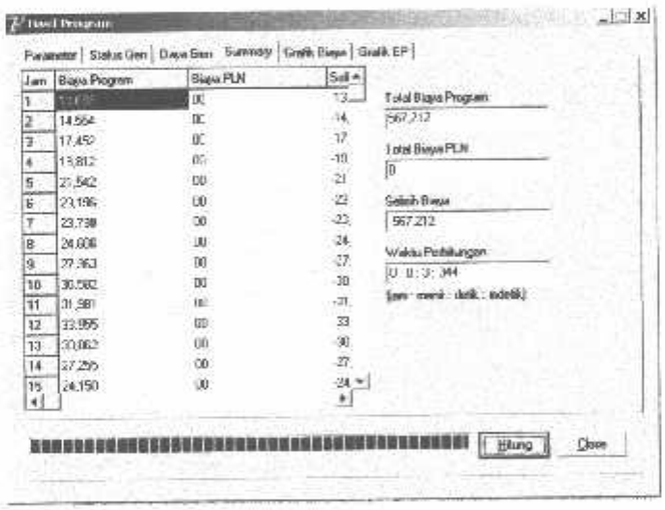
GAMBAR 4.6. HASIL PERHITUNGAN PENJADWALAN UNIT PEMBANGKIT MENURUT PROGRAM

Tabel 4.3
Hasil Perhitungan Penjadwalan Unit Pembangkit Menurut Program

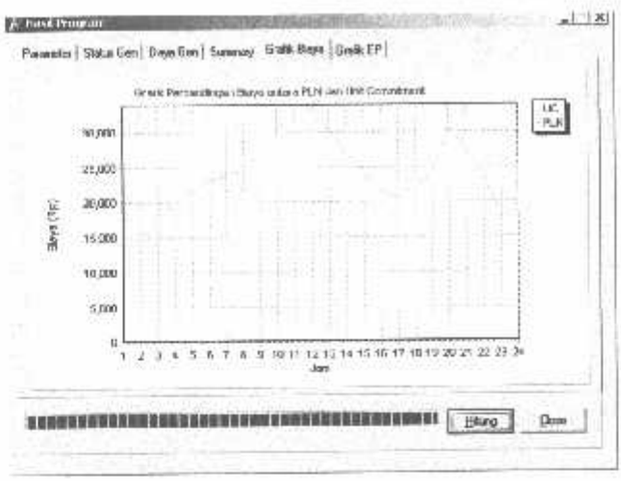
Jam / Unit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
4	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
6	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabel 4.4
Hasil Perhitungan Penjadwalan Unit Pembangkit menurut jurnal

Jam	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unit																								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
5	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
6	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0



GAMBAR 4.7 HASIL VALIDASI PROGRAM TERHADAP JURNAL



GAMBAR 4.8 KURVA BIAYA OPRASIONAL HASIL VALIDASI PROGRAM TERHADAP JURNAL

Dari hasil pengujian disini dapat dilihat bahwa program tersebut layak digunakan, karena dari hasil perhitungan program tersebut mendekati hasil yang ada di jurnal C. Christoper Asir Rajan, M. R. Mohan, " **An evolutionary programming based tabu search method for Solving Unit Commitment Problem**", IEEE Trans. On Power System, Vol. 19, No. 1, Februari 2004. dengan menghasilkan biaya total USD 567,212. Pada tampilan tersebut dapat dilihat bahwa hasil perhitungan dari metode kombinasi *Evolutionary Programming* dan *Tabu Search* mendekati yang ada di jurnal C. Christoper Asir Rajan, M. R. Mohan, " **An evolutionary programming based tabu search method for Solving Unit Commitment Problem**", IEEE Trans. On Power System, Vol. 19, No. 1, Februari 2004 yaitu USD 565,646, dengan selisih biaya sebesar USD 1,566 (0,28%). Sehingga program tersebut layak digunakan.

4.5 Hasil Perhitungan dan Analisa Data

Program optimasi pembebanan unit pembangkit termal pada PT. PJB dengan metode kombinasi *Evolutionary Programming* dan *Tabu Search*, terdiri dari bebrapa tahap yang secara keseluruhan dijabarkan sebagai berikut:

1. Tahap input data dengan inisialisasi data karatcristik tiap unit pembangkit dan data beban sistem tiap jamnya.
2. Mencari perhitungan dan pencarian nilai yang paling minimum untuk biaya, beban sistem, dan cadangan berputar.
3. Pencarian kombinasi penjadwalan unit-unit pembangkit yang paling murah untuk melayani beban sistem pada tiap jamnya.

Scluruh unit termal yang siap beroperasi dalam PT. Pembangkitan Jawa Bali terdiri dari 37 unit pembangkit. Pola kombinasi dari 37 unit pembangkit pada periode waktu 24 jam yaitu pada tanggal 27 juli, 30 juli, dan 31 juli 2005 dapat dilihat pada tabel 4.3, 4.4, dan 4.5 yang digunakan sebagai bahan perbandingan dengan hasil kombinasi metode kombinasi *Evolutionary Programming* dan *Tabu Search*.

Table 4.5
Kombinasi Unit pembangkit Termal
Pada Kamis, 27 Juli 2005

Jam	Status ON dan OFF Unit Pembangkit	Beban (MW)	Status ON Unit Pembangkit
01.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2300	14 unit
02.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2175	14 unit
03.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2090	13 unit
04.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2090	13 unit
05.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2240	13 unit
06.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2215	13 unit
07.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	1990	13 unit
08.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2250	13 unit
09.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2540	13 unit
10.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2590	13 unit
11.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2590	13 unit
12.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2340	13 unit
13.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2575	13 unit
14.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2575	13 unit
15.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2575	13 unit
16.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1	2475	13 unit
17.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2457	18 unit
18.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2951	18 unit
19.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2981	18 unit
20.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2981	18 unit
21.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2951	18 unit
22.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2664	18 unit
23.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2430	14 unit
24.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1	2400	13 unit

Keterangan : 1 = *ON* dan 0 = *OFF*

Tabel 4.6
Kombinasi Unit pembangkit Termal
Pada Sabtu, 30 Juli 2005

Jam	Status ON dan OFF Unit Pembangkit	Beban (MW)	Status ON Unit Pembangkit
01.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2525	13 unit
02.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2300	13 unit
03.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2170	12 unit
04.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2170	12 unit
05.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2470	12 unit
06.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2250	12 unit
07.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	1940	12 unit
08.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2065	12 unit
09.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2190	12 unit
10.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2190	12 unit
11.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2210	12 unit
12.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2165	12 unit
13.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2140	12 unit
14.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2190	12 unit
15.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2265	12 unit
16.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2130	12 unit
17.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2197	15 unit
18.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2849	15 unit
19.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2989	15 unit
20.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2934	15 unit
21.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2914	15 unit
22.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2582	15 unit
23.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0	2375	13 unit
24.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0	2300	12 unit

Keterangan : 1 = *ON* dan 0 = *OFF*

Tabel 4.7
Kombinasi Unit pembangkit Termal
Pada Minggu, 31 Juli 2005

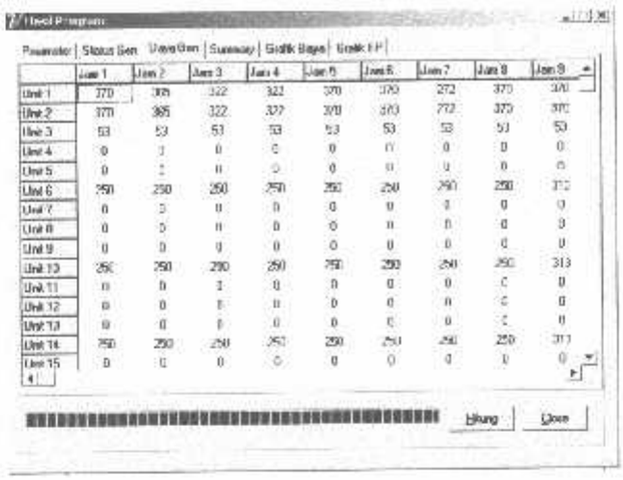
Jam	Status ON dan OFF Unit Pembangkit	Beban (MW)	Status ON Unit Pembangkit
01.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	2275	12 unit
02.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1755	12 unit
03.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1755	12 unit
04.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1740	12 unit
05.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1895	12 unit
06.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1970	12 unit
07.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1642	12 unit
08.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1565	12 unit
09.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1615	12 unit
10.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1675	12 unit
11.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1625	12 unit
12.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1575	12 unit
13.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1575	12 unit
14.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1575	12 unit
15.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	1575	12 unit
16.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 10	1575	15 unit
17.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 10	1689	14 unit
18.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 10	2689	14 unit
19.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 10	2929	14 unit
20.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 10	2924	14 unit
21.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 10	2904	14 unit
22.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 10	2632	14 unit
23.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	2330	12 unit
24.00	1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 10	2215	12 unit

Keterangan : 1 = *ON* dan 0 = *OFF*

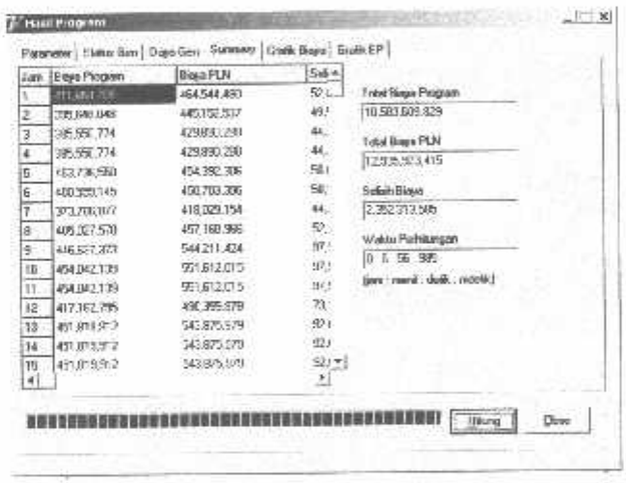
Tabel 4.8
Kombinasi Unit pembangkit Termal
Dengan Metode Kombinasi *Evolutionary Programming dan Tabu Search*
Pada Rabu, 27 Juli 2005

Jam	Status ON dan OFF Unit Pembangkit	Beban (MW)	Status ON Unit Pembangkit
01.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2300	10 unit
02.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2175	10 unit
03.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2090	10 unit
04.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2090	10 unit
05.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2240	10 unit
06.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2215	10 unit
07.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1990	10 unit
08.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2250	10 unit
09.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2540	10 unit
10.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2590	10 unit
11.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2590	10 unit
12.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2340	10 unit
13.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0	2575	11 unit
14.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2575	12 unit
15.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2575	12 unit
16.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2475	13 unit
17.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2457	14 unit
18.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2951	14 unit
19.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2981	15 unit
20.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2981	15 unit
21.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2951	15 unit
22.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2664	15 unit
23.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1	2430	15 unit
24.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1	2405	15 unit

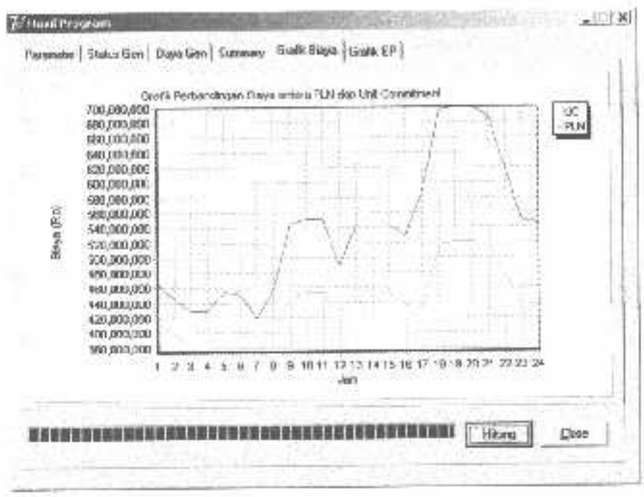
Keterangan : 1 = *ON* dan 0 = *OFF*



GAMBAR 4.9 DAYA GENERATOR PADA TANGGAL 27 JULI 2005



GAMBAR 4.10 HASIL PERHITUNGAN OPTIMASI PADA TANGGAL 27 JULI 2005

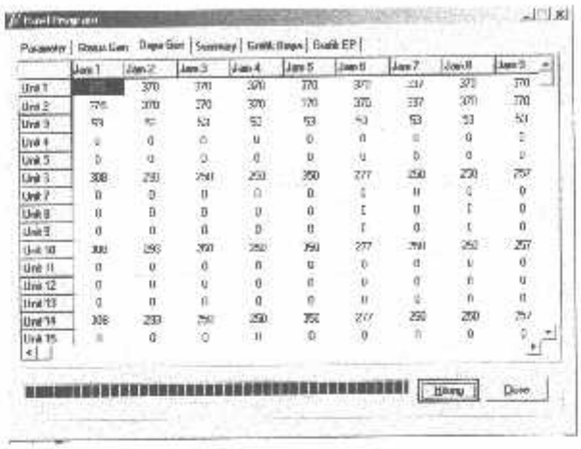


GAMBAR 4.11 KURVA HASIL PERHITUNGAN OPTIMASI PADA TANGGAL 27 JULI 2005

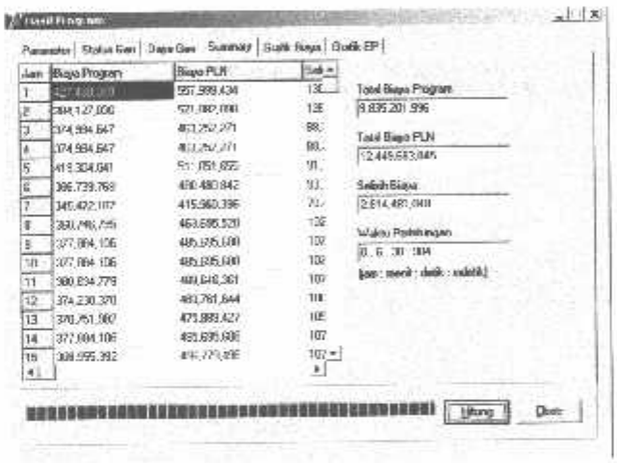
Tabel 4.9
Kombinasi Unit pembangkit Termal
Dengan Metode Kombinasi *Evolutionary Programming dan Tabu Search*
Pada Sabtu, 30 Juli 2005

Jam	Status ON dan OFF Unit Pembangkit	Beban (MW)	Status ON Unit Pembangkit
01.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2525	8 unit
02.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2300	8 unit
03.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2170	8 unit
04.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2170	8 unit
05.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2470	8 unit
06.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2250	8 unit
07.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1940	8 unit
08.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2065	8 unit
09.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2190	8 unit
10.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2190	8 unit
11.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2210	8 unit
12.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2165	9 unit
13.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2140	9 unit
14.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2190	10 unit
15.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2265	10 unit
16.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2130	10 unit
17.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0	2197	12 unit
18.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2849	14 unit
19.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2989	14 unit
20.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2934	14 unit
21.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2914	14 unit
22.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2582	14 unit
23.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2375	14 unit
24.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2300	14 unit

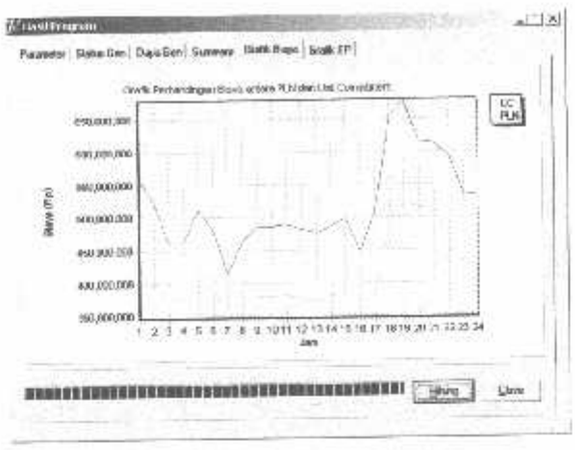
Keterangan : 1 = *ON* dan 0 = *OFF*



GAMBAR 4.12 DAYA GENERATOR PADA TANGGAL 30 JULI 2005



GAMBAR 4.13 HASIL PERHITUNGAN OPTIMASI PADA TANGGAL 30 JULI 2005

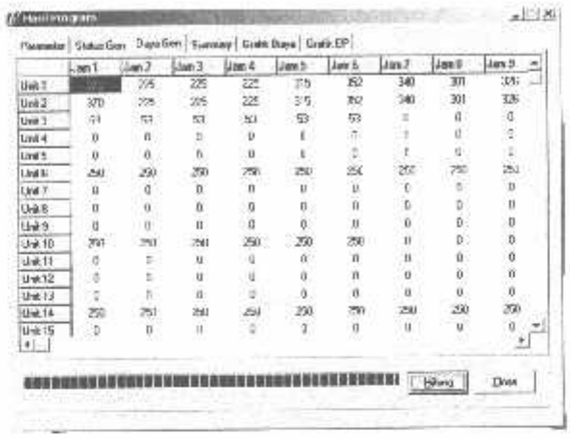


GAMBAR 4.14 KURVA HASIL PERHITUNGAN OPTIMASI PADA TANGGAL 30 JULI 2005

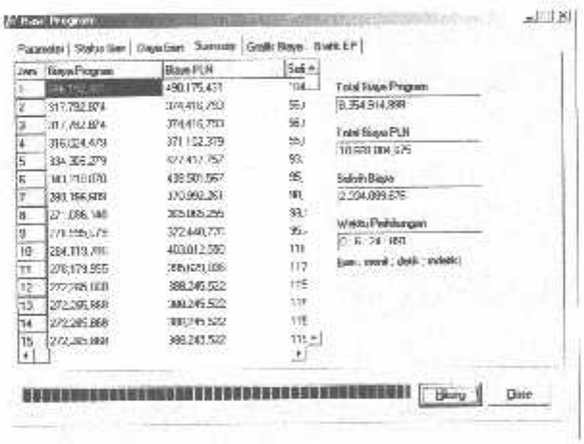
Tabel 4.10
Kombinasi Unit pembangkit Termal
Dengan Metode Kombinasi *Evolutionary Programming dan Tabu Search*
Pada Minggu, 31 Juli 2005

Jam	Status ON dan OFF Unit Pembangkit	Beban (MW)	Status ON Unit Pembangkit
01.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2275	9 unit
02.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1755	9 unit
03.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1755	9 unit
04.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1740	9 unit
05.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1895	8 unit
06.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1970	8 unit
07.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1642	6 unit
08.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1565	6 unit
09.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1615	6 unit
10.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1675	6 unit
11.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1625	6 unit
12.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1575	6 unit
13.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1575	6 unit
14.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1575	6 unit
15.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1575	6 unit
16.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1575	6 unit
17.00	1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	1689	6 unit
18.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0	2689	9 unit
19.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1	2929	14 unit
20.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1	2924	15 unit
21.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1	2904	15 unit
22.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1	2632	15 unit
23.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1	2330	15 unit
24.00	1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1	2215	15 unit

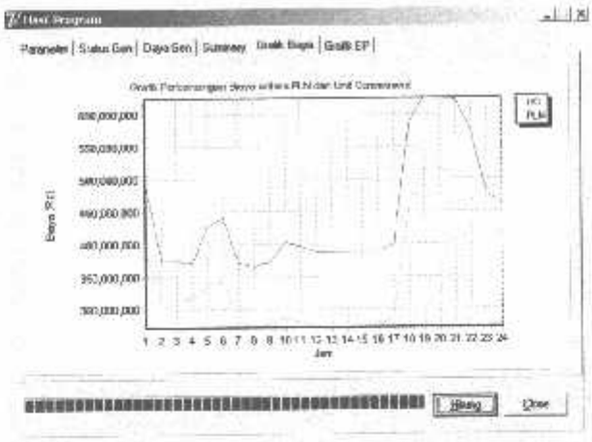
Keterangan : 1 = *ON* dan 0 = *OFF*



GAMBAR 4.15 DAYA GENERATOR PADA TANGGAL 31 JULI 2005



GAMBAR 4.16 HASIL PERHITUNGAN OPTIMASI PADA TANGGAL 31 JULI 2005



GAMBAR 4.17 GRAFIK HASIL PERHITUNGAN OPTIMASI PADA TANGGAL 31 JULI 2005

Setelah mendapatkan hasil yang paling optimal seperti pada penjelasan diatas, maka dilanjutkan dengan melakukan perhitungan total biaya operasi unit-unit pembangkit tiap periode jamnya dan perhitungan biaya operasi dalam system PT. Pembangkitan Jawa Bali sebagai perbandingan yang diperoleh dari hasil perhitungan fungsi objektif dengan memperhatikan data pembebanan harian pada PT Pembangkitan Jawa Bali. Hasil perhitungan biaya operasi untuk periode tiap jam dapat dilihat pada table sebagai berikut

Tabel 4.11
Perbandingan Biaya Operasional Per Jam PT. PJB
Dengan Metode *Kombinasi Evolutionary Programming dan Tabu Search*
Rabu, 27 Juli 2005

Jam	PT. PJB (Rupiah)	*EP-TS* (Rupiah)
1	464.644.480	411.654.705
2	445.152.537	395.648.048
3	429.890.280	385.550.774
4	429.890.280	385.550.774
5	454.392.306	403.736.550
6	450.703.396	400.559.145
7	418.029.154	373.766.877
8	457.160.966	405.027.578
9	544.211.424	446.637.373
10	551.612.015	454.042.139
11	551.612.015	454.042.139
12	490.355.878	417.162.795
13	543.875.979	460.597.925
14	543.875.979	469.400.441
15	543.875.979	469.400.352
16	529.091.873	455.172.972
17	589.448.318	458.079.945
18	695.328.855	529.631.143
19	701.267.068	538.618.356
20	701.267.068	538.618.315
21	686.841.778	534.163.160
22	617.098.701	491.680.279
23	550.541.913	470.528.851
24	545.855.172	467.567.038

Tabel 4.12
Perbandingan Biaya Operasional Per Jam PT. PJB
Dengan Metode *Kombinasi Evolutionary Programming dan Tabu Search*
Sabtu, 30 Juli 2005

Jam	PT. PJB (Rupiah)	*EP-TS* (Rupiah)
1	557.999.434	444.417.423
2	521.082.098	394.127.836
3	463.252.271	374.934.647
4	463.252.271	374.934.647
5	511.051.655	419.304.041
6	480.480.842	386.739.768
7	415.960.396	345.472.187
8	463.695.520	360.746.795
9	485.695.688	377.884.106
10	485.695.688	377.884.779
11	488.646.361	380.834.779
12	480.761.644	374.230.370
13	475.889.427	380.640.164
14	485.695.688	387.152.538
15	496.779.498	406.985.711
16	446.741.843	390.293.165
17	509.496.798	409.293.165
18	652.594.714	514.490.867
19	679.961.226	539.806.856
20	614.043.893	531.639.785
21	611.659.312	528.672.231
22	592.417.896	479.866.981
23	535.757.812	453.832.410
24	531.071.071	444.970.160

Tabel 4.13.
Perbandingan Biaya Operasional Per Jam PT. PJB
Dengan Metode *Kombinasi Evolutionary Programming dan Tabu Search*
Minggu, 31 Juli 2005

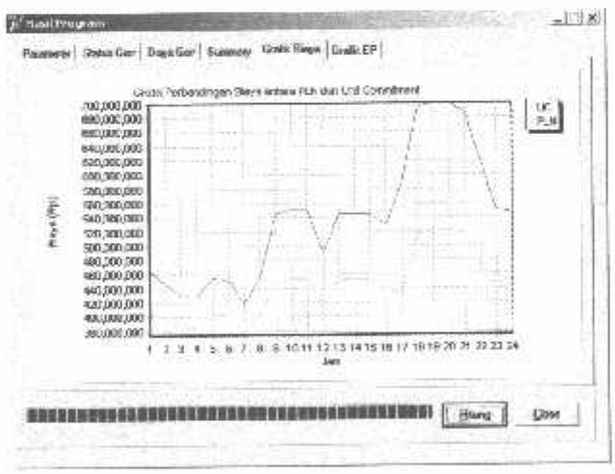
Jam	PT. PJB (Rupiah)	*EP-TS* (Rupiah)
1	490.175.431	408.305.310
2	374.418.793	340.850.633
3	374.418.793	340.850.633
4	371.102.379	340.850.633
5	427.417.757	340.142.789
6	438.501.567	349.036.703
7	370.992.261	280.196.609
8	365.065.255	271.086.140
9	372.448.770	276.995.079
10	403.012.550	284.119.786
11	395.629.036	278.179.955
12	388.245.522	272.265.868
13	388.245.522	272.265.868
14	388.245.522	272.265.868
15	388.245.522	272.265.868
16	388.245.522	272.265.868
17	398.447.105	285.787.551
18	585.092.495	460.283.930
19	625.067.195	521.833.890
20	624.322.126	538.844.414
21	621.344.129	535.880.415
22	572.596.233	495.970.538
23	478.673.520	458.720.234
24	549.055.664	445.267.629

Tabel 4.14
Perbandingan Total Biaya Operasional Per Jam PT. PJB
Dengan Metode *Kombinasi Evolutionary Programming dan Tabu Search*

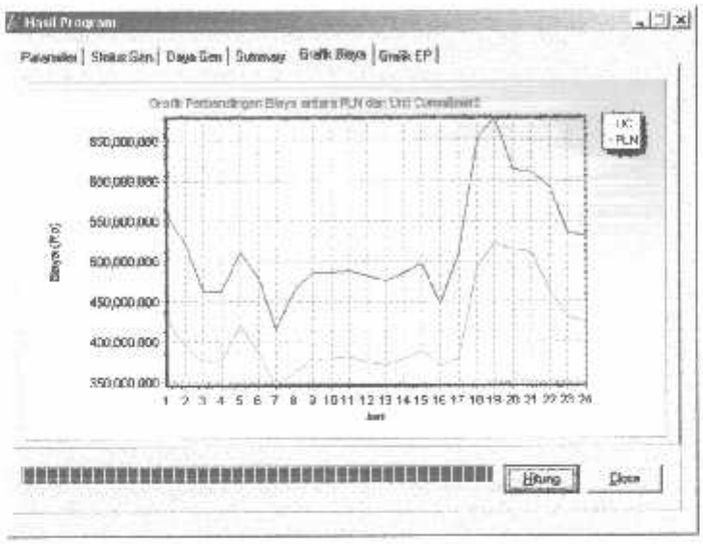
Periode Waktu (24 jam)	**Total Biaya PT. PJB (Rupiah)**	**Total Biaya *EP-TS* (Rupiah)**
Rabu, 27 Juli 2005	12.935.923.415	10.583.609.829
Sabtu, 30 Juli 2005	12.668.950.170	9.835.201.996
Minggu, 31 Juli 2005	10.689.004.675	8.354.914.999

Tabel 4.12 menunjukkan bahwa dengan menggunakan metode *Kombinasi Evolutionary Programming dan Tabu Search* terdapat pengurangan total biaya operasional pembangkitan dalam tiap periode 24 jam (1 hari). Hal ini berarti bahwa proses pemrograman menyebabkan pengurangan biaya operasi pembangkit yang cukup besar di banding dengan biaya operasi PT. PJB. Bila total biaya operasi setiap satu hari dihitung untuk kedua cara untuk Kombinasi *Evolutionary Programming dan Tabu Search* dan PT. PJB dapat dilihat pada grafik berikut

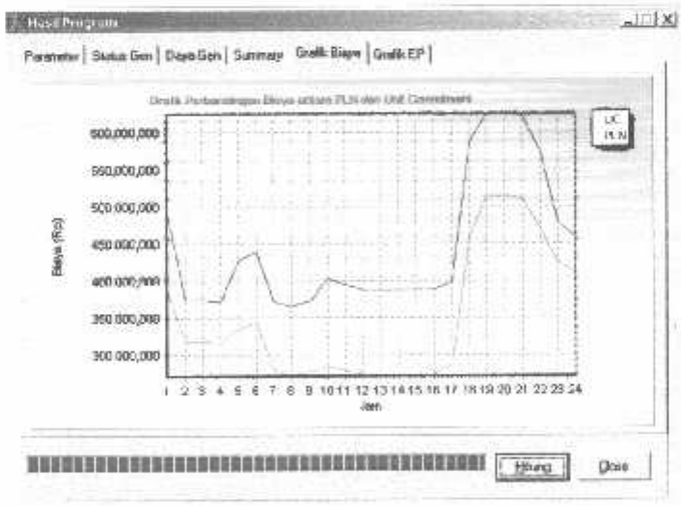
Grafik 4.1
Grafik Biaya PT. PJB dan Metode Kombinasi *Evolutionary Programming dan Tabu Search* pada Rabu, 27 Juli 2005



Grafik 4.2
Grafik Biaya PT. PJB dan Metode Kombinasi *Evolutionary Programming dan Tabu Search* pada Sabtu, 30 Juli 2005



Grafik 4.3
Grafik Biaya PT. PJB dan Metode Kombinasi *Evolutionary Programming dan Tabu Search* pada Minggu, 31 Juli 2005



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada analisa program komputer dengan menggunakan softwere Borland Delphi 8.0 yang dijalankan pada komputer AMD Athlon XP 2,4 GHz dengan *memory* 256 MHz dan hasil perhitungan terhadap penggunaan metode kombinasi *Evolutionary Programming* dan *Tabu Search* untuk menyelesaikan masalah komitmen unit atau penjadwalan unit-unit pembangkit termal pada PT. Pembangkitan Jawa Bali(PT. PJB) pada tanggal 27, 30, dan 31 Juli 2005, maka dapat diambil kesimpulan sebagai berikut:

1. Pada metode kombinasi *Evolutionary Programming* dan *Tabu Search* memberikan sebuah analisa perhitungan dan penyelesaian yang cukup efektif dan ekonomis dalam mengoptimalkan pembebanan dan penghematan total biaya operasi dibandingkan dengan total biaya operasional dari PT. Pembangkitan Jawa Bali(PT. PJB). Adapun selisih hasil perhitungan total biaya operasi antara metode kombinasi *Evolutionary Programming* dan *Tabu Search* dan PT. Pembangkitan Jawa Bali sebagai berikut:
   a. Pada tanggal 27 Juli 2005, total biaya operasional pada PT. Pembangkitan Jawa Bali (PT.PJB) sebesar Rp. 12.935.923.415,00 sedangkan total biaya hasil perhitungan operasional menggunakan metode kombinasi *Evolutionary Programming* dan *Tabu Search* sebesar Rp. 10.583.609.829,00.

Sehingga terjadi penghematan biaya operasional sebesar Rp. 2.352.313.585,00 atau sebesar 18%

b. Pada tanggal 30 Juli 2005, total biaya operasional pada PT. Pembangkitan Jawa Bali (PT.PJB) sebesar Rp. 12.668.950.170,00 sedangkan total biaya hasil perhitungan operasional menggunakan metode kombinasi *Evolutionary Programming* dan *Tabu Search* sebesar Rp. 9.835.201.996,00. Sehingga terjadi penghematan biaya operasional sebesar Rp. 2.614.481.049,00 atau sebesar 20%

c. Pada tanggal 31 Juli 2005, total biaya operasional pada PT. Pembangkitan Jawa Bali (PT.PJB) sebesar Rp. 10.689.004.675,00 sedangkan total biaya hasil perhitungan operasional menggunakan metode kombinasi *Evolutionary Programming* dan *Tabu Search* sebesar Rp.8.354.914.999,00. Sehingga terjadi penghematan biaya operasional sebesar Rp. 2.334.089.676,00 atau sebesar 21%

2. Proses eksekusi atau lamanya waktu perhitungan computer untuk metode kombinasi *Evolutionary Programming* dan *Tabu Search* adalah sebagai berikut:

   - Pada tanggal 27 Juli 2005 waktu eksekusi adalah ± 6 menit 56 detik 985 mdet
   - Pada tanggal 30 Juli 2005 waktu eksekusi adalah ± 6 menit 38 detik 984 mdet

- Pada tanggal 31 Juli 2005 waktu eksekusi adalah ± 6 menit 24 detik 891 mdet.

Dampak lamanya waktu perhitungan dari metode kombinasi *Evolutionary Programming* dan *Tabu Search* adalah berkurangnya waktu penentuan hasil penjadwalan dari unit-unit pembangkit pada PT. Pembangkitan Jawa Bali (PT. PJB) sehingga koordinasi antara unit-unit pembangkit termal pada PT. Pembangkitan Jawa Bali (PT.PJB) relatif lebih singkat.

3. Pada metode kombinasi *Evolutionary Programming* dan *Tabu Search* memberikan sebuah hasil yang optimal dengan waktu proses perhitungan yang relatif pendek yaitu kurang dari 7 menit sehingga metode tersebut dapat digunakan untuk melakukan penjadwalan operasi unit-unit pembangkit pada PT. PJB.

5.2 Saran

Berdasarkan kesimpulan diatas, dapat dibuat saran yang berhubungan dengan skripsi ini yaitu:

a. Metode kombinasi *Evolutionary Programming* dan *Tabu Search* dapat diterapkan pada PT. Pembangkitan Jawa Bali yang sudah terinterkoneksi Jawa sampai Bali, karena hasil perhitungan menunjukkan memberikan total biaya operasi pembangkitan yang lebih ekonomis atau lebih murah, kombinasi penjadwalan unit-unit pembangkit yang lebih efisien dengan lama waktu eksekusi yang singkat.

b. Untuk studi yang lebih luas, metode kombinasi *Evolutionary Programming* dan *Tabu Search* ini dapat dikembangkan dengan penambahan unit hidro artinya dapat dikembangkan untuk studi system penjadwalan hidrotermal

DAFTAR PUSTAKA

1. C. Christoper Asir Rajan, M. R. Mohan, " **an evolutionary programming based tabu search method for Solving Unit Commitment Problem**", IEEE Trans. On Power System, Vol. 19, No. 1, Februari 2004

2. Allen J. Wood dan W.F. Bruce. 1984. **Power Generation, Operation, and Control;** John Wiley and sons

3. Djiteng Marsudi, Ir, " **Operasi Sistem Tenaga Listrik**". Balai Penerbit dan Humas ISTN, 1990.

4. Abdul Kadir, " **Pembangkit Tenaga Listrik**", Universitas Indonesia(UI-Press), 1996.

5. Zuhal, " **Dasar Teknik Tenaga Listrik dan Elektronika Daya**", PT. Gramedia,Jakarta, 1990

6. Yusuf Ismail Nakhoda " **Operasi Sistem Tenaga Listrik**", Diktat kuliah Operasi Sistem Tenaga Listrik ITN Malang.

7. Hasegawa, E. Tanaka," **An Evolutionary Programming Solution to The Unit Commitment Problems**" IEEE Transaction On Power System, Vol 14, no. 4, pp 1452-1459, November 1999.

8. A.H. Mantawi et al," **A Unit Commitment By Tabu Search**" Inst. Elect. Eng. Gen. Transm. Dist., vol. 145, no. 1, pp 56-64.

9. D.B.Fogal,' **An introduction to simulated evolutionary optimization'**, IEEE Trans on Neural Network, Vol 5, No 1, pp. 3-14, January 1994

10. Yudhi eko prasetyo,"**Evaluasi komitmen unit pembangkit termal dengan menggunakan metode lagrange relaxation dan tabu search pada PT. Pembangkitan Jawa Bali**", skripsi, ITN Malang,2004

11. Firman hayu kristina putri,"**Komitmen unit pembangkit termal dengan menggunakan metode Evolutionary Programming pada PT.Pembangkitan Jawa Bali**", skripsi,ITN Malang,2004.

LAMPIRAN

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

BERITA ACARA UJIAN SKRIPSI

Nama : **ANTONI ANDI CHAROLI**

N.I.M : 99.12.143

Jurusan : TEKNIK ELEKTRO

Konsentrasi : ENERGI LISTRIK

Judul Skripsi : **ANALISIS KOMITMEN UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH* PADA PT. PEMBANGKITAN JAWA BALI**

Dipertahankan dihadapan majelis penguji jenjang strata satu (S-1),

Hari : Kamis

Tanggal : 6 Oktober 2005

Nilai : 78,25 (B^+)

Panitia Ujian

Ketua

Ir. Mochtar Asroni, MSME

Sekertaris

Ir. F. Yudi Limpraptono, MT

Anggota Penguji

Penguji I

Ir. Taufik Hidayat, MT

Penguji II

Irrine Budi S., ST, MT

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

LEMBAR BIMBINGAN SKRIPSI

Nama	: **ANTONI ANDI CHAROLI**
N.I.M	: 99.12.143
Jurusan	: TEKNIK ELEKTRO
Konsentrasi	: ENERGI LISTRIK
Judul Skripsi	:**ANALISA KOMITMEN UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH* PADA PT. PEMBANGKITAN JAWA BALI**
Tanggal Mengajukan	: 6 Agustus 2005
Tanggal Menyelesaikan	: 6 Februari 2006
Dosen Pembimbing	: **Ir. H. ALMIZAN ABDULLAH, MSEE**
Telah Dievaluasi Dengan Nilai	: 85 (Delapan Puluh Lima)

Mengetahui,
Ketua Jurusan Teknik Elektro

Ir.F Yudi Limpraptono, MT
NIP.103 950 0274

Diperiksa dan Disetujui,
Dosen Pembimbing

Ir.H.Almizan Abdullah, MSEE
NIP.130 900 0208

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

LEMBAR PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi jurusan Teknik Elektro jenjang strata satu (S-1), yang diselenggarakan pada:

Hari : Kamis
Tanggal : 6 Oktober 2005

Telah dilakukan perbaikan oleh:

Nama : **ANTONI ANDI CHAROLI**
N.I.M : 99.12.143
Jurusan : TEKNIK ELEKTRO
Konsentrasi : ENERGI LISTRIK
Judul Skripsi : **ANALISIS KOMITMEN UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH* PADA PT. PEMBANGKITAN JAWA BALI**

Perbaikan meliputi:

No.	Materi Perbaikan	Keterangan
1.	Tujuan lebih dispesifikkan	
2.	Tambahkan tanda kutipan pada Bab II	
3.	Bagaimana menentukan nilai populasi dalam program	
4.	Tambahkan pada bab IV tabel pembebanan setiap unit sebelum dan sesudah optimasi	
5.	Penulisan gambar dengan fontnya	

Anggota Penguji

Penguji I

Ir. Taufik Hidayat, MT

Penguji II

Irrine Budi S., ST, MT

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

LEMBAR PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi jurusan Teknik Elektro jenjang strata satu (S-1), yang diselenggarakan pada:

Hari : Kamis
Tanggal : 6 Oktober 2005

Telah dilakukan perbaikan oleh:

Nama : **ANTONI ANDI CHAROLI**

N.I.M : 99.12.143

Jurusan : TEKNIK ELEKTRO

Konsentrasi : ENERGI LISTRIK

Judul Skripsi : **ANALISIS KOMITMEN UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH* PADA PT. PEMBANGKITAN JAWA BALI**

Perbaikan meliputi:

No.	Materi Perbaikan	Keterangan
1.	Tujuan lebih dispesifikkan	
2.	Tambahkan tanda kutipan pada Bab II	
3.	Bagaimana menentukan nilai populasi dalam program	
4.	Tambahkan pada bab IV tabel pembebanan setiap unit sebelum dan sesudah optimasi	
5.	Penulisan gambar dengan fontnya	

Diperiksa dan Disetujui,
Dosen Pembimbing

Ir. H. Almizan Abdullah, MSEE

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : ANTONI ANDI CHAROLI
Nim : 99.12.143
Masa Bimbingan : 06 AGUSTUS 2005 s/d 06 FEBRUARI 2006
Judul Skripsi : **ANALISA KOMITMEN UNIT PEMBANGKIT TERMAL DENGAN METODE KOMBINASI *EVOLUTIONARY PROGRAMMING* DAN *TABU SEARCH* PADA PT. PJB**

No.	Tanggal	Uraian	Parap Pembimbing
1.	07-09-05	Bab IV : 1. Pada Algoritma Program, seyogyanya urutan ditulis lengkap 2. Perlu ditambahkan Sub. flowchart a.l. perhitungan penjadwalan	[paraf]
2.		3. Pada Uji Validasi perlu diperbandingkan hasil beban, dan biaya menurut program dan biaya menurut PJB 4. Jelaskan cara menentukan besar Parameter²	[paraf]
3.	12-09-05	Bab IV : 1. Nomor langkah-langkah pada Algoritma sebaiknya diurutkan. 2. Cantumkan jenis komputer yg dipakai.	[paraf]
4.		Bab II & III : OK.	
5.	13-09-05	Bab IV : Perbaiki Pendahuluan	[paraf]
6.	15-09-05	Bab IV & V : OK Siap ujian Komprehensif	[paraf]
7.	17-09-05	Bab IV : Tampilkan hasil perhitungan penjadwalan unit pembangkit menurut program secara lengkap (24 jam) untuk dibandingkan dg hasil jurnal	[paraf]
8.		Bab V : 1. Jangan pakai tabel tapi jelaskan poin² nya saja 2. Jelaskan lamanya waktu perhitungan	[paraf]
9.			
10.			

Malang, 17-9-2005
Dosen Pembimbing,

[tanda tangan]

Ir. H. ALMIZAN ABDULLAH, MSEE
Nip. P. 103 900 0208

Form.S-4b

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 (MW)	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
1	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	217983.548	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	17117480.3	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	352707.2	350680.77	803.669
	PLTG GILITIMUP 1-2 (HSD)	2 x 20	5	16	3	1	1	0	6.33	0	687181.85	683240.965	1762.3693
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	11560315	53685.135	400.845
	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	21011735	87825.15	57.33
	MTW GT 1/2 - OC (HSD)	2 x 140	72	138	36	10	0	0	0	0	14705521.25	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	333208.82	11.64715
	MTW CC - 3.3.1 (HSD)	640	315	605	36	10	3	2	160.1	96.42	43043339	286609.995	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	185	48	10	11	2	215.24	89.29	2949167.5	205217.145	83.79

Catatan

Harga Batubara	253	Rp/Kg
Harga MFO	1595.5	Rp/liter
Harga HSD	1595.5	Rp/liter
Harga Gas UP. Gresik	2.53	US$/MMBTU
Harga Gas UP. M.Karang	2.45	US$/MMBTU
Nilai Tukar	9000	Rp/US$

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL : RABU, 27 JULI 2005

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—GT 1.1	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95
	MKRNG10C1	95	95	95	95	95	95	95	95	95	95	95	95	95	90	95	95	95	95	95	95	95	95	95	95	95
	MKRNG20C1	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	95	95	95	95	95	95	95	63	63
	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	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	MKRNG #1	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	80	85	85	85	85	85	85	60	60
	#2	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	80	85	85	85	85	85	85	60	60
	#3	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	80	85	85	85	85	85	85	60	60
PLTU	MKRNG #4	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
	#5	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	130	130	130	130	130	130	130	130
PLTGU	MTWAR1OC —GT 1.1	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
	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
	MTWAR1OC	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
	MTWAR1JOC	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	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	125	125	100	100	325	315	315	315	315	300	450	390	350	310	400	425	200	550	550	550	550	550	550	550	550
	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																									
PLTG	MTWAR #3.1	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
	#3.2	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
	#3.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	80	80	80	50	50	40	40
PLTG	MTWAR #4.1	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
	#4.2	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
	#4.3	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
	PEMBELIAN DARI LUAR PLN																									
PLTP	GSLAK #4-#6	183	183	183	183	182	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183
	CIKARANG	110	110	110	110	110	110	110	110	110	110	110	110	75	75	75	120	120	120	120	120	120	120	90	90	80
PLTU	Krakatau 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		5005	5005	3845	5000	4954	4982	4832	4852	4910	5030	5190	5050	4786	4647	4894	5174	5341	5601	5573	5577	5517	5547	5214	5325	5406
(**) Beban Area-1		4934	4958	4931	4870	4602	4698	4621	4655	4620	4719	4680	4840	4743	4677	4869	5268	5312	5948	5614	5706	5739	5703	5687	5466	5436
Surplus (*) - (**)		81	107	109	110	333	287	211	196	295	311	310	184	39	-33	23	-133	35	1	-37	-129	-162	-156	-173	-141	-178
Cadangan Sekatika		72	72	71	71	65	63	118	117	64	70	78	70	119	117	126	74	81	45	27	27	27	25	25	58	58
Cadangan Luar		462	462	487	487	503	632	648	675	612	497	331	497	710	350	500	417	190	140	100	100	100	130	130	322	322

RENCANA : **HARI/TANGGAL : RABU , 27 JULI 2005**

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	MKRNG1OC – GT 1.1		95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95
	MKRNG1OC1		95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95
	MKRNG2OC1		63	95	95	95	63	63	63	63	63	63	95	95	95	95	95	75	63	63	63	63	63	63	63	73
	MKRNG2OC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3OC		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		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	MKRNG	#1	80	80	80	80	80	80	80	80	80	80	85	85	85	85	85	80	80	80	80	80	80	80	80	70
		#2	80	80	80	80	80	80	80	80	80	80	85	85	85	85	85	80	80	80	80	80	80	80	80	70
		#3	80	80	80	80	80	80	80	80	80	80	85	85	85	85	85	80	80	80	80	80	80	80	80	70
PLTU	MKRNG	#4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#5	130	130	130	130	130	130	130	130	130	130	130	130	130	130	130	130	130	120	120	120	120	120	120	125
PLTGU	MTWAR1.1OC – GT 1.1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR1OC1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2OC1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR12OC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR13OC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR1CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR3CC		550	560	550	550	500	550	550	500	550	550	560	560	540	550	560	550	560	550	500	475	450	450	425	[illegible]
	MTWAR	GT 2.1	0	0	0	0	0	0	0	65	80	80	130	135	135	135	135	135	120	120	65	0	0	0	0	28
	MTWAR	GT 2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	65	65	65	65	5
PLTG	MTWAR	#3.1	0	0	0	0	0	0	0	40	50	60	120	120	140	140	140	100	80	80	50	50	50	50	50	28
		#3.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#3.3	40	120	80	60	40	40	40	40	50	60	120	120	140	120	120	100	80	80	50	50	0	0	0	42
PLTG	MTWAR	#4.1	0	0	0	0	0	0	0	40	50	60	120	120	140	140	110	100	80	80	50	50	0	0	0	24
		#4.2	0	0	0	0	0	0	0	0	50	60	120	120	140	140	140	100	80	80	50	50	50	50	0	26
		#4.3	0	0	0	0	0	0	0	40	50	60	120	130	140	140	140	100	80	80	50	50	50	50	50	28
			PEMBELIAN DARI LUAR PLN																							
PLTP	GoLAK	#4-#6	163	163	183	183	183	183	163	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	181
	CIKARANG		120	120	120	120	120	120	120	120	120	135	135	135	135	135	135	135	135	135	135	110	110	110	110	115
PLTU	Krakatau 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			[illegible]	[illegible]	5501	[illegible]	5435	[illegible]	[illegible]	5528	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	5933	[illegible]	5574	[illegible]	[illegible]	[illegible]	[illegible]
(**) Beban Area-1			5671	5855	5739	5766	5755	5677	5618	5597	5593	5646	5797	5727	5754	5756	5750	5604	5732	5627	5412	5360	5237	5192	4955	[illegible]
Selisih (*) - (**)			-236	-298	-297	-282	-320	-234	-220	69	85	107	430	548	601	589	695	411	295	306	262	214	212	257	418	102
Cadangan Sekejap			37	36	36	35	37	37	61	94	74	77	78	53	27	29	[illegible]	62	62	61	76	97	91	91	87	84
Cadangan Putar			232	115	165	135	233	737	262	707	692	642	100	120	30	[illegible]	50	290	417	462	717	722	831	[illegible]	572	393

SUB SISTEM REGION_4

RENCANA : HARI/TANGGAL : RABU , 27 JULI 2005

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.	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
PLTA	SUTAMI	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
PLTA	BRANTAS	13	13	13	13	13	13	13	13	13	13	13	13	10	10	10	10	10	10	10	10	10	10	10	10	10
PLTU	PITON #1	360	360	360	360	360	360	360	360	360	360	360	360	310	310	310	360	360	360	360	360	360	360	360	310	310
	PITON #2	360	360	360	360	360	360	360	360	360	360	360	360	310	310	310	360	360	360	360	360	360	360	360	310	310
PLTGU 1	GRSIK11OC	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
	- GRSIK12OC	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
	- GRSIK13OC	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
	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	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	300	300	275	250	250	250	250	250	250	275	300	250	250	250	250	250	275	300	300	325	325	325	325	275	275
PLTGU 2	GRSIK21OC	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
	- GRSIK22OC	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
	- GRSIK23OC	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
	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	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164
	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	GRSIK31OC	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
	- GRSIK32OC	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
	- GRSIK33OC	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	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	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	300	300	300	250	250	250	250	250	250	300	300	300	300	250	300	300	300	300	300	325	325	325	325	300	300
PLTU	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
PLTU	GRSIK #3	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
	#4	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
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
	GRESIK #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
(*) Pembangkitan Area 4		4648	4613	4545	4513	4483	4427	4423	4427	4473	4536	4604	4493	4140	4070	4240	4490	4515	4508	4505	4840	4660	4600	4640	4415	4485
(**) Beban Area-4		2542	2458	2468	2400	2388	2375	2334	2445	2395	2535	2549	2340	2165	2135	2264	2376	2474	2499	2498	2545	2558	2562	2503	2387	2427
Selisih (*) (**)		2101	2155	2075	2043	2035	2047	2089	1977	2037	1983	2053	2147	1984	1937	1972	2113	2040	2065	2066	2083	2092	2113	2137	2033	1988
Cadangan Sekretika		87	65	82	57	56	56	56	56	56	80	85	59	33	87	80	59	80	83	83	80	81	50	50	91	91
Cadangan Putaran		370	400	420	520	510	540	540	540	540	445	460	470	820	890	720	470	445	395	395	320	310	320	320	545	515

SUB SISTEM REGION_4

RENCANA : HARI/TANGGAL : RABU , 27 JULI 2005

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.	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
PLTA	SUTAMI	25	25	25	25	25	25	25	50	50	90	90	90	90	90	90	90	90	50	50	25	25	25	25	38
PLTA	BRANTAS	10	10	10	10	10	10	10	13	13	20	20	20	20	20	20	17	17	13	13	10	10	10	10	12
PLTU	PITON #1	360	360	360	360	360	360	360	310	310	360	370	370	370	370	370	370	370	370	370	360	360	360	360	355
	PITON #2	360	360	360	360	360	360	360	310	310	360	370	370	370	370	370	370	370	370	370	360	360	360	360	355
PLTGU 1	GRSIK11OC = C_{11}	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
	GRSIK12OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK13OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	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	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	325	325	325	325	325	325	250	250	250	250	325	350	350	350	350	350	350	325	275	275	275	275	275	292
PLTGU 2	GRSIK21OC = C_{21}	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
	GRSIK22OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	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	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164	164
	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	GRSIK31OC = C_{31}	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
	GRSIK32OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK33OC	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	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	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	325	325	325	325	325	325	300	250	250	275	325	325	325	325	325	325	325	325	325	275	275	275	275	298
PLTU	GRSIK #1	50	50	50	50	50	50	50	50	50	50	80	80	80	80	80	80	80	50	50	50	50	50	50	54
	#2	50	50	50	50	50	50	50	50	50	50	80	80	80	80	80	80	80	50	50	50	50	50	50	54
PLTL	GRSIK #3	90	90	90	90	90	90	90	90	90	135	167	167	167	167	167	167	167	167	140	100	90	90	90	105
	#4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTG	GLIMR #1	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
	#2	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
	GRESIK #1	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
	#2	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
(*) Pembangkitan Area 4		4640	4680	4640	4650	4640	4605	4545	3473	4582	4821	5248	5273	5273	5273	5219	5220	5170	5101	5012	4829	4710	4715	4545	4657
(**) Beban Area 4		2574	2600	2593	2585	2509	2499	2471	4540	2756	3244	3533	3515	3541	3468	3444	3355	3257	3102	2934	2759	2711	2655	2650	2902
Selisih (*) - (**)		2066	2079	2046	2073	2131	2105	2073	1933	1826	1846	1724	1758	1731	1805	1830	1915	2048	1999	2077	2070	1999	2015	1895	2004
Cadangan Seketika		50	38	50	50	50	48	81	84	96	69	35	19	19	19	19	19	19	38	43	54	62	68	61	53
Cadangan Putar		320	280	320	320	320	370	529	650	602	591	234	209	209	209	209	209	209	334	411	460	490	515	490	410

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL : SABTU , 30 JULI 2005

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	MKRNG1OC – (1)		95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95
	MKRNG1OC1		95	95	95	95	95	95	95	95	95	95	95	95	90	95	95	85	95	95	95	95	95	95	95	95	95
	MKRNG2OC1		65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	95	95	95	95	95	95	95	95	65	65
	– MKRNG2OC		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
	- MKRNG3OC		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		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	MKRNG	#1	80	80	80	80	80	80	80	80	80	80	80	80	60	60	85	85	85	85	85	85	85	85	85	85	85
		#2	80	80	80	80	80	80	80	80	80	80	80	80	60	60	85	85	85	85	85	85	85	85	85	85	85
		#3	80	80	80	80	80	80	80	80	80	80	80	80	60	60	85	85	85	85	85	85	85	85	85	85	85
PLTU	MKRNG	#4	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
		#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
PLTGU	MTWAR1OC – (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
	MTWAR1OC1		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
	MTWAR2OC1		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
	- MTWAR1OC		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
	- MTWAR3OC		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
	MTWAR1CC		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
	MTWAR2CC		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
	MTWAR3CC		425	425	425	350	350	325	325	325	350	425	500	425	425	425	475	475	525	550	550	550	550	550	550	525	500
	MTWAR	GT 2.1	60	60	60	60	60	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
			PEMBANGKITAN MUARA KARANG																								
PLTG	MTWAR	#3.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	60	100	100	140	140	140	100	50
		#3.2	50	50	50	50	50	50	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#3.3	50	50	50	50	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTG	MTWAR	#4.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	80	100	140	140	140	100	60
		#4.2	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
		#4.3	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
			PEMBELIAN DARI LUAR PLN																								
PLTP	GSLAK	#4+#5	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183
	CIKARANG		110	110	110	110	110	110	110	110	110	110	110	110	75	75	75	120	120	120	120	120	120	120	120	90	90
PLTU	Krakatau 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			4505	4870	4645	4750	4735	4590	4595	4545	4550	4765	4890	4710	4515	4605	4655	4910	5070	5140	5225	5265	5320	5370	5105	5140
(**) Beban Area-1			4938	4867	4750	4700	4662	4573	4532	4504	4554	4651	4868	4939	4613	4477	4619	4735	5011	5053	5159	5196	5244	5343	5311	5108	5095
Selisih (*) - (**)			27	3	65	50	73	22	33	41	36	71	-79	-124	2	193	276	235	159	87	69	49	81	-19	2	52	15
Cadangan Seketika			92	97	85	81	85	73	73	71	73	82	98	79	81	78	27	24	29	9	9	10	9	0	0	40	34
Cadangan Putar			500	595	640	710	730	595	595	505	460	285	160	335	400	345	320	90	130	130	115	95	15	15	15	150	255

(*) Termasuk Beban Pompa

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL : SABTU , 30 JULI 2005

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	MKRNG1OC		95	95	95	85	95	95	95	95	95	95	95	95	95	95	95	95	86	95	95	96	95	95	95	95
	MKRNG1OC1		95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95
	MKRNG2OC1		65	65	65	65	65	65	65	65	65	65	95	95	95	95	95	95	65	65	65	65	65	65	65	74
	MKRNG2OC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3OC		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		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	MKRNG	#1	85	85	85	85	85	60	60	60	60	60	85	85	85	85	80	80	80	60	80	80	80	80	80	79
		#2	85	85	85	85	85	60	60	60	60	60	85	85	85	85	80	80	80	60	80	80	80	80	80	79
		#3	85	85	85	85	85	60	60	60	60	60	85	85	85	85	80	80	80	80	80	80	80	60	80	79
PLTU	MKRNG	#4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#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
PLTGU	MTWAR1OC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR1OC1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2OC1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2OC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR3OC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR1CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2CC		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR3OC1		500	550	550	550	550	400	490	420	420	100	521	550	550	550	550	550	550	550	525	500	450	425	425	480
	MTWAR	CT 2.1	0	0	0	0	0	0	0	65	80	80	100	120	120	120	100	100	100	100	80	80	80	90	80	33
	MTWAR	CT 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
PLTG	MTWAR	#3.1	60	140	130	80	90	40	40	40	40	40	60	130	130	120	80	60	60	60	80	60	60	60	60	54
		#3.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
		#3.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
PLTG	MTWAR	#4.1	60	140	130	80	80	40	40	40	40	40	60	130	120	110	90	60	60	60	60	60	60	0	0	49
		#4.2	0	0	0	0	0	0	0	40	40	40	60	130	130	110	80	60	60	60	60	60	0	0	0	19
		#4.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMBELIAN DARI LUAR PLN																									
PLTP	GSLAK	#4-#6	183	183	183	183	183	183	183	183	163	183	183	183	183	183	183	183	183	103	183	183	183	183	183	183
	CIKARANG		120	120	120	120	120	120	120	120	120	120	135	135	135	135	135	135	135	135	135	110	110	110	110	115
PLTU	Krakatau 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			5079	5289	5115	5029	5019	4814	4914	4700	4714	4604	5324	5570	5579	5529	5394	5334	5254	5294	5164	5105	4995	4835	4835	5017
(**) Beban Area-1			5072	5194	5094	5024	5107	4953	5001	4891	4875	5040	5488	5688	5639	5583	5511	5483	5392	5224	5084	5010	4884	4729	4685	5004
Selisih (*) - (**)			7	95	21	5	-88	-139	-87	-192	-161	-141	-142	-108	-60	-54	-117	-149	-138	70	71	95	111	106	150	5
Cadangan Seketika			34	3	59	54	54	83	83	83	82	87	43	26	26	23	29	28	52	52	100	98	93	95	95	55
Cadangan Putar			255	45	315	315	315	530	530	820	825	725	315	60	60	110	245	305	385	385	480	505	475	475	475	353

SUB SISTEM REGION_4

RENCANA : HARI/TANGGAL : SABTU , 30 JULI 2005

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	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
PLTA	BRANTAS	11	11	11	11	11	11	11	11	11	11	11	11													
PLTU	PITON #1	360	360	360	360	360	360	360	360	360	360	360	350	350	300	300	300	300	300	300	300	300	300	300	300	300
	PITON #2	360	360	360	360	360	360	360	360	360	360	360	300	250	250	250	250	250	250	250	250	250	250	250	250	250
PLTGU 1	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
	GRSIK1OC1	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
	GRSIK2OC1	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	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
	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	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	400	400	350	325	300	300	300	300	350	400	400	325	250	250	250	250	250	250	250	250	250	270	250	250	250
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
	GRSIK1OC2	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
	GRSIK2OC2	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	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
	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	250	250	225	185	185	164	164	164	184	184	224	164	164	164	164	164	164	164	164	164	164	164	164	164	164
	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
	GRSIK1OC3	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
	GRSIK3OC3	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
	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
	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	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	350	350	325	300	300	300	300	300	350	400	400	325	250	250	250	250	250	300	300	300	300	300	300	300	300
PLTU	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
PLTU	GRSIK #3	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
	#4	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
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
	GRESIK #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
(*) Pembangkitan Area-4		4803	4803	4803	4717	4686	4647	4687	4687	4787	4857	4852	4746	3650	3650	3598	3599	3698	3848	3848	3848	3813	3909	3998	3790	3790
(**) Beban Area-4		2507	2503	2400	2199	2407	2259	2391	2412	2438	2510	2740	2240	2028	2003	2106	2208	2259	2344	2344	2367	2382	2387	2395	2327	2238
Selisih (*) - (**)		2296	2406	2333	2314	2280	2388	2276	2264	2349	2349	2512	2302	1945	1845	1492	1390	1439	1506	1505	1482	1496	1522	1534	1471	1518
Cadangan Sekatika		80	80	75	75	69	68	68	68	74	79	82	100	87	83	83	83	83	86	86	86	86	89	88	80	86
Cadangan Putar		495	495	595	605	710	721	731	731	611	511	446	856	1231	1601	1801	1601	1501	1351	1351	1351	1351	1291	1311	1401	1401

SUB SISTEM REGION_4

RENCANA : HARI/TANGGAL : SABTU , 30 JULI 2005

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	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	25	25	25	25	25	25	25	50	50	90	90	90	90	90	90	90	90	50	50	25	25	25	25	38
PLTA	BRANTAS						14	14	14	14	14	22	22	22	22	22	19	19	18	18	14	14	14	14	12
PLTU	PITON #1	300	300	300	300	300	300	300	300	300	310	360	370	370	370	370	360	360	360	360	360	360	360	360	333
	PITON #2	250	250	250	250	250	250	250	250	300	310	360	370	370	370	370	360	360	360	360	360	360	360	360	309
PLTGU 1	GRSIK11OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1OC1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2OC1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK12OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK13OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	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	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	250	250	250	300	325	325	325	325	300	325	400	425	425	425	425	425	425	400	326	300	300	300	300	318
PLTGU 2	GRSIK21OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1OC2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2OC2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK22OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK23OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	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	164	164	164	164	164	164	164	164	164	164	250	250	250	225	200	164	164	164	164	164	164	164	164	179
	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	GRSIK31OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1OC3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2OC3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK32OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK33OC	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	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	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	300	300	300	300	300	300	300	300	300	300	375	425	425	425	425	425	425	400	325	300	300	300	300	323
PLTU	GRSIK #1	50	50	50	50	50	50	50	50	50	50	90	90	90	80	80	80	80	80	50	50	50	50	50	56
	#2	50	50	50	50	50	50	50	50	50	50	90	90	90	90	80	80	80	80	50	50	50	50	50	56
PLTU	GRSIK #3	90	90	90	90	90	90	90	90	90	135	167	167	167	167	167	167	167	140	90	90	90	90	90	105
	#4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTG	GLTMR #1	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
	#2	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
	GRESIK #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
(*) Perhangkitan Area-4		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4030	4234	[illegible]	[illegible]	5570	5525	[illegible]	[illegible]	[illegible]	5403	5255	[illegible]	[illegible]	[illegible]	[illegible]	4551	[illegible]
(**) Beban Area-4		2241	2393	2250	2365	[illegible]	2222	2268	2321	2423	2766	[illegible]	3251	3262	3241	3246	3196	3035	2922	2787	2664	2503	2405	2462	[illegible]
Selisih (*) - (**)		1508	1456	1488	1573	1633	1753	1757	1709	1811	1878	2286	2319	2308	2285	2254	2265	2386	2333	2358	2327	2106	2087	2093	1650
Cadangan Seketika		86	86	86	88	89	91	94	94	98	107	59	21	21	27	26	23	22	82	16	71	67	62	62	74
Cadangan Putaran		1351	1351	1351	1301	1276	1315	1300	1419	1430	1085	278	157	177	222	247	323	323	448	245	810	839	721	721	833

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL : MINGGU , 31 JULI 2005

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	MKRNG1OC + GT 1		95	95	95	95	95	95	95	80	80	95	95	95	95	80	95	95	95	95	95	95	95	95	95	95	95
	MKRNG1OC1		95	95	95	95	95	95	95	80	80	95	95	95	95	80	95	95	95	95	95	95	95	95	95	95	95
	MKRNG2OC1		65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65
	MKRNG2OC		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
	MKRNG3OC		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		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	MKRNG	#1	80	80	60	50	50	50	50	50	50	80	80	80	60	60	60	60	60	60	80	80	80	80	80	80	80
		#2	60	60	50	50	60	50	50	50	50	80	80	80	60	60	60	60	60	60	80	80	80	80	80	80	80
		#3	80	80	60	50	50	60	50	50	50	80	80	80	60	80	60	60	60	60	80	80	80	80	80	80	80
PLTU	MKRNG	#4	50	50	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		#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
PLTGU	MTWAR1OC + GT 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
	MTWAR1OC1		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
	MTWAR2OC1		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
	MTWAR1OC		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
	MTWAR3OC		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
	MTWAR1CC		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
	MTWAR2CC		0	0	0	300	300	300	300	300	300	300	300	300	300	300	300	300	300	300	300	300	300	330	330	320	300
	MTWAR3CC		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
			PEMBANGKITAN MUARA TAWAR																								
PLTG	MTWAR	#3.1	50	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		#3.2	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
		#3.3	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
PLTG	MTWAR	#4.1	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
		#4.2	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
		#4.3	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
			PEMBELIAN DARI LUAR PLN																								
PLTP	GSLAK	#4-#6	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183
	CIKARANG		75	75	75	75	75	75	75	75	75	75	75	75	50	50	50	50	50	50	50	50	50	50	50	50	50
PLTU	Krakatau 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			1900	1666	4475	5495	5495	4495	4495	4475	4555	4556	4655	5333	5105	5035	4933	3993	3993	4093	4205	4204	4493	4545	4541	4505	4457
(**) Beban Area-1			4810	4585	4480	4422	4389	4384	4367	4307	4204	4334	4572	4405	4238	4106	4022	4011	4151	4267	4357	4329	4388	4109	4358	4376	4403
Selisih (*) - (**)			190	80	95	73	106	111	168	168	181	240	33	-72	-180	-103	11	-18	-139	-174	-154	-125	-129	-126	-115	-143	-46
Cadangan Sekunder			83	76	69	65	65	65	65	73	69	69	70	170	174	184	184	184	188	176	175	178	177	117	176	174	
Cadangan Putar			590	590	540	445	445	445	445	475	555	358	335	607	807	912	892	922	922	822	712	712	552	672	672	682	722

SUB SISTEM REGION_1

RENCANA : HARI/TANGGAL : MINGGU , 31 JULI 2005

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	MKRNG1OC - GT	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	94
	MKRNG1OC1	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	95	94
	MKRNG2OC1	65	65	65	65	65	65	65	65	65	65	65	95	95	95	95	95	95	65	65	65	65	65	65	69
	MKRNG2OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MKRNG3OC	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	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	MKRNG #1	80	80	80	80	80	80	80	60	60	60	80	80	80	80	80	80	80	80	80	80	60	60	60	71
	#2	80	80	80	80	80	80	80	60	60	60	80	80	80	80	80	80	80	80	80	80	60	60	60	71
	#3	80	80	80	80	80	80	80	60	60	60	80	80	80	80	80	80	80	80	80	80	60	60	60	71
PLTU	MKRNG #4	100	100	100	100	100	100	100	100	100	100	120	120	120	120	120	120	120	120	120	120	120	100	100	103
	#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
PLTGU	MTWAR1OC #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
	MTWAR1OC1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2OC1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2OC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR3OC	0	0	0	0	0	0	72	72	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
	MTWAR1CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	MTWAR2CC	325	325	325	325	325	325	250	250	250	0	0	0	0	0	0	0	0	0	0	0	0	0	0	196
	MTWAR3CC	0	0	0	0	0	0	0	0	0	350	445	550	550	550	550	550	550	550	500	500	400	350	350	163
	MTWAR GT2.1	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 GT2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTG	MTWAT #3.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#3.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#3.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTG	MTWAR #4.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#4.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	#4.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PEMBELIAN DARI LUAR PLN																								
PLTP	GSLAK #4.5	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183
	CIKARANG	50	50	50	50	50	50	50	50	75	120	120	120	120	120	120	120	120	120	120	120	120	100	100	95
PLTU	Krakatau Steel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(*) Perkiraan Beban-1		4161	4752	4492	4232	4232	4242	4216	4356	4624	4977	5301	5109	5159	5106	5159	5159	5122	4975	4825	4840	4705	4795	4660	4761
(**) Beban Aktual-1		4241	4856	4595	4299	3959	4163	4208	4317	4503	4710	5156	5274	5499	5319	5307	5361	5261	5110	5002	4864	4720	4676	4553	4622
Selisih: (*) - (**)		-76	-104	-103	-67	273	79	11	-158	-316	-339	-152	-165	-340	-160	-148	-206	-152	-31	-87	-124	-14	29	52	-46
Cadangan Sekettika		173	178	178	111	177	177	175	172	172	180	37	37	12	12	12	12	35	38	82	39	71	63	89	113
Cadangan + Outage		342	657	897	677	677	677	838	698	844	792	185	80	30	30	30	30	80	110	230	345	490	440	540	546

SUB SISTEM REGION_4

RENCANA : HARI/TANGGAL : MINGGU, 31 JULI 2005

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	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	25	25	25	25	25	25	25	50	50	90	90	90	90	90	90	90	90	50	50	25	25	25	25	38
PLTA	BRANTAS	7	7	7		7	14	14	14	14	22	22	22	22	22	22	19	19	18	18	14	14	14	14	12
PLTU	PITON #1	225	225	225	225	225	225	225	250	296	305	350	370	370	370	370	370	370	360	360	360	360	350	350	304
	PITON #2	225	225	225	225	225	225	225	250	298	305	360	370	370	370	370	370	370	360	360	360	360	350	350	304
PLTGU 1	GRSIK1100 - GT 1.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1300	250	250	250	250	250	250	250	250	250	300	375	425	425	425	420	415	400	400	350	350	325	300	300	300
PLTGU 2	GRSIK2100 - 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
	GRSIK1002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2200	164	164	164	164	164	164	164	164	164	164	185	225	250	250	225	225	164	164	164	164	164	164	164	172
	GRSIK2300	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	GRSIK3100 - GT 3.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK1003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK2003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK3200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK3300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK3100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK3200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	GRSIK3300	250	250	250	250	250	250	250	250	250	300	375	400	400	400	400	400	400	400	325	325	300	300	300	291
PLTU	GRSIK #1	50	50	50	50	50	50	50	50	50	50	80	80	80	80	80	80	80	80	50	50	50	50	50	55
	#2	50	50	50	50	50	50	50	50	50	50	80	80	80	80	80	80	80	80	50	50	50	50	50	55
PLTU	GRSIK #3	90	90	90	90	90	90	90	90	90	135	167	167	167	167	167	167	167	167	130	140	90	90	90	106
	#4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PLTG	GLTMR #1	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
	#2	0	0	0	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	4
	GRESIK #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
(*) Pembangkitan Area-4		3362	3358	3384	3362	3468	3430	3470	3558	4258	4497	5155	5382	5435	5410	5380	5372	5276	5145	4877	4867	4640	4503	4535	4159
(**) beban Area-4		1950	2005	2030	1954	1965	2111	2108	2114	2328	2601	3121	3163	3211	3190	3167	3147	3019	2941	2734	2682	2632	2432	2402	2320
Selisih (*) - (**)		1419	1363	1358	1415	1404	1319	1362	1444	1727	1898	2034	2223	2224	2221	2217	2225	2257	2204	2227	2229	2139	2103	2127	1839
Cadangan Sekatika		74	74	74	74	74	76	78	81	80	99	111	44	26	46	44	44	60	57	67	74	56	97	97	68
Cadangan Putar		1554	1554	1554	1554	1554	1633	1672	1553	1202	1115	457	227	177	202	232	237	333	423	825	636	729	684	584	1013

```
unit uObjFunc;

interface

uses uUtils,uGenerator,SysUtils;

type
  TObjFunc=class
  private
    FNgen,FNjam:integer;
    function GetBeban:dArr1;
    function GetRes:dArr1;
    function GetPLN:dArr2;
    function GetGen:TGenArr;
    procedure SetGen(const rGen:TGenArr);
    procedure SetBeban(const rBeban:dArr1);
    procedure SetRes(const rRes:dArr1);
    procedure SetPLN(const rPLN:dArr2);
    function isON(const rFlip:double):boolean;
    function isRampRate(const rJam:integer;const rPL:dArr2):boolean;
    function GetSortAFLC(const rXs:iArr1):iArr1;
    function CalcAFLC(const rXs:integer;
             const rNo:integer):double;
    procedure RepairAFLC(var rChrom:bArr2);
    procedure UpdateXs(const rChrom:bArr1;
              var rXs:iArr1);
    function CreateChromBase:bArr2;
    function CreateChromONOFF:bArr2;
    function GetSortChrom(const rXs:iArr1;
             const rRank:integer):bArr1;
    function HitungCostGen(const rPL:dArr2):dArr2;
    procedure GetSwap(var rChrom:bArr2);
    function doCariGreyZone(const rChrom:bArr2):bArr2;
    procedure UpdateChrom(var rChrom:bArr2);
  protected
    FBeban,FRes:dArr1;
    FPLN:dArr2;
    FGen:TGenArr;
    function isServe(const rJam:integer;const rChrom:bArr1):boolean;
    function HitungEcoDis(const rJam:integer;
             const rChrom1:bArr1):dArr1;overload;
    function HitungEcoDis(const rJam:integer;
             const rChrom1:bArr1;
             var rLmd:double):dArr1;overload;
    function HitungCostSUC(const rPL:dArr2):dArr2;overload;
    function HitungCostSUC(const rChrom:bArr2):dArr2;overload;
  public
    constructor Create;overload;
    constructor Create(const rBeban,rRes:dArr1;
                const rPLN:dArr2;
                const rGen:TGenArr);overload;
    function getSortAFLCGen:iArr1;
    function getRandomChrom(const rFlip:double):bArr2;
    function getChromFromAnother(const rChrom:bArr2;
             const rPflip:double):bArr2;
    function getConstructSolution(const rFlip:double):bArr2;
    function getNeighborSolution(const rChrom:bArr2):bArr2;
    procedure setLocalSearch(var rChrom:bArr2);
    procedure doHitungChrom(var rChrom:bArr2;
              var rCostTotal:double);overload;
    procedure doHitungChrom(var rChrom:bArr2;
              var rPL:dArr2;
              var rCostPerJam:dArr1;
              var rCostTotal:double);overload;
    procedure doHitungPLN(
```

```
            var rCostPerJam:dArr1;
            var rCostTotal:double);
    procedure doExecute(var rChrom:bArr2;
            var rPL:dArr2;
            var rCostPerJam:dArr1;
            var rCostTotal:double);
    function getPmax(const rNo:integer):double;
    function getCostMax:double;
    destructor Destroy;override;
    property Ngen:integer read FNgen write FNgen;
    property Njam:integer read FNjam write FNjam;
    property Gen:TGenArr read GetGen write SetGen;
    property Beban:dArr1 read GetBeban write SetBeban;
    property PLN:dArr2 read GetPLN write SetPLN;
    property Res:dArr1 read GetRes write SetRes;
  end;

var gObjFunc:TObjFunc;

implementation

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

constructor TObjFunc.Create(const rBeban,rRes:dArr1;
            const rPLN:dArr2;
            const rGen:TGenArr);
var i,j,Ncek:integer;
begin
  inherited Create;
  FNgen:=high(rGen);
  FNjam:=high(rBeban);
  Ncek:=high(rRes);
  if FNjam<>Ncek then raise Exception.Create('Dimensi matrix tidak
sama!');
  SetLength(FGen,FNgen+1);
  SetLength(FBeban,FNjam+1);
  SetLength(FRes,FNjam+1);
  for i:=1 to FNgen do
  begin
    FGen[i]:=TPembangkit.Create(rGen[i]);
  end;
  for i:=1 to FNjam do
  begin
    FBeban[i]:=rBeban[i];
    FRes[i]:=rRes[i];
  end;
  SetLength(FPLN,FNgen+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      FPLN[i,j]:=rPLN[i,j];
    end;
  end;
end;

//data accessing
function TObjFunc.GetBeban:dArr1;
var i:integer;
```

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

function TObjFunc.GetRes:dArr1;
var i:integer;
begin
  SetLength(result,FNjam+1);
  for i:=1 to FNjam do
  begin
    result[i]:=FRes[i];
  end;
end;

function TObjFunc.GetPLN:dArr2;
var i,j:integer;
begin
  SetLength(result,FNgen+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      result[i,j]:=FPLN[i,j];
    end;
  end;
end;

function TObjFunc.GetGen:TGenArr;
var i:integer;
begin
  SetLength(result,FNgen+1);
  for i:=1 to FNgen do
  begin
    result[i]:=TPembangkit.Create(FGen[i]);
  end;
end;

procedure TObjFunc.SetGen(const rGen:TGenArr);
var i:integer;
begin
  FNgen:=high(rGen);
  SetLength(FGen,FNgen+1);
  for i:=1 to FNgen do
  begin
    FGen[i]:=TPembangkit.Create(rGen[i]);
  end;
end;

procedure TObjFunc.SetBeban(const rBeban:dArr1);
var i,Ncek:integer;
begin
  if FNjam>0 then
  begin
    Ncek:=high(rBeban);
    if FNjam<>Ncek then raise Exception.Create('Dimensi matrik tidak
sama!');
  end
  else
  begin
    FNjam:=high(rBeban);
  end;
```

```
end;

function TObjFunc.isServe(const rJam:integer;const rChrom:bArr1):boolean;
var i:integer;
    load,sBebanMin,sBebanMax:double;
begin
  result:=true;
  sBebanMin:=0;
  sBebanMax:=0;
  for i:=1 to FNgen do
  begin
    if rChrom[i]=true then
    begin
      sBebanMin:=sBebanMin+FGen[i].Pmin;
      sBebanMax:=sBebanMax+FGen[i].Pmax;
    end;
  end;
  load:=FBeban[rJam]+FRes[rJam];
  if load<sBebanMin then result:=false;
  if load>sBebanMax then result:=false;
end;

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

function TObjFunc.CalcAFLC(const rXs:integer;
         const rNo:integer):double;
var SUC:double;
begin
  SUC:=0;
  if rXs<0 then
  begin
    if abs(rXs)>=FGen[rNo].Tdown then
    begin
      if abs(rXs)>(FGen[rNo].Tdown+FGen[rNo].Tcold) then
      begin
        SUC:=FGen[rNo].Sc;
      end
      else
      begin
        SUC:=FGen[rNo].Sh;
      end;
    end;
  end;
  result:=(FGen[rNo].GetBiaya(FGen[rNo].Pmax)+SUC)/FGen[rNo].Pmax;
end;
```

```
function TObjFunc.GetSortAFLC(const rXs:iArr1):iArr1;
var i,j:integer;
    AFLC:dArr1;
begin
  SetLength(result,FNgen+1);
  SetLength(AFLC,FNgen+1);
  for i:=1 to FNgen do
  begin
    result[i]:=i;
    AFLC[i]:=CalcAFLC(rXs[i],i);
  end;
  for i:=1 to FNgen-1 do
  begin
    for j:=i to FNgen do
    begin
      if AFLC[i]>AFLC[j] then
      begin
        Swap(AFLC[i],AFLC[j]);
        Swap(result[i],result[j]);
      end;
    end;
  end;
end;

function TObjFunc.GetSortChrom(const rXs:iArr1;
         const rRank:integer):bArr1;
var i:integer;
    SortAFLC:iArr1;
begin
  SetLength(result,FNgen+1);
  for i:=1 to FNgen do
  begin
    result[i]:=false;
  end;
  SortAFLC:=GetSortAFLC(rXs);
  for i:=1 to rRank do
  begin
    result[SortAFLC[i]]:=true;
  end;
end;

procedure TObjFunc.CpdaeXs(const rChrom:bArr1;
          var rXs:iArr1);
var i:integer;
begin
  for i:=1 to FNgen do
  begin
    if rChrom[i]=true then
    begin
      if rXs[i]<0 then
      begin
        rXs[i]:=1;
      end
      else
      begin
        rXs[i]:=rXs[i]+1;
      end;
    end
    else
    begin
      if rXs[i]<0 then
      begin
        rXs[i]:=rXs[i]-1;
      end
```

```
      else
      begin
        rXs[i]:=-1;
      end;
    end;
  end;
end;

function TObjFunc.CreateChromBase:bArr2;
var i,j,k:integer;
    chroml:bArr1;
    Xs:iArr1;
    serve:boolean;
begin
  SetLength(result,FNgen+1,FNjam+1);
  SetLength(chroml,FNgen+1);
  SetLength(Xs,FNGen+1);
  for i:=1 to FNgen do
  begin
    Xs[i]:=FGen[i].InitSt;
  end;
  for i:=1 to FNjam do
  begin
    for j:=1 to FNgen do
    begin
      chroml:=GetSortChrom(Xs,j);
      serve:=isServe(i,chroml);
      if serve=true then
      begin
        for k:=1 to FNgen do
        begin
          result[k,i]:=chroml[k];
        end;
        UpdateXs(chroml,Xs);
        break;
      end;
    end;
  end;
end;

function TObjFunc.CreateChromONOFF:bArr2;
var i,j,k:integer;
    chroml:bArr1;
    Xs:iArr1;
    serve:boolean;
begin
  SetLength(result,FNgen+1,FNjam+1);
  SetLength(chroml,FNgen+1);
  SetLength(Xs,FNgen+1);
  for i:=1 to FNgen do
  begin
    Xs[i]:=FGen[i].InitSt;
  end;
  for i:=1 to FNjam do
  begin
    for j:=FNgen downto 1 do
    begin
      chroml:=GetSortChrom(Xs,j);
      serve:=isServe(i,chroml);
      if serve=true then
      begin
        for k:=1 to FNgen do
        begin
          result[k,i]:=chroml[k];
        end;
```

```
        UpdateXs(chrom1,Xs);
        break;
      end;
    end;
  end;
end;

procedure TObjFunc.RepairAFLC(var rChrom:bArr2);
var i,j,NUnitON:integer;
    Xs,SortAFLC:iArr1;
    Chrom1:bArr1;
begin
  SetLength(Xs,FNgen+1);
  SetLength(Chrom1,FNgen+1);
  SetLength(SortAFLC,FNgen+1);
  for i:=1 to FNgen do
  begin
    Xs[i]:=FGen[i].InitSt;
  end;
  for i:=1 to FNjam do
  begin
    NUnitON:=0;
    for j:=1 to FNgen do
    begin
      if rChrom[j,i]=true then
      begin
        inc(NUnitON);
      end;
    end;
    for j:=1 to FNgen do
    begin
      rChrom[j,i]:=false;
      Chrom1[j]:=false;
    end;
    SortAFLC:=GetSortAFLC(Xs);
    for j:=1 to NUnitON do
    begin
      rChrom[SortAFLC[j],i]:=true;
      Chrom1[SortAFLC[j]]:=true;
    end;
    UpdateXs(Chrom1,Xs);
  end;
end;

function TObjFunc.getSortAFLCGen:iArr1;
var i,j:integer;
    AFLC:dArr1;
begin
  SetLength(AFLC,Ngen+1);
  SetLength(result,Ngen+1);
  for i:=1 to Ngen do
  begin
    AFLC[i]:=FGen[i].AFLC;
    result[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
        Swap(AFLC[i],AFLC[j]);
        Swap(result[i],result[j]);
      end;
    end;
```

```
var di,dj,j,Ton,Toff,L,tl:integer;
    chrom1:bArr1;
begin
  SetLength(result,FNgen+1,tNjam+1);
  for di:=1 to FNgen do
  begin
    for dj:=1 to FNjam do
    begin
      result[di,dj]:=rChrom[di,dj];
    end;
  end;
  repeat
    di:=round(1+random*(FNgen-1));
    dj:=round(1+random*(FNjam-1));
    if result[di,dj]=true then
    begin
      //move backward in time to find length of the ON period
      Ton:=1;
      for j:=dj-1 downto 1 do
      begin
        tl:=j;
        if result[di,j]=true then
        begin
          inc(Ton);
        end
        else
        begin
          tl:=j+1;
          break;
        end;
      end;
      //move forward in time to find length of the ON period
      for j:=dj+1 to FNjam do
      begin
        if result[di,j]=true then
        begin
          inc(Ton);
        end
        else
        begin
          break;
        end;
      end;
      if Ton=FGen[di].Tup then
      begin
        //turn the unit OFF in all hours comprising Ton
        result[di,dj]:=false;
        for j:=dj-1 downto 1 do
        begin
          if result[di,j]=true then
          begin
            result[di,j]:=false;
          end
          else
          begin
            break;
          end;
        end;
        for j:=dj+1 to FNjam do
        begin
          if result[di,j]=true then
          begin
            result[di,j]:=false;
          end
          else
```

```
          else
          begin
            break;
          end;
        end;
      end
      else
      begin
        L:=round(1+random*(Toff-PGen[di].Tdown));
        //turn the unit ON for tl to tl+L
        for j:=tl to tl+L do
        begin
          result[di,j]:=true;
        end;
      end;
    end;
    SetLength(chrom1,PNgen+1);
    for j:=1 to PNgen do
    begin
      chrom1[j]:=result[j,dj];
    end;
  until isServe(dj,chrom1)=true;
end;

function TObjFunc.getConstructSolution(const rFlip:double):bArr2;
var i,j:integer;
    chromBase,chromONOFF:bArr2;
begin
  chromBase:=CreateChromBase;
  chromONOFF:=CreateChromONOFF;
  SetLength(result,Ngen+1,Njam+1);
  for i:=1 to Ngen do
  begin
    for j:=1 to Njam do
    begin
      if chromBase[i,j]=true then
      begin
        result[i,j]:=true;
      end
      else
      begin
        if chromONOFF[i,j]=true then
        begin
          if isON(rFlip)=true then
          begin
            result[i,j]:=true;
          end
          else
          begin
            result[i,j]:=false;
          end;
        end;
      end;
    end;
  end;
  RepairAFLC(result);
  GetSwap(result);
end;

procedure TObjFunc.setLocalSearch(var rChrom:bArr2);
var i,j:integer;
    CostBest,CostCek:double;
    greyChrom:bArr2;
begin
  doHitungChrom(rChrom,CostBest);
```

```
  greyChrom:=coCar_GreyKone(rChrom);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      if greyChrom[i,j]=true then
      begin
        rChrom[i,j-1]:=true;
        doHiturgChrom(rChrom,CostCek);
        if CostBest>CostCek then
        begin
          CostBest:=CostCek;
        end
        else
        begin
          rChrom[i,j-1]:=false;
        end;
      end;
    end;
  end;
end;

procedure TObjFunc.GetSwap(var rChrom:bArr2);
var i,j,k,init,pos:integer;
begin
  for i:=1 to FNgen do
  begin
    init:=FGen[i].InitSt;
    for j:=1 to FNjam do
    begin
      if rChrom[i,j]=true then
      begin
        if init<0 then
        begin
          if abs(init)>=FGen[i].Idown then
          begin
            init:=1;
          end
          else
          begin
            pos:=j-init;
            if pos<1 then
            begin
              pos:=1;
            end;
            for k:=pos to j-1 do
            begin
              rChrom[i,k]:=true;
            end;
          end;
        end
        else if init>0 then
        begin
          init:=init+1;
        end;
      end
      else if rChrom[i,j]=false then
      begin
        if init<0 then
        begin
          init:=init-1;
        end
        else if init>0 then
        begin
          if init>=FGen[i].Tup then
```

```
          begin
            init:=-1;
          end
          else
          begin
            rChrom[i,j]:=true;
            init:=init-1;
          end;
        end;
      end;
    end;
  end;
end;

procedure TObjFunc.UpdateChrom(var rChrom:bArr2);
var i,j:integer;
    chromBase:bArr2;
begin
  chromBase:=CreateChromBase;
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      if (rChrom[i,j]=false) and (chromBase[i,j]=true) then
      begin
        rChrom[i,j]:=true;
      end;
    end;
  end;
  RepairAFLC(rChrom);
  GetSwap(rChrom);
end;

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

function TObjFunc.HitungEcoDis(const rJam:integer;
         const rChroml:bArr1):dArr1;
var i,j:integer;
    Status:bArr1;
    LoadCek,Pa,Pb,Lmd,LoadSplit:double;
    aBeban,diffa2,dilfa1,Cek,tes:double;
begin
  SetLength(Status,FNgen+1);
  for i:=1 to FNgen do
  begin
    Status[i]:=rChroml[i];
    FGen[i].Daya:=0;
  end;
  aBeban:=FBeban[rJam];
  LoadCek:=aBeban;
  LoadSplit:=aBeban;
  for i:=1 to 15 do
  begin
```

```
Pa:=0;
Pb:=0;
for j:=1 to FNgen do
begin
  if Status[j] then
  begin
    diffa2:=FGen[j].a2*2;
    diffa1:=FGen[j].a1;
    Pa:=Pa+1/diffa2;
    Pb:=Pb+diffa1/diffa2;
  end;
end;
if Pa<>0 then
begin
  Lmd:=(LoadSplit+Pb)/Pa;
end
else
begin
  Lmd:=LoadSplit+Pb;
end;
Cek:=0;
for j:=1 to FNgen do
begin
  if Status[j] then
  begin
    diffa2:=2*FGen[j].a2;
    diffa1:=FGen[j].a1;
    FGen[j].Daya:=(Lmd-diffa1)/diffa2;
    if FGen[j].Daya<FGen[j].Pmin then
    begin
      FGen[j].Daya:=FGen[j].Pmin;
    end;
    if FGen[j].Daya>FGen[j].Pmax then
    begin
      FGen[j].Daya:=FGen[j].Pmax;
    end;
  end;
  Cek:=Cek+FGen[j].Daya;
end;
tes:=LoadCek-Cek;
if (tes<0.0001) and (tes>-0.0001) then
begin
  break;
end
else if tes>0 then
begin
  for j:=1 to fNgen do
  begin
    if Status[j] then
    begin
      if FGen[j].Daya=FGen[j].PMax then
      begin
        Status[j]:=false;
        LoadSplit:=LoadSplit-FGen[j].Daya;
        if LoadSplit<0 then
        begin
          LoadSplit:=LoadSplit+FGen[j].Daya;
          Status[j]:=true;
        end;
      end;
    end;
  end;
end
else if tes<0 then
begin
```

```
      for j:=1 to fNgen do
      begin
        if Status[j] then
        begin
          if FGen[j].Daya=FGen[j].Pmin then
          begin
            Status[j]:=false;
            LoadSplit:=LoadSplit-FGen[j].Daya;
            if LoadSplit<0 then
            begin
              LoadSplit:=LoadSplit+FGen[j].Daya;
              Status[j]:=true;
            end;
          end;
        end;
      end;
    end;
    SetLength(result,FNgen+1);
    for i:=1 to FNgen do
    begin
      result[i]:=0;
      if rChrom1[i] then
      begin
        result[i]:=FGen[i].Daya;
      end;
    end;
  end;

function TObjFunc.HitungFnoDis(const rJam:integer;
          const rChrom1:bArr1;
          var rLmd:double):dArr1;
var i,j:integer;
    Status:bArr1;
    LoadCek,Pa,Pb,LoadSplit:double;
    aBeban,diffa2,diffa1,Cek,res:double;
begin
  SetLength(Status,FNgen-1);
  for i:=1 to FNgen do
  begin
    Status[i]:=rChrom1[i];
    FGen[i].Daya:=0;
  end;
  aBeban:=FDeban[rJam];
  LoadCek:=aBeban;
  LoadSplit:=aBeban;
  for i:=1 to 15 do
  begin
    Pa:=0;
    Pb:=0;
    for j:=1 to FNgen do
    begin
      if Status[j] then
      begin
        diffa2:=FGen[j].a2*2;
        diffa1:=FGen[j].a1;
        Pa:=Pa+1/diffa2;
        Pb:=Pb+diffa1/diffa2;
      end;
    end;
    if Pa<>0 then
    begin
      rLmd:=(LoadSplit+Pb)/Pa;
    end
    else
```

```
begin
  rLmd:=LoadSplit+Pb;
end;
Cek:=0;
for j:=1 to FNgen do
begin
  if Status[j] then
  begin
    diffa2:=2*FGen[j].a2;
    diffa1:=FGen[j].a1;
    FGen[j].Daya:=(rLmd-diffa1)/diffa2;
    if FGen[j].Daya<FGen[j].Pmin then
    begin
      FGen[j].Daya:=FGen[j].Pmin;
    end;
    if FGen[j].Daya>FGen[j].Pmax then
    begin
      FGen[j].Daya:=FGen[j].Pmax;
    end;
  end;
  Cek:=Cek+FGen[j].Daya;
end;
tes:=LoadCek-Cek;
if (tes<0.0001) and (tes>-0.0001) then
begin
  break;
end
else if tes>0 then
begin
  for j:=1 to fNgen do
  begin
    if Status[j] then
    begin
      if FGen[j].Daya=FGen[j].PMax then
      begin
        Status[j]:=false;
        LoadSplit:=LoadSplit-FGen[j].Daya;
        if LoadSplit<0 then
        begin
          LoadSplit:=LoadSplit+fGen[j].Daya;
          Status[j]:=true;
        end;
      end;
    end;
  end;
end
else if tes<0 then
begin
  for j:=1 to FNgen do
  begin
    if Status[j] then
    begin
      if FGen[j].Daya=FGen[j].Pmin then
      begin
        Status[j]:=false;
        LoadSplit:=LoadSplit-FGen[j].Daya;
        if LoadSplit<0 then
        begin
          LoadSplit:=LoadSplit+FGen[j].Daya;
          Status[j]:=true;
        end;
      end;
    end;
  end;
end;
```

```
  end;
  SetLength(result,FNgen+1);
  for i:=1 to FNgen do
  begin
    result[i]:=0;
    if rChrom[i] then
    begin
      result[i]:=FGen[i].Daya;
    end;
  end;
end;

function TObjFunc.HitungCostGen(const rPL:dArr2):dArr2;
var i,j:integer;
begin
  SetLength(result,FNgen+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      result[i,j]:=FGen[i].GetDaya(rPL[i,j]);
    end;
  end;
end;

function TObjFunc.HitungCostSUC(const rPL:dArr2):dArr2;
var i,j,init,tcold:integer;
begin
  SetLength(result,FNgen+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    init:=FGen[i].InitSt;
    tcold:=FGen[i].Tdown+FGen[i].Tcold;
    for j:=1 to FNjam do
    begin
      result[i,j]:=0;
      if rPL[i,j]<>0 then
      begin
        if init>0 then
        begin
          init:=init+1;
        end
        else if init<0 then
        begin
          if abs(init)<=tcold then
          begin
            result[i,j]:=FGen[i].Sh;
          end
          else
          begin
            result[i,j]:=FGen[i].Sc;
          end;
          init:=1;
        end;
      end
      else if rPL[i,j]=0 then
      begin
        if init>0 then
        begin
          init:=-1;
        end
        else if init<0 then
        begin
          init:=init-1;
        end;
```

```
      end;
    end;
  end;
end;

function TObjFunc.HitungCostSUC(const rChrom:bArr2):dArr2;
var i,j,init,tcold:integer;
begin
  SetLength(result,FNgen+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    init:=FGen[i].InitSt;
    tcold:=FGen[i].Tdown+FGen[i].Tcold;
    for j:=1 to FNjam do
    begin
      result[i,j]:=0;
      if rChrom[i,j]=true then
      begin
        if init>0 then
        begin
          init:=init+1;
        end
        else if init<0 then
        begin
          if abs(init)<=tcold then
          begin
            result[i,j]:=FGen[i].Sh;
          end
          else
          begin
            result[i,j]:=FGen[i].Sc;
          end;
          init:=1;
        end;
      end
      else if rChrom[i,j]=false then
      begin
        if init>0 then
        begin
          init:=-1;
        end
        else if init<0 then
        begin
          init:=init-1;
        end;
      end;
    end;
  end;
end;

procedure TObjFunc.doHitungChrom(var rChrom:bArr2;
          var rCostTotal:double);
var i,j:integer;
    PLa:dArr1;
    PL,CostGen,CostSUC:dArr2;
    chrom1:bArr1;
begin
  UpdateChrom(rChrom);
  SetLength(PL,FNgen+1,FNjam+1);
  SetLength(chrom1,FNgen+1);
  SetLength(PLa,FNgen+1);
  for i:=1 to FNjam do
  begin
    for j:=1 to FNgen do
    begin
```

```
      chrom1[j]:=rChrom[j,i];
    end;
    PLa:=HitungEcoDis(i,chrom1);
    for j:=1 to FNgen do
    begin
      PL[j,i]:=PLa[j];
    end;
  end;
  CostGen:=HitungCostGen(PL);
  CostSUC:=HitungCostSUC(PL);
  rCostTotal:=0;
  for i:=1 to FNjam do
  begin
    for j:=1 to FNgen do
    begin
      rCostTotal:=rCostTotal+CostGen[j,i]+CostSUC[j,i];
    end;
  end;
end;

procedure TObjFunc.doHitungChrom(var rChrom:bArr2;
          var rPL:dArr2;
          var rCostPerJam:dArr1;
          var rCostTotal:double);
var i,j:integer;
    PLa:dArr1;
    CostGen,CostSUC:dArr2;
    chrom1:bArr1;
begin
  UpdateChrom(rChrom);
  SetLength(rPL,FNgen+1,FNjam+1);
  SetLength(rCostPerJam,FNjam+1);
  SetLength(chrom1,FNgen+1);
  SetLength(PLa,FNgen+1);
  for i:=1 to FNjam do
  begin
    for j:=1 to FNgen do
    begin
      chrom1[j]:=rChrom[j,i];
    end;
    PLa:=HitungEcoDis(i,chrom1);
    for j:=1 to FNgen do
    begin
      rPL[j,i]:=PLa[j];
    end;
  end;
  CostGen:=HitungCostGen(rPL);
  CostSUC:=HitungCostSUC(rPL);
  SetLength(rCostPerJam,FNjam+1);
  rCostTotal:=0;
  for i:=1 to FNjam do
  begin
    rCostPerJam[i]:=0;
    for j:=1 to FNgen do
    begin
      rCostPerJam[i]:=rCostPerJam[i]+CostGen[j,i]+CostSUC[j,i];
    end;
    rCostTotal:=rCostTotal+rCostPerJam[i];
  end;
end;

procedure TObjFunc.doHitungPLN(var rCostPerJam:dArr1;
          var rCostTotal:double);
var i,j:integer;
    CostGen,CostSUC:dArr2;
```

```
begin
  SetLength(rCostPerJam,FNjam+1);
  CostGen:=HitungCostGen(FPLN);
  CostSUC:=HitungCostSUC(FPLN);
  rCostTotal:=0;
  for i:=1 to FNjam do
  begin
    rCostPerJam[i]:=0;
    for j:=1 to FNgen do
    begin
      rCostPerJam[i]:=rCostPerJam[i]+CostGen[j,i]+CostSUC[j,i];
    end;
    rCostTotal:=rCostTotal+rCostPerJam[i];
  end;
end;

//data output
function TObjFunc.doCariGreyZone(const rChrom:bArr2):bArr2;
var i,j,init,tcold:integer;
begin
  SetLength(result,FNgen+1,FNjam+1);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      result[i,j]:=false;
    end;
  end;
  for i:=1 to FNgen do
  begin
    init:=FGen[i].InitSt;
    tcold:=FGen[i].Tdown+FGen[i].Tcold;
    for j:=1 to FNjam do
    begin
      if rChrom[i,j]=true then
      begin
        if init<0 then
        begin
          if abs(init)-(tcold+1) then
          begin
            result[i,j]:=true;
          end;
          init:=1;
        end
        else if init>0 then
        begin
          init:=init+1;
        end;
      end
      else if rChrom[i,j]=false then
      begin
        if init<0 then
        begin
          init:=init-1;
        end
        else if init>0 then
        begin
          init:=-1;
        end;
      end;
    end;
  end;
end;

procedure TObjFunc.doExecute(var rChrom:bArr2;
```

```
          var rPL:dArr2;
          var rCostPerJam:dArr1;
          var rCostTotal:double);
var i,j:integer;
    CostBest,CostCek:double;
    greyChrom:bArr2;
begin
  rChrom:=CreateChromBase;
  GetSwap(rChrom);
  doHitungChrom(rChrom,CostBest);
  greyChrom:=doCariGreyZone(rChrom);
  for i:=1 to FNgen do
  begin
    for j:=1 to FNjam do
    begin
      if greyChrom[i,j]=true then
      begin
        rChrom[i,j-1]:=true;
        doHitungChrom(rChrom,CostCek);
        if CostBest>CostCek then
        begin
          CostBest:=CostCek;
        end
        else
        begin
          rChrom[i,j-1]:=false;
        end;
      end;
    end;
  end;
  doHitungChrom(rChrom,rPL,rCostPerJam,rCostTotal);
end;

function TObjFunc.getPmax(const rNo:integer):double;
begin
  if (rNo<0) or (rNo>FNgen) then
  begin
    raise Exception.Create('Diluar batas index!');
  end;
  result:=FGen[rNo].Pmax;
end;

function TObjFunc.getCostMax:double;
var i:integer;
begin
  result:=0;
  for i:=1 to FNgen do
  begin
    result:=result+FGen[i].GetBiaya(FGen[i].Pmax);
  end;
end;

end.
```

```
unit uTabuSearch;

interface

uses uUtils,uObjFunc,uKvoVar,uDecode;

type
  TTabuSearch=class
  private
    FNgen,FNjam,FLength,FCount:integer;
    FPflip:double;
    FTabuList:TPopulasi;
    FBestIndi:TIndividu;
    function getIndividu(const rIndi:TIndividu):TIndividu;
    function isSameChrom(const rChrom1,rChrom2:iArr2):boolean;
    function isSameList(const rChrom:iArr2):boolean;
    function getBestListPosition:integer;
    function getWorstListPosition:integer;
    procedure InitTabuList;
    function getRandomNeighbor(const rNeigh:integer):TIndividu;
    function isAcceptedTabuList(var rIndi:TIndividu):boolean;
    function getBestChrom:iArr2;
  public
    constructor Create(const rCount,rNgen,rNjam:integer;
                       const rPflip:double);
    function getNewPopulasi(const rPopSize,rNeigh:integer):TPopulasi;
    property BestChrom:iArr2 read getBestChrom;
  end;

implementation

{ TTabuSearch }

constructor TTabuSearch.Create(const rCount,rNgen,rNjam:integer;
            const rPflip: double);
begin
  inherited Create;
  FCount:=rCount;
  FNgen:=rNgen;
  FNjam:=rNjam;
  FLength:=FNjam div 2;
  FPflip:=rPflip;
  InitTabuList;
end;

//data processing
function TTabuSearch.getIndividu(const rIndi:TIndividu):TIndividu;
var i,j:integer;
begin
  SetLength(result.chrom,FNgen,FLength);
  for i:=0 to FNgen-1 do
  begin
    for j:=0 to FLength-1 do
    begin
      result.chrom[i,j]:=rIndi.chrom[i,j];
    end;
  end;
  result.fitness:=rIndi.fitness;
end;

function TTabuSearch.isSameChrom(const rChrom1,rChrom2:iArr2):boolean;
var i,j:integer;
```

```
begin
  result:=true;
  for i:=0 to FNgen-1 do
  begin
    for j:=0 to Flength-1 do
    begin
      if rChrom1[i,j]<>rChrom2[i,j] then
      begin
        result:=false;
        break;
      end;
    end;
  end;
end;

function TTabuSearch.isSameList(const rChrom:iArr2):boolean;
var i:integer;
begin
  result:=false;
  for i:=0 to FCount-1 do
  begin
    if isSameChrom(rChrom,FTabuList[i].chrom)=true then
    begin
      result:=true;
      break;
    end;
  end;
end;

function TTabuSearch.getBestListPosition:integer;
var i:integer;
    min:double;
begin
  min:=FTabuList[0].fitness;
  result:=0;
  for i:=1 to FCount-1 do
  begin
    if min>FTabuList[i].fitness then
    begin
      min:=FTabuList[i].fitness;
      result:=i;
    end;
  end;
end;

function TTabuSearch.getWorstListPosition:integer;
var i:integer;
    max:double;
begin
  max:=FTabuList[0].fitness;
  result:=0;
  for i:=1 to FCount-1 do
  begin
    if max<FTabuList[i].fitness then
    begin
      max:=FTabuList[i].fitness;
      result:=i;
    end;
  end;
end;

procedure TTabuSearch.InitTabuList;
var i:integer;
    chromMin:bArr2;
begin
```

```
  SetLength(FTabuList,FCount);
  for i:=0 to FCount-1 do
  begin
    chromBin:=gObjFunc.getRandomChrom(FPflip);
    gObjFunc.doHitungChrom(chromBin,FTabuList[i].fitness);
    FTabuList[i].chrom:=DecodeChromBinToInt(chromBin);
  end;
end;

function TTabuSearch.getRandomNeighbor(const rNeigh:integer):TIndividu;
var i,pos:integer;
    min:double;
    chromBin:bArr2;
    tmpPop:TPopulasi;
begin
  SetLength(tmpPop,rNeigh);
  for i:=0 to rNeigh-1 do
  begin
    chromBin:=gObjFunc.getNeighborSolution(const rChrom:bArr2);
    gObjFunc.doHitungChrom(chromBin,tmpPop[i].fitness);
    tmpPop[i].chrom:=DecodeChromBinToInt(chromBin);
  end;
  min:=tmpPop[0].fitness;
  pos:=0;
  for i:=1 to rNeigh-1 do
  begin
    if min>tmpPop[i].fitness then
    begin
      min:=tmpPop[i].fitness;
      pos:=i;
    end;
  end;
  result:=getIndividu(tmpPop[pos]);
end;

function TTabuSearch.isAcceptedTabuList(var rIndi:TIndividu):boolean;
var pos:integer;
begin
  result:=false;
  pos:=getWorstListPosition;
  if isSameList(rIndi.chrom)=true then
  begin
    if rIndi.fitness<FTabuList[pos].fitness then
    begin
      FTabuList[pos]:=getIndividu(rIndi);
      result:=true;
    end;
  end
  else
  begin
    if rIndi.fitness<=FTabuList[pos].fitness then
    begin
      FTabuList[pos]:=getIndividu(rIndi);
      result:=true;
    end
    else
    begin
      pos:=getBestListPosition;
      rIndi:=getIndividu(FTabuList[pos]);
      result:=true;
    end;
  end;
end;

function TTabuSearch.getBestChrom:iArr2;
```

```
var i,j:integer;
begin
  SetLength(result,FNgen,FLength);
  for i:=0 to FNgen-1 do
  begin
    for j:=0 to FLength-1 do
    begin
      result[i,j]:=FBestIndi.chrom[i,j];
    end;
  end;
end;

function TTabuSearch.getNewPopulasi(const
rPopSize,rNeigh:integer):TPopulasi;
var sa:integer;
    IndiNeigh:TIndividu;
begin
  sa:=0;
  SetLength(result,rPopSize);
  repeat
    IndiNeigh:=getRandomNeighbor(rNeigh);
    if isAcceptedTabuList(IndiNeigh)=true then
    begin
      result[sa]:=getIndividu(IndiNeigh);
      sa:=sa+1;
    end;
  until sa=rPopSize;
end;

end.
```

```
unit uEvoPro;

interface

uses uUtils,uRandom,uObjFunc,uHasil,uDecode,uEvoVar,uTabuSearch;

type
  TEvoPro=class
  private
    FMaxGer,FPopSize,FLength,FChromMin,FChromMax:integer;
    FNTabuList:integer;
    FNger,FNjam:integer;
    FBetha,FPflip,FMin1,FAvg1,FMax1,FSumFitness,FKa,FPMutasi:double;
    FParent,FChild:TPopulasi;
    FBestIndi:TIndividu;
    FRandom:TRandomMU;
    FMin,FAvg,FMax:dArr1;
    FTabuSearch:TTabuSearch;
    function getMin:dArr1;
    function getAvg:dArr1;
    function getMax:dArr1;
    function getIndividu(const rIndi:TIndividu):TIndividu;
    function FindIndiMax:TIndividu;
    function FindChromMaxTS(const rPop:TPopulasi):bArr2;
    procedure SwapIndi(var rIndi1,rIndi2:TIndividu);
    procedure InitParent;
    procedure Statistik;
    function Seleksi:integer;
    procedure Mutasi;
    procedure EvolutionaryStrategy;
    procedure Kompetisi;
    procedure doHitung;
    function getBestChrom:bArr2;
  public
    constructor Create(const
rMaxGen,rPopSize,rNgen,rNjam,rNTabuList:integer;
                const rBetha,rPflip,rKa,rPMutasi:double);
    destructor Destroy;override;
    property MaxGen:integer read FMaxGen write FMaxGen;
    property PopSize:integer read FPopSize write FPopSize;
    property Betha:double read FBetha write FBetha;
    property Min:dArr1 read getMin;
    property Avg:dArr1 read getAvg;
    property Max:dArr1 read getMax;
    property BestChrom:bArr2 read getBestChrom;
  end;

implementation

//constructor
constructor TEvoPro.Create(const
rMaxGen,rPopSize,rNgen,rNjam,rNTabuList:integer;
            const rBetha,rPflip,rKa,rPMutasi:double);
var div2:integer;
begin
  inherited Create;
  FMaxGen:=rMaxGen;
  FPopSize:=rPopSize;
  FNgen:=rNgen;
  FNjam:=rNjam;
  FNTabuList:=rNTabuList;
  div2:=2;
  FLength:=FNjam div div2;
  FBetha:=rBetha;
  FPflip:=rPflip;
```

```
  FPMutasi:=rFMutasi;
  FKa:=rKa;
  FRandom:=TRandom.Create;
  FChromMin:=-1;
  FChromMax:=2;//round(pangkat(2,div2)-1);
  FTabuSearch:=TTabuSearch.Create(10,FNgen,FNjam,FPflip);
end;

//destructor
destructor TEvoPro.Destroy;
begin
  try
    FTabuSearch.Free;
    FRandom.Free;
  finally
    inherited Destroy;
  end;
end;

//data accessing
function TEvoPro.getMin:dArr;
var i:integer;
begin
  SetLength(result,FMaxGen);
  for i:=0 to FMaxGen-1 do
  begin
    result[i]:=FMin[i];
  end;
end;

function TEvoPro.getAvg:dArr;
var i:integer;
begin
  SetLength(result,FMaxGen);
  for i:=0 to FMaxGen-1 do
  begin
    result[i]:=FAvg[i];
  end;
end;

function TEvoPro.getMax:dArr;
var i:integer;
begin
  SetLength(result,FMaxGen);
  for i:=0 to FMaxGen-1 do
  begin
    result[i]:=FMax[i];
  end;
end;

//code  arti
//  1 1   2
//  1 0   1
//  0 0   0
//  0 1  -1

//data proccessing

function TEvoPro.getIndividu(const rindi:TIndividu):TIndividu;
var i,j:integer;
begin
  SetLength(result.chrom,FNgen,FLength);
  for i:=0 to FNgen-1 do
  begin
    for j:=0 to FLength-1 do
```

```
    begin
      result.chrom[i,j]:=rIndi.chrom[i,j];
    end;
  end;
  result.fitness:=rIndi.fitness;
end;

function TEvoPro.FindIndiMax:TIndividu;
var i:integer;
begin
  result:=getIndividu(FParent[0]);
  for i:=1 to FPopSize-1 do
  begin
    if result.fitness<FParent[i].fitness then
    begin
      result:=getIndividu(FParent[i]);
    end;
  end;
end;

function TEvoPro.FindChromMaxTS(const rPop:TPopulasi):bArr2;
var i,Npop:integer;
    best:TIndividu;
begin
  Npop:=high(rPop)+1;
  best:=getIndividu(rPop[0]);
  for i:=1 to Npop-1 do
  begin
    if best.fitness<rPop[i].fitness then
    begin
      best:=getIndividu(rPop[i]);
    end;
  end;
  result:=DecodeChromIntToBin(best.chrom);
end;

procedure TEvoPro.SwapIndi(var rIndi1,rIndi2:TIndividu);
var tmp:TIndividu;
begin
  tmp:=getIndividu(rIndi1);
  rIndi1:=getIndividu(rIndi2);
  rIndi2:=getIndividu(tmp);
end;

procedure TEvoPro.InitParent;
var i:integer;
    cost:double;
    chromBin:bArr2;
    LocPop:TPopulasi;
begin
  SetLength(FParent,FPopSize);
  SetLength(FChild,FPopSize);
  SetLength(FMin,FMaxGen);
  SetLength(FAvg,FMaxGen);
  SetLength(FMax,FMaxGen);
  for i:=0 to FPopSize-1 do
  begin
    SetLength(FParent[i].chrom,FNgen,FLength);
    SetLength(FChild[i].chrom,FNgen,FLength);
    LocPop:=FTabuSearch.getNewPopulasi(10,4);
    chromBin:=FindChromMaxTS(LocPop);
    //chromBin:=gObjFunc.getRandomChrom(FFlip);
    gObjFunc.doHitungChrom(chromBin,cost);
    FParent[i].fitness:=FKa/cost;
    FParent[i].chrom:=DecodeChromBinToInt(chromBin);
```

```
          begin
            FChild[i].chrom[j,k]:=FChromMin;
          end;
        end
        else
        begin
          FChild[i].chrom[j,k]:=FParent[mate].chrom[j,k];
        end;
      end;
    end;
    chromBin1:=DecodeChromIntToBin(FChild[i].chrom);
    gObjFunc.doHitungChrom(chromBin1,cost1);
    LocPop:=FTabuSearch.getNewPopulasi(4,2);
    chromBin2:=FindChromMaxTS(LocPop);
    gObjFunc.doHitungChrom(chromBin2,cost2);
    if cost1<cost2 Lain
    begin
      FChild[i].fitness:=FKa/cost1;
      FChild[i].chrom:=DecodeChromBinToInt(chromBin1);
    end
    else
    begin
      FChild[i].fitness:=FKa/cost2;
      FChild[i].chrom:=DecodeChromBinToInt(chromBin2);
    end;
  end;
end;

procedure TEvoPro.EvolutionaryStrategy;
var i,j,k,mate:integer;
    dX,cost:double;
    IndiMax:TIndividu;
    chromBin:bArr2;
begin
  IndiMax:=FindIndiMax;
  for i:=0 to FPopSize-1 do
  begin
    mate:=Seleksi;
    for j:=0 to FNgen-1 do
    begin
      for k:=0 to FLength-1 do
      begin
        if FRandom.NextBoolean(FPMutasi)=true then
        begin
          dX:=FRandom.NextGaussian(0,FBetha);
          FChild[i].chrom[j,k]:=round(FParent[mate].chrom[j,k]+dX);
          if FChild[i].chrom[j,k]>FChromMax then
          begin
            FChild[i].chrom[j,k]:=FChromMax;
          end;
          if FChild[i].chrom[j,k]<FChromMin then
          begin
            FChild[i].chrom[j,k]:=FChromMin;
          end;
        end
        else
        begin
          FChild[i].chrom[j,k]:=FParent[mate].chrom[j,k];
        end;
      end;
    end;
    chromBin:=DecodeChromIntToBin(FChild[i].chrom);
    gObjFunc.doHitungChrom(chromBin,cost);
    FChild[i].fitness:=FKa/cost;
    FChild[i].chrom:=DecodeChromBinToInt(chromBin);
```

```
  end;
  for i:=0 to FPopSize-1 do
  begin
    if FBestIndi.fitness<FChild[i].fitness then
    begin
      FBestIndi:=getIndividu(FChild[i]);
    end;
  end;
end;

procedure TEvoPro.Kompetisi;
var i,j,pos:integer;
    tmpPop:TPopulasi;
    Score:iArr;
begin
  SetLength(tmpPop,2*FPopSize);
  SetLength(Score,2*FPopSize);
  for i:=0 to FPopSize-1 do
  begin
    tmpPop[i]:=getIndividu(FParent[i]);
    tmpPop[FPopSize+i]:=getIndividu(FChild[i]);
    Score[i]:=0;
    Score[FPopSize+i]:=0;
  end;
  for i:=0 to 2*PopSize-1 do
  begin
    for j:=0 to 2*PopSize-1 do
    begin
      repeat
        pos:=FRandom.NextInt(0,2*FPopSize-1);
      until pos<>i;
      if tmpPop[i].fitness>tmpPop[pos].fitness then
      begin
        Score[i]:=Score[i]+1;
      end;
    end;
  end;
  for i:=0 to 2*PopSize-2 do
  begin
    for j:=i to 2*PopSize-1 do
    begin
      if Score[i]<Score[j] then
      begin
        Swap(Score[i],Score[j]);
        SwapIndi(tmpPop[i],tmpPop[j]);
      end;
    end;
  end;
  for i:=0 to FPopSize-1 do
  begin
    FParent[i]:=getIndividu(tmpPop[i]);
  end;
end;

procedure TEvoPro.doHitung;
var i:integer;
    tmpIndi:TIndividu;
begin
  InitParent;
  Statistik;
  FBestIndi:=FindIndiMax;
  for i:=0 to FMaxGen-1 do
  begin
    Mutasi;
    Kompetisi;
```

An Evolutionary Programming-Based Tabu Search Method For Solving The Unit Commitment Problem

C. Christober Asir Rajan and M. R. Mohan

Abstract—**This paper presents a new approach to solving the short-term unit commitment problem using an evolutionary programming-based tabu search (TS) method. The objective of this paper is to find the generation scheduling such that the total operating cost can be minimized, when subjected to a variety of constraints. This also means that it is desirable to find the optimal generating unit commitment in the power system for the next H hours. Evolutionary programming, which happens to be a global optimization technique for solving unit commitment problem, operates on a system, which is designed to encode each unit's operating schedule with regard to its minimum up/down time. In this, the unit commitment schedule is coded as a string of symbols. An initial population of parent solutions is generated at random. Here, each schedule is formed by committing all of the units according to their initial status ("hot start"). Here, the parents are obtained from a predefined set of solutions (i.e., each and every solution is adjusted to meet the requirements). Then, a random decommitment is carried out with respect to the unit's minimum downtimes, and TS improves the status by avoiding entrapment in local minima. The best population is selected by evolutionary strategy. The Neyveli Thermal Power Station (NTPS) Unit-II in India demonstrates the effectiveness of the proposed approach; extensive studies have also been performed for different power systems consisting of 10, 26, and 34 generating units. Numerical results are shown comparing the cost solutions and computation time obtained by using the evolutionary programming method and other conventional methods like dynamic programming, Lagrangian relaxation, and simulated annealing and tabu search in reaching proper unit commitment.**

Index Terms—**Dynamic programming, evolutionary programming, Lagrangian relaxation, Tabu search, unit commitment.**

I. Introduction

IN power stations, the investment is quite expensive, and the resources in operating them are considerably becoming sparse of which the focus turns to optimizing the operating cost of the power station. In today's world, it becomes an utmost necessity to meet the demand as well as optimize the generation. Unit commitment (UC) in power systems refers to the optimization problem for determining the on/off states of generating units that minimize the operating cost for a given time horizon. The solution of the unit commitment problem (UCP) is a complex optimization problem. The exact solution of the UCP can be obtained by a complete enumeration of all feasible combinations of generating units, which could be a very huge number. The unit commitment has commonly been formulated as a nonlinear, large-scale, mixed-integer combinational optimization problem.

Tabu search (TS) is a powerful optimization procedure that has been successfully applied to a number of combinatorial optimization problems. It has the ability to avoid entrapment in local minima. TS employs a flexible memory system (in contrast to "memoryless" systems, such as simulated annealing and genetic algorithm, and rigid memory system such as in branch-and-bound). Specific attention is given to the short-term memory component of TS, which has provided solutions superior to the best obtained with other methods for a variety of problems.

Research endeavors, therefore, have been focused on efficient, near-optimal UC algorithms, which can be applied to large-scale power systems and have reasonable storage and computation time requirements. A survey of existing literature [1]–[22] on the problem reveals that various numerical optimization techniques have been employed to approach the complicated unit commitment problem. More specifically, these are the dynamic programming (DP) method, the mixed integer programming (MIP) method, the Lagrangian relaxation (LR) method, the branch-and-bound (BB) method, the expert system (ES), the fuzzy theorem (FT) method, the hop field (H) method, the TS method, the genetic algorithm (GA), the artificial neural network (ANN), the integration of genetic algorithm, tabu search, simulated annealing (GTS), the TS and decomposition method (TSD), the extended neighborhood search algorithm (ENSA), evolutionary programming (EP), and so on. The major limitations of the numerical techniques are the problem dimensions, large computational time, and complexity in programming.

The DP method [1], [2], [13] is flexible, but the disadvantage is the "curse of dimensionality," which results in more mathematical complexity and increase in computation time if the constraints are taken in to consideration. The MIP methods [3], [4] for solving the UCPs fail when the number of units increases because they require a large memory and suffer from great computational delay. The LR approach [5]–[8] to solve the short-term UC problems was found to provide a faster solution but will fail to obtain solution feasibility and solution quality problems and becomes complex if the number of units increases. The BB method [9] employs a linear function to represent fuel cost and start-up cost and obtains lower and upper bounds. The difficulty of this method is the exponential growth in the execution time for systems of practical size. An ES algorithm [10], [13] rectifies the complexity in calculations and saving in computation time, but it will face the problem if the new schedule is different from the schedule in the database. In the FT method [11], [13], using fuzzy set solves the forecasted load schedules error but it will also suffer from complexity. The H neural network

Manuscript received July 8, 2003.

C. C. A. Rajan is with the Department of Electrical and Electronics Engineering, Pondicherry Engineering College, Pondicherry 605014, India (e-mail: asir_70@hotmail.com).

M. R. Mohan is with the School of Electrical Engineering, Anna University, Tamil Nadu 600025, India (e-mail: mohan@annauniv.edu).

Digital Object Identifier 10.1109/TPWRS.2003.821422

where

U_{it} unit i status at hour t = 1(if unit is ON)— 0 (if unit is OFF);
V_{it} unit i start up/shut down status at hour t = 1 if the unit is started at hour t and 0 otherwise;
F_T total operating cost over the schedule horizon ($/h);
S_{it} startup cost of unit i at hour t ($/h).

b. Constraints

Depending on the nature of the power system under study, the UCP is subject to many constraints, the main being the load balance constraints and the spinning reserve constraints. The other constraints include the thermal constraints, fuel constraints, security constraints, etc. [24]

1) Load Balance Constraints: The real power generated must be sufficient enough to meet the load demand and must satisfy the following factors given in (4):

$$\sum_{i=1}^{N} P_{it} U_{it} = PD_t \tag{4}$$

where

PD_t system peak demand at hour t (MW);
N number of available generating units;
$U(0,1)$ uniform distribution with parameters 0 and 1;
$UD(a,b)$ discrete uniform distribution with parameters a and b.

2) Spinning Reserve Constraints: The spinning reserve is the total amount of real power generation available from all synchronized units minus the present load plus the losses. The reserve is considered to be a prespecified amount or a given percentage of the forecasted peak demand. It must be sufficient enough to meet the loss of the most heavily loaded unit in the system. This has to satisfy the equation given in (5)

$$\sum_{i=1}^{N} P_{\max_i} U_{it} >= (PD_t + R_t); 1 \le t \le T \tag{5}$$

where

$P_{\max_i}$ maximum generation limit of unit i;
R_t spinning reserve at time t (MW);
T scheduled time horizon (24 h).

3) Thermal Constraints: The temperature and pressure of the thermal units vary very gradually and the units must be synchronized before they are brought online. A time period of even 1 h is considered as the minimum downtime of the units. There are certain factors, which govern the thermal constraints, like minimum uptime, minimum downtime, and crew constraints.

a) Minimum uptime: If the units have already been shut down, there will be a minimum time before they can be restarted and the constraint is given in (6)

$$Ton_i \ge Tup_i \tag{6}$$

where

Ton_i duration for which unit i is continuously ON (in hours);
Tup_i unit i minimum up time (in hours).

b) Minimum downtime: If all the units are running already, they cannot be shut down simultaneously and the constraint is given in (7)

$$Toff_i \ge Tdown_i \tag{7}$$

where

$Tdown_i$ unit i minimum down time (in hours);
$Toff_i$ duration for which unit i is continuously OFF (in hours).

4) Must Run Units: Generally, in a power system, some of the units are given a must run status in order to provide voltage support for the network.

5) Ramping Constraints: If the ramping constraints are included, the quality of the solution will be improved but the inclusion of ramp-rate limits can significantly enlarge the state space of production simulation and, thus, increase its computational requirements, and it results in significantly more states to be evolved and more strategies to be saved. Hence, the CPU time will be increased.

When ramp-rate limits are ignored, the number of generators consecutive online/offline hours at hour t provides adequate state description for making its commitment decision at hour (t + 1). When ramp-rate limits are modeled, the state description becomes inadequate. An additional status, generators energy generation capacity at hour t is also required for making its commitment decision at hour (t + 1). These additional descriptions add one more dimension to the state space and, thus, significantly increase the computational requirements. Therefore, we have not included them in this algorithm.

III. Tabu Search

A. Overview

In solving the UCP, two types of variables need to be determined. The unit's status variables U and V, which are integer variables and the units, output power variables P that are continuous variables. The problem can then be decomposed into two subproblems, a combinatorial problem in U and V and a nonlinear optimization problem in P.

TS is used to solve the combinatorial optimization, and the nonlinear optimization is solved via a quadratic programming routine [14]. The flowchart for TS is shown in "Fig. 1."

The proposed algorithm contains three major steps:

- first, generating randomly feasible trail solutions;
- second, calculating the objective function of the given solution by solving the EDP;
- third, applying the TS procedures to accept or reject the solution in hand.

B. TS General Algorithm

Step 0) Assume that the fuel costs to be fixed for each hour and all of the generators share the loads equally.
Step 1) By optimum allocation, find the initial feasible solution (U_i, V_i).
Step 2) Demand is taken as the control parameter.
Step 3) Generate the trial solution.
Step 4) Calculate the total operating cost F_i as the summation of running cost and start up–shut down cost.
Step 5) Tabulate the fuel cost for each unit for every hour.

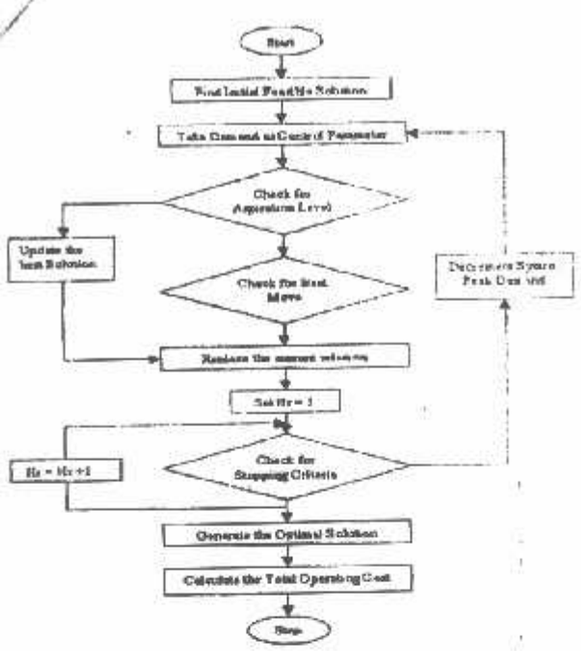


Fig. 1. Flowchart of TS algorithm.

C. Generating Trial Solution

The neighbors should be randomly generated, feasible, and span as much as possible the problem solution space. Because of the constraints in the UCP, this is not a simple matter. The most difficult constraints to satisfy are the minimum up/down times. The implementation of new rules to obtain randomly feasible solutions faster is done by the rules are described in [14].

D. Generating an Initial Solution

The TS algorithm requires a starting feasible schedule, which satisfies all of the system and units constraints. This schedule is randomly generated. The algorithm given in [14] is used to find this starting solution.

E. Operating Cost Calculation

Once a trial solution is obtained, the corresponding total operating cost is determined. Since the production cost is a quadratic function, the EDP is solved using a quadratic programming routine. The startup cost is then calculated for the given schedule.

F. Stopping Criteria

There may be several stopping criteria for the search. For this implementation, the search is stopped if the following conditions are satisfied.

- The load balance constraints are satisfied.
- The spinning reserve constraints are satisfied.

G. Tabu List (TL)

Tabu list (TL) is controlled by the trial solutions in the order in which they are made. Each time a new element is added to the "bottom" of a list, the oldest element on the list is dropped from the "top." Empirically, TL sizes, which provide good results, often grow with the size of the problem and stronger restrictions are generally coupled with smaller sizes [14]. Best sizes of TL lie in an intermediate range between these extremes. In some applications, a simple choice of TL size in a range centered on seven seems to be quite effective.

H. Aspiration Criteria

Another important criteria of TS arises when the move under consideration has been found to be tabu. Associated with each entry in the tabu list there is a certain value for the evaluation function called "aspiration level." Normally, the aspiration level criteria are designed to override tabu status if a move is "good enough" [14].

I. Implementation

To solve for the UCP using TS, software in the Turbo C package is developed. The software provides an interactive approach in dealing with the various data input required for solving the UCP from the user.

IV. Evolutionary Programming

A. Introduction

EP is a mutation-based evolutionary algorithm applied to discrete search spaces. D. Fogel (Fogel, 1988) [26]–[28] extended the initial work of his father L. Fogel (Fogel, 1962) [26]–[28] for applications involving real-parameter optimization problems. Real-parameter EP is similar in principle to evolution strategy (ES), in that normally distributed mutations are performed in both algorithms. Both algorithms encode mutation strength (or variance of the normal distribution) for each decision variable and a self-adapting rule is used to update the mutation strengths. Several variants of EP have been suggested (Fogel, 1992) [26]–[28].

B. Evolutionary Strategies

For the case of evolutionary strategies, Fogel remarks "evolution can be categorized by several levels of hierarchy: the gene, the chromosome, the individual, the species, and the ecosystem" [26]–[28]. Thus, while genetic algorithms stress models of genetic operators, ES emphasize mutational transformation that maintain behavioral linkage between each parent and its offspring at the level of the individual. ES are a joint development of Bienert *et al.* [26]–[28]. The first applications were experimental and addressed some optimization problems in hydrodynamics.

C. EP General Algorithm

EP [21], [26]–[28] is conducted as a sequence of operations and is given below. The schematic diagram of the EP algorithm is shown in "Fig. 2."

1) The initial population is determined by setting $s_i = S_i \sim U(a_k, b_k)^k$ $i = 1, \ldots, m$, where S_i is a random vector, s_i is the outcome of the random vector, $U(a_k, b_k)^k$ denotes a uniform distribution ranging over $[a_k, b_k]$ in each of k dimensions, and m is the number of parents.
2) Each s_i, $i = 1, \ldots, m$, is assigned a fitness score $\theta(s_i) = \theta(F(s_i), s_i)$, where F maps $s_i \to \mathbb{R}$ and denotes the true

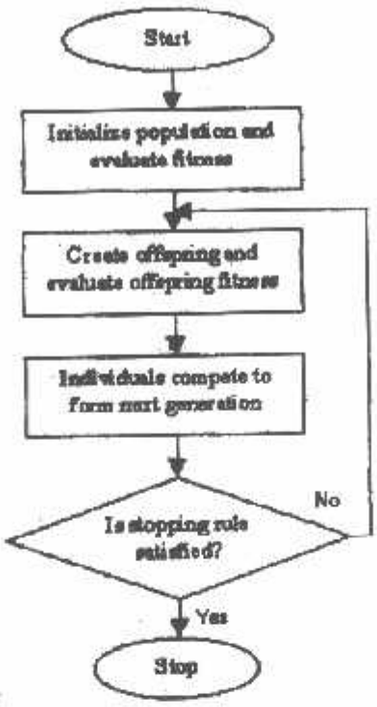


Fig. 2. Schematic diagram of EP algorithm.

fitness of s_i, v_i represents random alteration in the instantiation of s_i, random variation imposed on the evaluation of $F(s_i)$, or satisfies another relation $\approx$, and $G(F(s_i), v_i)$ describes the fitness score to be assigned. In general, the functions F and G can be as complex as required. For example, F may be a function not only of a particular s_i, but also of other members of the population, conditioned on a particular s_i.

3) Each s_i, $i = 1, \ldots, m$, is altered and assigned to s_{i+m} such that

$$s_{i+m} = s_{i,j} + N(0, \beta j\, \vartheta(s_i) + z_j),\ j = 1, \ldots, k$$

$N(0, \beta j \vartheta(s_i) + z_j)$ represents a Gaussian random variable with mean μ and variance σ^2, β_j is a constant of proportionality to scale $\vartheta(s_i)$, and z_j represents an offset to guarantee a minimum amount of variance.

4) Each s_{i+m}, $i = 1, \ldots, m$, is assigned a fitness score

$$\vartheta(s_{i+m}) = G(F(s_{i+m}), s_{i+m}).$$

5) For each s_i, $i = 1, \ldots, 2m$, a value wi is assigned according to

$$w_i = \sum_{t=1}^{c} w_t^*$$

$$w_t^* = \begin{cases} 1, & \text{if } \vartheta(s_i) \le \vartheta(s_i) \\ 0, & \text{otherwise} \end{cases}$$

where $\rho = \lfloor 2mu_1 + 1 \rfloor$, $\rho \neq i$, and $\lfloor x \rfloor$ denote the greatest integer less than or equal to x, c is the number of competitions, and u1 ~ U(0,1).

6) The solutions s_i $I = 1 \ldots 2m$ are ranked in descending order of their corresponding value W_i [with preference to their actual scores $\vartheta(s_i)$ if there are more than m solutions attaining a value of c]. The first m solutions are transcribed along with their corresponding values $\vartheta'(s_i)$ to be the basis of the next generation.

7) The process proceeds to step 3 unless the available execution time is exhausted or an acceptable solution has been discovered.

D. Evolutionary Programming for UCP

1) Initialize the parent vector $p = [p_1, p_2, \ldots p_n]$, $i = 1, 2, \ldots N_p$ such that each element in the vector is determined by $p_j \sim \text{random}(p_{j\min}, p_{j\max})$, $j = 1, 2, \ldots N$, with one generator as a dependent generator.
2) Initialize the parent vector $p = [p_1, p_2, \ldots p_n]$, $i = 1, 2, \ldots N_p$ such that each element in the vector is determined by $p_j \sim \text{random}(p_{j\min}, p_{j\max})$, $j = 1, 2, \ldots N$, with one generator as dependent generator.
3) Calculate the overall objective function if the UCP is given in (3) using the trail vector p_i, and find the minimum of F_{Ti}.
4) Initialize the parent vector $p = [p_1, p_2, \ldots p_n]$, $i = 1, 2, \ldots N_p$ such that each element in the vector is determined by $p_j \sim \text{random}(p_{j\min}, p_{j\max})$, $j = 1, 2, \ldots N$, with one generator as a dependent generator.
5) Calculate the overall objective function if the UCP is given in (3) using the trail vector p_i, and find the minimum of F_{Ti}.
6) Create the offspring trail solution p_i' using the following steps.
   (a) Calculate the standard deviation
   $$\sigma_j = \beta \left(\frac{F_{Ti}}{\min(F_{Ti})} \right) (P_{j\max} - P_{j\min}).$$
   (b) Add a Gaussian random variable N $(0, \sigma_j^2)$ to all of the state variable of p_i, to get p_i'.
7) Select the first N_p individuals from the total $2N_p$ individuals of both p_i and p_i' using the following steps for next iteration:
   (a) Evaluate $r = (2N_p \text{ random}(0, 1) + 1)$.
   (b) Evaluate each trail vector by $W_{pi} = \text{sum}(W_x)$ where $x = 1, 2, \ldots N_p$, $i = 1, 2, \ldots 2N_p$, such that $W_x = 1$ if $F_{Tij}/(F_{Tij} + F_{Tir}) <$ random(0,1); otherwise, $W_x = 0$.
8) Sort the W_{pi} in descending order and the first N_p individuals will survive and are transcribed along with their elements to form the basis of the next generation.
9) The above procedure is repeated from step (2) until a maximum number of generations N_m is reached.
10) Selection process is done using evolutionary strategy.

V. Evolutionary Programming-Based TS for UCP

A. Tabu Search

1) Take the parent as the initial feasible solution.
2) Take demand as control parameter and generate the trial solution.
3) Check for the stopping criterion.

4) If false, decrement system peak demand, and go to step 2.
5) If true, generate the optimal solution, and calculate the total operating cost.

3. EP-Based TS

In the TS technique for solving UCP, initial operating schedule-status in terms of maximum real power generation of each unit is given as input. As we know that TS is used to improve any given status by avoiding entrapment in local minima, the offspring obtained from the EP algorithm is given as input to TS, and the refined status is obtained. In addition, evolutionary strategy selects the final status.

1) Get the demand for 24 h and number of iterations to be carried out.
2) Generate population of parents (N) by *adjusting* the existing solution to the given demand to the form of state variables.
3) Unit downtime makes a random recommitment.
4) Check for constraint in the new schedule by TS. If the constraints are not met, then repair the schedule as given in section V-C.
5) Perform ELD and calculate total production cost for each parent.
6) Add the Gaussian random variable to each state variable and, hence, create an offspring. This will further undergo some repair operations as given in Section V-D. Following these, the new schedules are checked in order to verify that all constraints are met.
7) Improve the status of the evolved offspring, and verify the constraints by TS.
8) Formulate the rank for the entire population.
9) Select the best N number of population for next iteration.
10) Has the iteration count been reached? If yes, go to step 11, else go to step 2.
11) Select the best population (s) by evolutionary strategy [26], [27].
12) Print the optimum schedule.

The flowchart for EPTS for UCP is shown in Fig. 3.

C. Repair Mechanism

A repair mechanism to restore the feasibility of the constraints is applied and described as follows.

- Pick at random one of the OFF units at one of the violated hours.
- Apply the rules in Section III-C to switch the selected unit from OFF to ON keeping the feasibility of the downtime constraints.
- Check for the reserve constraints at this hour. Otherwise, repeat the process at the same hour for another unit.

D. Making Offspring Feasible

While solving the constrained optimization problem, there are various techniques to repair an infeasible solution [21], [26]. In this paper, we have chosen the technique, which evolves only the feasible solutions. That is, the schedule which satisfies the set of constraints as mentioned earlier. Here, in this paper, the selection routine is involved as "culling force" to eliminate the feasible schedules. Before the best solution is selected by evolutionary strategy, the trail is made to correct the unwanted mutations.

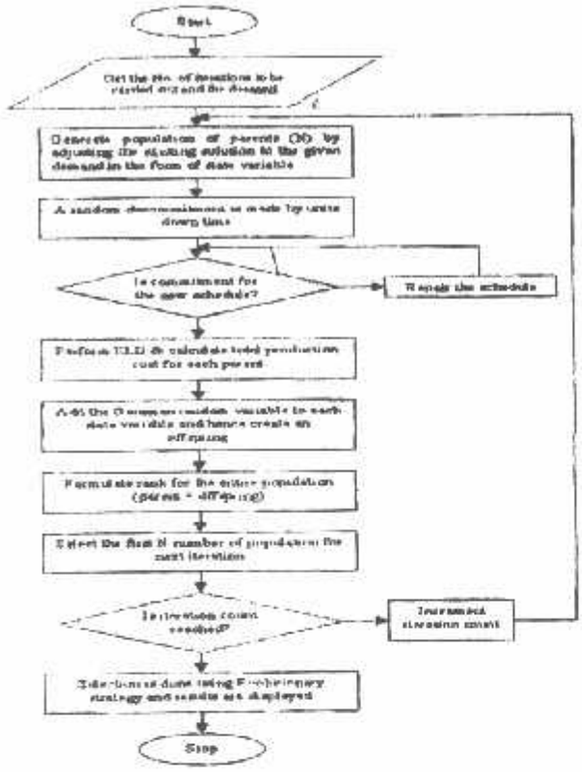


Fig. 3. Flowchart for EPTS for UCP.

E. Implementation

Software programs were developed using MATLAB software package, and the test problem was simulated for ten independent trials using LPA. The training and identification part as implemented in the tabu search technique is employed here and considered as a process involving random recommitment, constraint verification, and offspring creation.

VI. Numerical Results

A NTPS in India with seven generating units, each with a capacity of 210 MW, has been considered as a case study. A time period of 24 h is considered; the unit commitment problem is solved for these seven units and also compared with 10, 26, and 34 generating unit power systems.

The required inputs for solving the UCP are briefed here. The total number of generating units, the maximum real power generation of each unit, and the cost function parameters of each unit are tabulated for a day, respectively, as shown in Tables I and II for NTPS. The status of unit i at time t and the startup/shutdown status obtained are the necessary solution for SA, TS, EP, EPTS, DP, and LR methods for NTPS. The comparison of the total costs and CPU time is shown in Table III for NTPS, 10, 26, and 34 generating unit power systems. Fig. 4 represents the total production cost obtained

TABLE I
DAILY GENERATION OF SEVEN UNITS (IN MEGAWATTS)

Hour	Pmax	I	II	III	IV	V	VI	VII
1	840	60	80	100	101	149	150	200
2	757	60	0	100	100	147	150	200
3	775	60	0	100	115	150	150	200
4	773	60	0	100	113	150	150	200
5	770	60	0	100	110	150	150	200
6	778	60	0	100	118	150	150	200
7	757	60	0	100	100	147	150	200
8	778	60	0	100	118	150	150	200
9	770	60	0	100	110	150	150	200
10	764	60	0	100	104	150	150	200
11	598	60	0	99	97	142	0	200
12	595	60	0	100	96	139	0	200
13	545	0	0	100	99	146	0	200
14	538	0	0	99	97	142	0	200
15	535	0	0	100	96	139	0	200
16	466	0	0	0	116	150	0	200
17	449	0	0	0	101	148	0	200
18	439	0	0	0	97	142	0	200
19	466	0	0	0	116	150	0	200
20	463	0	0	0	113	150	0	200
21	460	0	0	0	110	150	0	200
22	434	0	0	0	95	139	0	200
23	530	60	0	0	120	150	0	200
24	840	60	80	100	101	149	150	200

TABLE II
GENERATION SYSTEM OPERATION DATA

Unit	P_{min} (MW)	P_{max} (MW)	Running Cost C_i (Rs)	Running Cost B_i (Rs/MWh)	Running Cost A_i (Rs/MWh²)	Start-up cost So_i (Rs)	Start-up cost D_i (Rs)	Start-up cost E_i (Rs)
1	15	60	750	70	0.255	4250	29.5	10
2	20	80	1250	75	0.198	5050	29.5	10
3	30	100	2000	70	0.198	5700	28.5	10
4	25	120	1600	70	0.191	4700	32.5	9
5	30	150	1450	75	0.106	5650	32	9
6	50	150	4950	65	0.0675	14100	37.5	4.5
7	75	200	4100	60	0.074	11350	32	5.5

by each parent for three iterations in EP method. Similarly, the total production cost is obtained for four and ten iterations. Fig. 5 gives the plot of EP average performance from 100 runs. Fig. 6 gives the plot of number of iteration versus the time taken to complete those iterations and the maximum production cost obtained under each iteration. From these results, the EPTS method had less total cost and consumed less CPU time in all of the power systems considered including NTPS.

In our proposed method, the EP method was used in conjunction with the tabu search method. As we indicated in the paper, the EP algorithm has also proved to be an efficient tool for solving the important economic dispatch problem for units with "nonsmooth" fuel cost functions as referred in [26]. Such functions may be included in the proposed EP search for practical problem solving. There is no obvious limitation on the size of the problem that must be addressed, for its data structure is such that the search space is reduced to a minimum; no "relaxation of constraints" is required; instead, populations of feasible solutions are produced at each generation and throughout the evolution process. The main advantages of the proposed algorithm are speed.

The proposed EPTS approach was compared to the related methods in the references indented to serve this purpose, such as the DP with a zoom feature, the SA, and the GA approaches. In addition, with the use of TS method, the status is improved by avoiding the entrapment in local minima. By means of stochastically searching multiple points at one time and considering trial solutions of successive generations, the EPTS approach avoids entrapping in local optimum solutions. Also, disadvantages of huge memory size required by the SA method are eliminated. Moreover, intellectual schemes of encoding and decoding entailed by the GA approach are not needed in the proposed EPTS approach. The problem of power unbalance previously existing in the solution of GA is circumvented as well in this paper. In comparison with the results produced by the referenced techniques, the EPTS method obviously displays a satisfactory performance with respect to the quality of its evolved solutions and to its computational requirements.

TABLE III
COMPARISONS OF COST AND CPU TIME FOR NTPS, 10-, 26-, AND 34-UNIT SYSTEMS

System	Methods	Total Cost (pu)	CPU Time (sec)
7 Unit (NTPS)	SA	0.94780	84
	TS	0.95531	110
	EP	0.93461	66
	EPTS	0.92609	65
	DP	1.00000	130
	LR	0.97483	115
10 unit	SA	0.94422	200
	TS	0.95322	230
	EP	0.94071	186
	EPTS	0.92349	183
	DP	1.00000	260
	LR	0.97183	235
26 Unit	SA	0.93906	1810
	TS	0.94887	1820
	EP	0.92814	1785
	EPTS	0.91136	1780
	DP	1.00000	1860
	LR	0.96642	1878
34 Unit	SA	0.93500	6800
	TS	0.94382	6842
	EP	0.92400	6792
	EPTS	0.90918	6780
	DP	1.00000	6865
	LR	0.96197	6824

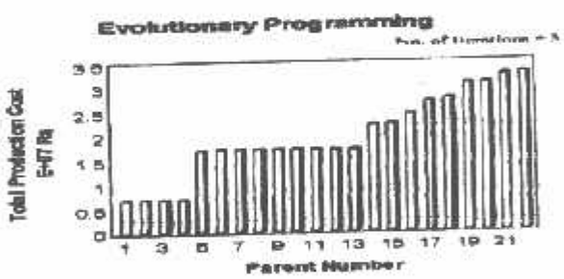


Fig. 4. Total production cost for three iterations.

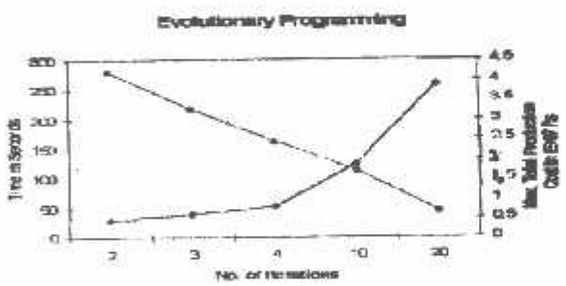


Fig. 6. Number of iterations versus time taken and maximum product on cost.

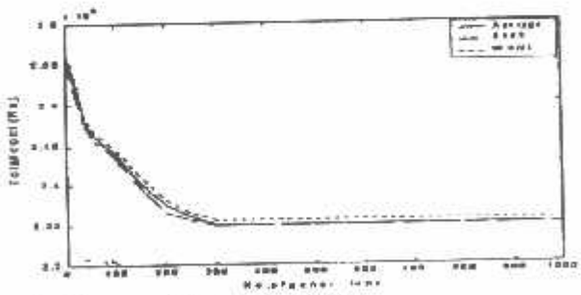


Fig. 5. EP average performance from 100 runs.

VII. CONCLUSION

This paper presents an EP-based tabu search method to the unit commitment problem. In this method, the essential processes simulated in the procedure are mutation, competition, and selection. The mutation rate is computed as a function of the ratio of the total cost by the schedule of interest to the cost of the best schedule in the current population. Competition and selection are applied to select from among the parents and the offspring, the best solutions to form the basis of the subsequent generation. In this proposed work, the parents are obtained from a predefined set of solutions (i.e., each and every solution is obtained from the TS method). Then, a random recommitment is carried out with respect to the unit's minimum downtimes, and the selection process is done using evolutionary strategy.

In comparison with the results produced by the referenced techniques (EP, DP, LR, and SA and TS), the EPTS method obviously displays satisfactory performance. There is no obvious limitation on the size of the problem that must be addressed, for its data structure is such that the search space is reduced