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

PENJADWALAN PERAWATAN PADA SISTEM TENAGA DI PLTUG SEKTOR GRESIK DENGAN MENGGUNAKAN METODE GENETIK ALGORITMA

SKRIPSI

Disusun Oleh :

HENRY ARDHIAN AKLISGANDHI
NIM : 99.12.082

APRIL 2005

بسم الله الرحمن الرحيم

Terima kasih juga penulis ucapkan kepada keluarga besar Rully Pramadhana (my little brother ketreng), keluarga besar Mukti Wibowo (my best friend in sad and happy), Micronik CS Mania Multiplayer (temanku perang)

I'll remember *that you very special person that i have all this time* ***Someone*** *thanks for your* ***spirit.*** *You bring me a mean about true love, which mean it is a best gift from God that you can't realize before but you can feel it in your heart...deepest heart.*

Thanks to my favorite computer (*Momok Tet*) i can't live without you by my side, my cell phone (N-GAGE; i can't buy anything without you by my hand), My motorcycle.

My college friends, **Erman (Kerman) friend forever,**the three Lion, Wahyu (bom-bom N-dhut), Mukti Wibowo (get some chick dude go for it !!), Herman, thank to " Tumpah di Dalam " for the most moron joke, bom-bom home stay (suite room), Arif S.B, Rully (Ketreng), Boby (Kevin) I'll wait for your opportunity across ocean, Bang Thamrin (I hope we will see you in your own workland), Ardian, Rahmad, Rendra, Acong (Rendra), Agus, Jirin (never say enough, be a man, never let her play your heart), Joko S, Memed, Aris (Bpk), Rahmat (Blimbing), Farit (Lawang), Yudi, *Diana*, Ayu; Klampok suite place, Tambor place, my brother Teguh (sogol), Aris (Slorok), Degan, Pam, Farid (Paidi), Kristian (Blaky), Aris (Gemboz), Ubed. Dan seluruh teman-teman Elektro S.T'99 lainnya yang belum disebutkan satu persatu, sekali lagi terima kasih penulis ucapkan. *I dedicated this paper for the most i* ***love*** *.....all of you.*

If One Day .....
If one day you feel like crying...Call me.
I don't promise that i'll make you laught, but i can cry with u.

If one day you want to run away....don't be afraid to call me
I don't promise to ask you to stop, but i can run with you

If one day you don't want to listen to anyone...call me
I promise to be there for you. And i promise to be very quite

But if one day you call.....
And there is no answer.....
Come fast to see me.
Perhaps i need you.....☺

created by : Henry Ardhian A.
e-Mail: momok_tet_w@yahoo.com.

KATA PENGANTAR

Syukur *Alhamdulillaahi Robbil'aalamiin* yang pertama penulis ucapkan Kehadlirat *ALLOH S.W.T* yang telah memberikan *Taufik, Hidayah* serta *Inayah-Nya* kepada kita semua dan khususnya kepada penulis sehingga terselesainya Skripsi ini dengan baik, serta tidak lupa saya panjatkan syukur kepada *Baginda Rosululloh Muhammad S.A.W* yang telah membawa kita dari *Zaman Jahiliyah* ke *Zaman yang Terang Benderang.* Adapun Skripsi ini dengan judul **"Penjadwalan Perawatan Pada Sistem Tenaga di PLTUG Gresik Dengan Menggunakan Metode Algoritma Genetika"**.

Ucapan terima kasih yang sebesar – besarnya tak lupa penulis sampaikan kepada :

1. Bapak Dr.Ir. Abraham Lomi, MSEE selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. Mochtar Asroni, MSME, selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang.
3. Bapak Ir. F. Yudi Limpraptono, MT, selaku Ketua Jurusan Teknik Elektro Institut Teknologi Nasional Malang.
4. Bapak Ir. H. Almizan Abdullah, MSEE selaku Dosen Pembimbing pertama yang telah banyak memberikan bimbingan, saran dan pemikiran serta supportnya sehingga terselesaikannya skripsi ini.

5. Ayanda dan Ibunda tercintaku, Mbak Yus dan Cak Pat, Dik Lugti dan keponakanku Hane, terimakasih atas dukungan Moral dan Spiritualnya selama ini.
6. Keluarga besar Rully Pramadhana, Keluarga besar Mukti Wibowo, terima kasih atas segala perhatian serta dukungan terhadap penulis selama ini.
7. Temen – temen *Micronic CS* pada umumnya, Bowo (Owob), Dodik (DW), Mery (Kadal), Pendik (Pentil), Koko (Ngingoeng), Aris (KJ), Bang Hasan (YeYe), Apip (Picolo) dan CS Mania lainnya terima kasih atas supportnya.
8. Serta rekan-rekan Mahasiswa Elektro ITN pada umumnya dan Elektro Energi Listrik pada khususnya serta teman-teman semua.

Pada akhirnya penulis mengharapkan kritik dan saran dari pembaca demi kesempurnaan dari skripsi ini, karena penulis menyadari masih banyak kekurangan di dalam skripsi ini. Dan penulis berharap semoga skripsi ini dapat bermanfaat di kemudian hari.Amin

Malang, Maret 2005

Penulis

ABSTRAKSI

" Penjadwalan Perawatan Pada Sistem Tenaga di PLTUG Gresik Dengan Menggunakan Metode Algoritma Genetika"

(Henry Ardhian Aklisgandhi, 99.12.082, TEKNIK ENERGI LISTRIK, 53 halaman)

(Dosen Pembimbing : Ir. H. Almizan Abdullah, MSEE)

Kata kunci : *Maintenace, scheduling, power system, hydrothermal, genetic algorithm*

Dalam penjadwalan perawatan yang dibahas skripsi ini adalah menentukan periode kerja perawatan pada unit tenaga sehingga dapat mempertahankan cadangan pada level tertentu. Dimana kompleksitas masalah penjadwalan perawatan meningkat secara dramatis dengan bertambahnya jumlah unit.

Id; dasar yang melatari *Genetic Algoritm* adalah menjalankan komputer supaya melakukan seperti yang dilakukan oleh alam. Sebagai ilmuwan komputer Holland berkonsentrasi pada algoritma yang memanipulasi string digit biner. Dia melihat bahwa algoritma ini adalah suatu bentuk abstrak dari revolusi alam. Algoritma Holland dapat disajikan dengan serangkaian langkah prosedural untuk memindahkan dari satu populasi kromosom tiruan ke populasi baru. Ini menggunakan seleksi alam dan teknik yang diilhami dari genetik yang dikenal sebagai mutasi dan *crossover*.

DAFTAR ISI

HALAMAN JUDUL i

LEMBAR PERSETUJUAN ii

KATA PENGANTAR iii

ABSTRAKSI v

DAFTAR ISI vi

DAFTAR GAMBAR ix

DAFTAR TABEL xi

BAB I PENDAHULUAN

1.1. Latar Belakang 1

1.2. Rumusan Masalah 4

1.3. Tujuan Pembahasan 4

1.4. Batasan Masalah 5

1.5. Metodologi Penulisan 5

1.6. Sistematika Penulisan 6

1.7. Konstribusi Penulisan 6

BAB II TEORI DASAR

2.1. Teori Penjadwalan Perawatan Pada Sistem Tenaga 7

2.1.1. Pendahuluan 7

2.1.2. Jadwal Perawatan Pada Sistem Tenaga 9

2.1.3. Latar Belakang Penggunaan Program Komputer Pada Penjadwalan Perawatan 12

2.2. Teori Algoritma Genetika 13

2.2.1. Pendahuluan 13

2.2.2. Perkembangan Algoritma Genetika Untuk Penjadwalan Pemeliharaan 18

BAB III PLTGU UNIT PEMBANGKITAN GRESIK

3.1. Sejarah PLTGU Gresik 25

3.2. Konfigurasi dan Kapasitas Unit Pembangkitan Gresik 26

3.3. Komponen-Komponen Utama Yang Terdapat di Unit Pembangkitan Gresik 28

3.4. Data Jurnal Yang Dijadikan Acuan 32

3.5. Data Input Real 33

BAB IV ANALISA DATA DENGAN METODE GA PADA PT. PJB

4.1. Program Komputer Metode GA 35

4.2. Algoritma Program 35

4.3. Flowchart Program 37

4.4. Tampilan Program 38

4.4.1. Program Validasi Program Komputer 38

4.4.2. Program Real 40

4.4.3. Perbandingan Program Validasi Program Komputer Dengan Program Real 50

4.4.4. Perbandingan Antara Hasil Program Dengan Data PLN 51

4.4.5. Validasi program komputer terhadap program jurnal 52

BAB V PENUTUP

5.1. Kesimpulan 53

5.2. Saran 53

DAFTAR PUSTAKA

LAMPIRAN

DAFTAR GAMBAR

Gambar 2.1. Umur ekonomis peralatan 9

2.2. Diagram kesiapan peralatan dalam satu tahun (8760 jam) 11

2.3. 16-bit binary string kromosom tiruan 13

2.4. Seleksi roda roulete 16

2.5. Operasi crossover 17

2.6. Operasi mutasi 18

3.1. Rencana perawatan UP Gresik tahun 2004 34

4.1. Flowchart program 37

4.2. Tampilan utama 38

4.3. Menu Tampilan Data 39

4.4. Tampilan Program 39

4.5. Tampilan program utama 40

4.6. Menu tampilan data pada data utama 41

4.7. Tampilan data maintenance 42

4.8. Tampilan data pembangkit 43

4.9. Tampilan data beban 44

4.10. Tampilan data parameter Genetic Algoritm 45

4.11. Proses GA 46

4.12. Fitness minimum, rata-rata, dan maksimum 47

4.13. Grafik fitness Genetic Algorithm.....48

4.14. Jadwal maintenance tiap unit, kapasitas yang tersedia
dan pelanggaran maintenance 49
4.15. Grafik fitness Genetic Algorithm (validasi) 50
4.15. Grafik fitness Genetic Algorithm (real) 50

DAFTAR TABEL

Tabel 2.1. Data unit dan kebutuhan perawatan 19

2.2. Unit kumpulan gen 22

3.1. Tipe pembangkit di unit pembangkitan Gresik 26

3.2. Data unit dan kebutuhan perawatan di jurnal 32

3.3. Data unit dan kebutuhan perawatan 33

4.1. Perbandingan kapasitas tersedia PT. PJB dengan metode GA 51

4.2. Validasi 52

4.3. Jurnal 52

BAB I

PENDAHULUAN

1.1. Latar Belakang

Tugas perencanaan memainkan peran utama dalam menjamin pengoperasian yang aman dan ekonomis dari sistem daya modern. Tugas-tugas ini dibagi dalam dua kategori : perencanaan sistem untuk periode waktu yang sangat panjang dan perencanaan pengoperasian untuk periode yang lebih pendek dengan cakupan dari beberapa menit sampai beberapa tahun. Bila perencanaan sistem dikaitkan dengan investasi jangka panjang dalam sistem tenaga dan dihubungkan dengan pemasangan unit pembangkit baru dan pembangunan saluran transmisi daya yang baru, maka perencanaan operasi melaksanakan penjadwalan perawatan unit tenaga, komitmen unit dan ekonomi dispatch. Penjadwalan perawatan, yang dibahas dalam skripsi ini, menentukan periode kerja perawatan pada unit tenaga sehingga dapat mempertahankan cadangan pada level tertentu[1]. Penjadwalan perawatan memerlukan pertimbangan beberapa faktor yang berbeda seperti ketersediaan kru perawatan, keperluan spesifik untuk order perawatan unit, dan sejarah perawatan yang lalu. Tugas penjadwalan perawatan dapat dirumuskan secara matematiks sebagai masalah pemrograman integer atau integer campuran, dan kemudian diperkecil untuk solusi masalah optimasi kombinasi. Ini berarti bahwa kompleksitas masalah penjadwalan maintenance meningkat secara dramatis dengan bertambahnya jumlah unit.

Bertambah pentingnya penjadwalan perawatan dalam sistem tenaga didasarkan pada tingginya biaya perawatan peralatan dan kebutuhan untuk meningkatkan keamanan sistem. Unit berkapasitas besar yang digunakan pada sistem daya modern menghasilkan kemungkinan kehilangan daya pembangkitan yang besar dengan keluarnya sebuah unit (outage tunggal). Sebagai hasilnya, sistem daya harus mempunyai cadangan yang lebih tinggi. Dengan demikian pengoperasian unit yang efektif memungkinkan sistem tenaga untuk mencapai penghematan yang subtansial. Tetapi, pengoperasian seperti itu harus diselesaikan dibawah sejumlah keadaan yang tidak pasti dan tidak dapat diprediksi. Hal ini merangsang suatu minat yang besar pada perkembangan prosedur penjadwalan perawatan secara otomatis. Metode yang ada yang digunakan untuk penjadwalan perawatan meliputi :

1. *Dynamic programming*. Metode ini menggusahakan untuk menggunakan objek yang berbeda seperti cadangan bersih minimum dan meminimalkan resiko kekurangan permintaan daya[2,3]. Dynamic programming mempunyai beban dimensionalitas untuk masalah skala besar seperti masalah penjadwalan maintenance.
2. *Integer programming and mixed integer programing*. Metode ini memeberlakukan beberapa kendala artifisial seperti keperluan untuk merawat unit hanya sekali selama periode waktu yang diamati[4,5]. Sebagai tambahan, pemrograman integer membuat masalah penjadwalan ini sulit untuk menyajikan kendala komplek yang ada dalam masalah penjadwalan perawatan yang riil.

3. *Rule-based and frame-based expert systems.* Metode Branch dan Bound[5] yang digabungkan dengan pengetahuan heuristic telah terbukti efektif[6], tetapi expert systems masih mempunyai kesulitan untuk diatasi. Rintangan utama disini adalah akuisisi pengetahuan , validasi, dan pengecekan konsistensi. Disamping itu expert systems tidak dapat menjamin jadwal perawatan yang optimal.
4. *Artificial neural network.* Disamping fakta bahwa jaringan syaraf tiruan saat ini jauh dari tujuan untuk meniru fungsi otak manusia, ANN banyak digunakan pada aplikasi praktis dalam sistem tenaga termasuk tugas penjadwalan perawatan[7].

Skripsi ini menyajikan pendekatan alternative untuk pendjadwalan perawatan dalam sistem tenaga bersandar pada Algoritma Genetika (*Genetic Algorithms).*

1.2. Rumusan Masalah

Bertambah pentingnya penjadwalan perawatan pada sistem tenaga didasarkan pada kenyataan akan tingginya biaya perawatan peralatan dan kebutuhan untuk meningkatkan keandalan sistem. Untuk itu diperlukan penentuan penjadwalan perawatan pada unit pembangkit yang andal. Salah satu metode yang dapat dipakai adalah Metode Algoritma Genetik.

Berdasarkan gambaran permasalahan tersebut maka skripsi ini diberi judul :

"PENJADWALAN PERAWATAN PADA SISITEM TENAGA DI PT. PEMBANGKIT JAWA BALI DENGAN MENGGUNAKAN METODE GENETIK ALGORITMA"

1.3. Tujuan Pembahasan

Tujuan dari penulisan skripsi ini adalah untuk :

1. Menentukan periode kerja perawatan pada unit tenaga sehingga dapat mempertahankan cadangan pada level tertentu[1].
2. Meningkatkan kinerja dari unit-unit pembangkit dengan penjadwalan perawatan yang tepat.

1.4. Batasan Masalah

Untuk mencegah meluasnya masalah yang timbul kearah yang tidak relevan, perlu kiranya diberi batasan masalah. Pembatasan masalah dibatasi sebagai berikut :

1. Sistem yang ditinjau adalah jadwal perawatan pada unit-unit pembangkit yang ada di PJB UP GRESIK.
2. Waktu penjadwalan adalah berapa kali dalam setahun unit mengalami perawatan dan harus dihentikan pengoperasiannya.
3. Metode yang digunakan hanya untuk penjadwalan perawatan pada unit-unit pembangkit.
4. Dalam pendekatan penjadwalan perawatan unit-unit pembangkit hanya menggunakan *Genetic Algorithm*.
5. Tidak membahas tentang analisa ekonomi.

1.5. Metodologi Penulisan

Agar didapat maksud dan tujuan pembahasan, maka metodologi yang digunakan :

1. Studi literatur, yaitu dengan mempelajari buku-buku yang berkaitan denagan masalah GA dan penjadwalan perawatan.
2. Pengambilan dan pengumpulan data-data, dengan melihat dari dekat keadaan yang sebenarnya tentang penjadwalan perawatan pada unit-unit pembangkit yang ada di PJB UP GRESIK.
3. Menyimpulkan permasalahan penjadwalan perawatan.

1.6. Sistematika Penulisan

Bab I : Memberikan gambaran umum mengenai latar belakang, rumusan masalah, tujuan pembahasan, batasan masalah, metodologi penulisan, sistematika penulisan, dan kontribusi.

Bab II : Membahas teori dasar.

Bab III : Membahas tentang PJB UP Gresik serta data-data yang diperlukan untuk menyelesaikan permasalahan yang ada dan alternatif pemecahan yang mungkin dapat diterapkan di lapangan. Dicantumkan juga data validasi program komputer.

Bab IV : Melakukan analisa optimasi dan solusi yang diambil serta evaluasinya dengan metode Genetik Algoritma.

Bab V : Kesimpulan.

1.7. Kontribusi Penulisan

1. Mendapatkan program optimalisasi penjadwalan perawatan unit pembangkit pada PLTGU Gresik.
2. Dapat dipakai masukan oleh PT. PLN untuk memastikan penjadwalan perawatan pada unit-unit pembangkit.

BAB II

TEORI DASAR

2.1. Teori Penjadwalan Perawatan Pada Sistem Tenaga

2.1.1. Pendahuluan

Perawatan peralatan sangat diperlukan untuk mempertahankan unjuk kerja peralatan. Di lain pihak perawatan peralatan sistem tenaga listrik sebagian besar memerlukan pembebasan tegangan yang berarti bahwa peralatan yang dipelihara harus dikeluarkan dari operasi (tidak beroperasi). Hal ini menyebabkan berkurangnya kemampuan penyediaan daya dari sistem tenaga listrik terutama apabila yang dirawat adalah unit pembangkit, jadi mengurangi keandalan sistem. Oleh karenanya perawatan peralatan dalam sistem tenaga listrik, khususnya unit pembangkit perlu dikoordinir melalui suatu jadwal perawatan, agar syarat perawatan peralatan dan syarat keandalan sistem kedua-duanya dapat dipenuhi.

Masalah yang unik dalam operasi sistem tenaga listrik adalah bahwa :
Daya yang dibangkitkan/diproduksi harus selalu sama dengan daya yang dikonsumsi oleh para pemakai tenaga listrik yang secara teknis umumnya dikatakan sebagai beban sistem.
Apabila daya yang dibangkitkan lebih kecil daripada beban sistem maka frekwensi dan tegangan akan turun, sebaliknya apabila lebih besar maka maka frekwensi dan tegangan akan naik. Mutu listrik yang baik adalah apabila frekwensi dan tegangan tidak terlalu jauh menyimpang dari nilai nominal, untuk ini haruslah diusahakan agar daya yang dibangkitkan selalu sama dengan beban.

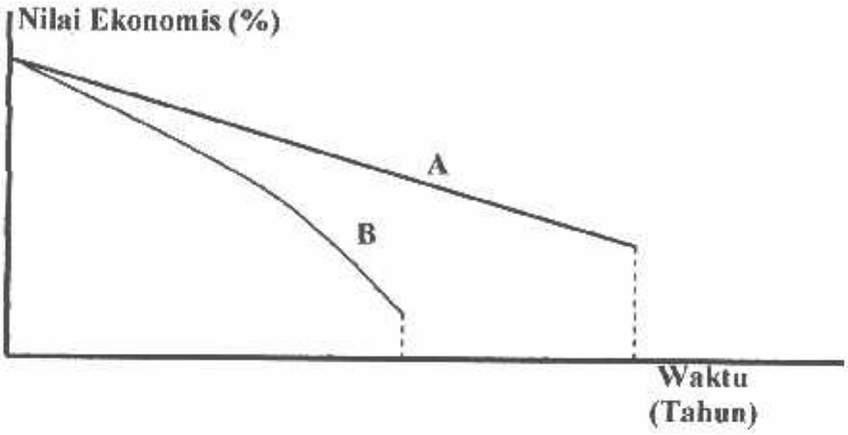
Daya yang dibangkitkan maupun beban terdiri dari daya nyata MW dan daya reaktif MVAR. Daya nyata hubungannya adalah dengan frekwensi sedangkan daya reaktif hubungannya dengan tegangan. Dalam keadaan steady state frekwensi adalah sama dalam seluruh sistem sehingga pengaturan frekwensi dapat dilakukan dengan pengaturan daya nyata MW. Tetapi tidak demikian halnya dengan masalah pengaturan tegangan karena pengaturan tegangan tergantung pada sumber daya reaktif MVAR setempat.

Dilain pihak besarnya beban sistem yang harus dilayani tidaklah konstan besarnya melainkan selalu berubah besarnya sepanjang waktu tergantung kepada keperluan para pemakai tenaga listrik. Tidak ada rumus yang eksak yang dapat memastikan besarnya beban besarnya beban untuk setiap saat, melainkan yang dapat dilakukan hanyalah memperkirakan besarnya beban dengan melihat angka statistik serta mengadakan analisa beban.

Unit pembangkit serta peralatan listrik lainnya yang dipergunakan untuk penyediaan tenaga listrik harus secara rutin dirawat sesuai buku instruksi pemeliharaan dari pabrik. Perawatan ini perlu dikoordinir agar unit pembangkit beserta peralatan lainnya yang tidak dirawat, yang siap operasi, masih cukup untuk dapat memikul beban sistem. Dalam kenyataannya unit pembangkit yang beroperasi banyak yang mengalami gangguan dan terpakasa dikeluarkan dari operasi, untuk menghadapi kemungkinan ini perlu diperhitungkan/direncanakan adanya unit pembangkit cadangan. Besarnya daya pembangkitan cadangan beserta jumlah unitnya sesungguhnya merupakan ukuran keandalan penyediaan daya dalam sistem.

2.1.2. Jadwal Perawatan Pada Sistem Tenaga

Peralatan dalam Sistem Tenaga Listrik perlu dirawat secara periodik sesuai petunjuk dari buku perawatan peralatan yang dibuat oleh pabriknya. Penundaan perawatan akan memperbesar kemungkinan rusaknya peralatan oleh karenanya jadwal perawatan sedapat mungkin harus ditaati.



Gambar 2.1.
Umur Ekonomis Peralatan

Apabila peralatan dirawat menurut jadwal perawatan maka nilai ekonomisnya akan turun menurut garis A pada gambar 2.1, tapi apabila perawatannya sering tertunda maka nilai ekonomisnya akan lebih cepat menurun misalnya menuruti garis B.

Dilain pihak untuk menjamin keandalan Sistem Tenaga Listrik, perawatan peralatan perlu dikoordinir agar tetap tersedia cadangan pembangkitan yang cukup dan juga tidak terdapat bagian dari sistem yang menjadi overloaded. Dalam sistem yang terdiri dari ratusan peralatan maka masalah koordinasi perawatan peralatan ini menjadi sulit dan sebaiknya tersedia fasilitas komputer untuk menanganinya.

Koordinasi perawatan peralatan yang kurang baik juga akan memperbesar biaya operasi oleh karenanya untuk sistem yang besar adalah memadai menggunakan komputer untuk menentukan jadwal perawatan yang optimum.

Secara garis besar bagian-bagian dari peralatan yang beroperasi dalam sistem yang perlu pemeriksaan dan perawatan periodik adalah :

a. Bantalan-bantalan yang menyangga poros yang berputar dengan kecepatan tinggi.

b. Bagian-bagian mekanik yang bergeser satu sama lain seperti piston ring pada mesin diesel dan kompresor.

c. Ruang baker dan bagian-bagian yang dilalui gas-gas hasil pembakaran.

d. Bagian-bagian yang memepertemukan dua zat yang tinggi perbedaan suhunya seperti alat-alat pemanas maupun pendingin.

e. Bejana-bejana yang bertekanan tinggi dan packing-packing tekanan tinggi serta katup-katup pengaman.

f. Bagian-bagian yang bertugas/bersifat mengumpulkan kotoran, seperti saringan dan kolam tando atau kolam pengendap pada PLTA.

g. Kontak-kontak listrik yang bergerak (moving contacts).

h. Sistem pentanahan peralatan listrik, khususnya yang terletak di luar gedung, misalnya sistem pentanahan Saluran Udara Tegangan Tinggi (SUTT).

i. Alat-alat ukur dan alat-alat pengaman perlu ditera secara periodik.

j. Isolasi peralatan perlu diukur secara periodik.

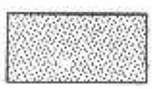
k. Bagian mekanik yang dilalui cairan atau gas yang berkecepatan tinggi seperti pengabut atau sudu jalan turbin.

l. Sambungan-sambungan listrik khususnya apabila terletak di tempat yang banyak bergetar dan / atau banyak mengalami perubahan suhu.

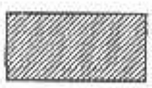
m. Katup-katup mekanik untuk cairan maupun gas/uap dan mekanisme switch gear.

Perawatan yang teratur selain memperpanjang umur ekonomis peralatan juga memepertinggi keandalan peralatan. Dengan memperhatikan gambar 2.2. maka perawatan peralatan dapat memeperkecil Forced Outage Hours yang berarti peralatan dapat lebih diandalkan bagi kepentingan operasi.

Jam-jam peralatan dalam keadaan siap (Operating Availabie Hours)



Jam-jam peralatan dalam pemeliharaan yang direncanakan (Planned Outage Hours)



Jam-jam peralatan dalam keadaan gangguan (Forced Outage Hours)

Gambar 2.2. Diagram kesiapan peralatan dalam satu tahun (8760 jam)

2.1.3. Latar Belakang Penggunaan Program Komputer Pada Penjadwalan Perawatan

Keandalan operasi, biaya produksi dan pengeluaran modal dalam sebuah sistem tenaga dipengaruhi oleh penghentian operasi unit pembangkit (outage) untuk perawatan. Disamping itu biaya operasi langsung untuk perawatan semakin meningkat akibat semakin kompleknya peralatan. Dengan makin meningkatnya jumlah dan kerumitan unit-unit, dipertanyakan apakah metode manual murni untuk penjadwalan perawatan dapat menjamin jadwal yang mendekati optimumnya.

Aspek lainnya adalah meningkatnya volume kertas kerja yang diperlukan untuk menabulasi tanggal pemulaian perawatan, kapasitas cadangan mingguan, kebutuhan tenaga kerja, kerja yang selesai dan kerja yang sedang berjalan. Juga timbul pertanyaan apakah laporan yang dipersiapkan secara manual dapat sejalan dengan sebuah penjadwalan perawatan yang secara berkelanjutan berubah dalam suatu upaya untuk menyesuaikan diri dengan unit pembangkit baru yang tidak beroperasi karena terpaksa (forced outage), penundaan dalam pemasangan kapasitas baru, masalah tenaga kerja dan keterbatasan suku cadang.

Kunci masalah lainnya dalam sistem dimana cadangan sangat ketat adalah suatu kebutuhan untuk meminimasi pembelian kapasitas yang mahal untuk menutup biaya perawatan dan unit pembangkit yang tidak beroperasi karena terpaksa.

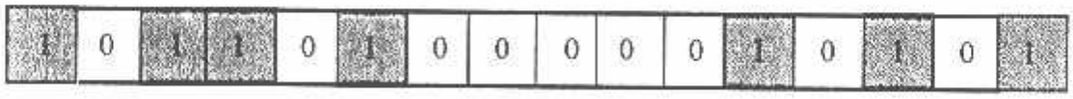
Dalam hal ini khususnya penggunaan program komputer untuk penjadwalan perawatan, dimana teknik komputerisasi dapat menyediakan kualitas

penjadwalan yang tinggi, menyediakan cara untuk menentukan cepat tidaknya penanganan dalam kaitannya dengan masalah operasi tak terduga dan dapat mengurangi kertas kerja tabulasi dan pemetaan data perawatan sistem.

2.2 Teori Algoritma Genetika (Genetic Algorithm)

2.2.1. Pendahuluan

Pada awal 1970-an, John Holland , salah satu dari penemu evolusi komputasi, memperkenalkan konsep algoritma genetika (GA)[8]. Ide dasar yang melatarbelakangi GA adalah menjalankan komputer supaya melakukan seperti yang dilakukan oleh alam. Sebagai ilmuwan komputer, Holland berkonsentrasi pada algoritma yang memanipulasi string digit biner. Dia melihat bahwa algoritma ini sebagai suatu bentuk abstrak dari evolusi alam. Algoritma Holland dapat disajikan dengan serangkaian langkah prosedural untuk memindahkan dari satu populasi "kromosom" tiruan ke populasi baru. Ini menggunakan seleksi "alam" dan teknik yang diilhami dari genetik yang dikenal sebagai mutasi dan crossover. Tiap kromosom terdiri dari sejumlah "gen", dan tiap gen dilambangkan dengan 0 dan 1, seperti terlihat pada gambar 2.3.



Gambar 2.3.
16-bit binary string kromosom tiruan

Alam mempunyai kemampuan untuk beradaptasi dan belajar tanpa harus diberitahu apa yang harus dilakukan. Dengan kata lain, alam mempunyai kromosom yang bagus secara tidak sengaja (secara acak). GA melakukan hal yang

sama. Ada dua mekanisme dasar yang terkait dengan GA dengan masalah yang diselesaikannya : encoding dan evaluasi.

Pada kerja Holland, encoding diselesaikan dengan menggunakan kromosom yang dilambangkan dengan string 1 dan 0. walaupun beberapa tipe lain dari encoding telah ditemukan[9], tidak ada yang bekerja paling baik untuk semua masalah. Pada skripsi ini digunakan string bit seperti teknik yang popular pada umumnya.

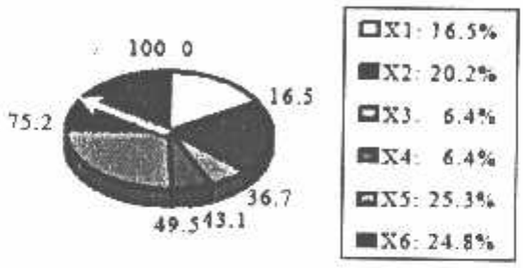
Fungsi evaluasi digunakan untuk mengukur performa kromosom dan fitness, pada masalah yang diselesaikan. Fungsi evolusi dalam GA memainkan peranan yang sama seperti peranan yang dimainkan lingkungan dalam evolusi alam. GA menggunakan ukuran fitness kromosom individual untuk menyelesaikan reproduksi. Ketika reproduksi berlangsung, operator crossover mengubah bagian dari dua kromosom tunggal, dan operator mutasi mengubah nilai gen pada lokasi kromosom yang dipilih secara random. Sebagai hasilnya, setelah sejumlah reproduksi yang terjadi secara berurutan, kromosom yang kurang layak mati, sedang yang terbaik dapat survive secara sedikit demi sedikit mendominasi populasi. Ini merupakan pendekatan sederhana, sekalipun begitu mekanisme reproduksi kasar memperlihatkan perilaku yang sangat kompleks dan kemampuan menyelesaikan beberapa masalah sulit.

Penentuan masalah yang sudah terdefinisi secara jelas dan penyajian string biner untuk bakal solusi, GA dasar dapat diwakili dengan langkah-langkah sebagai berikut[9,10] :

1. **Langkah 1**: melambangkan domain variable masalah sebagai kromosom dengan panjang tetap, memilih ukuran populasi kromosom N, kemungkinan kromosom p_c dan kemungkinan mutasi p_m.
2. **Langkah 2** : mendefinisikan fungsi fitness untuk mengukur performa atau fitness kromosom individu dalam domain masalah. Fungsi fitness membangun dasar untuk penyelesaian kromosom yang akan dikawinkan bersama selama reproduksi.
3. **Langkah 3**: secara acak menghasilkan populasi awal kromosom dengan ukuran N: $x_1, x_2, \ldots, x_N$.
4. **Langkah 4**: hitung fitness tiap kromosom individu $f(x_1), f(x_2), \ldots, f(x_N)$.
5. **Langkah 5**: pilih pasangan kromosom untuk perkawinan dari populasi saat ini. Kromosom induk dipilih dengan probabilitas yang tinggi dipilh untuk perkawinan dari pada kromosom yang kurang layak.
6. **Langkah 6**: buat pasangan kromosom keturunan dengan menggunakan operator genetic crossover dan mutasi.
7. **Langkah 7**: letakkan kromosom keturunan yang terbentuk dalam populasi yang baru.
8. **Langkah 8**: ulangi langkah 5 sampai ukuran populasi kromosm baru menjadi sama dengan ukuran populasi awal, N.
9. **Langkah 9**: ganti populasi kromosom awal (induk) dengan populasi baru (keturunan).
10. **Langkah 10** : kembali ke langkah 4 dan ulangi proses sampai kriteria terminasi (penghentian) terpenuhi.

GA menunjukkan proses iterasi (perulangan). Tiap perulangan disebut generasi. Sejumlah generasi yang khas untuk GA sederhana dapat bervariasi dari 50 sampai 500[10]. seluruh perangkat generasi disebut *run*. Pada akhir run, kami mengharapkan menemukan satu atau lebih kromosom dengan fitness tinggi.

Algoritma genetic menggunakan tiga operator genetik : seleksi, crossover, dan mutasi.

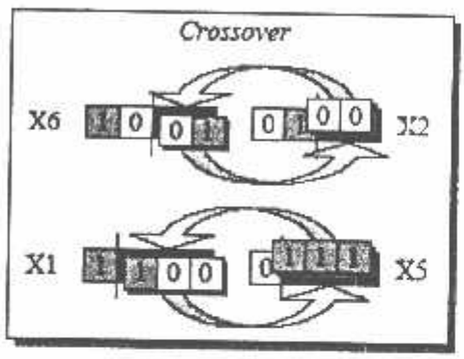


Gambar 2.4.
Seleksi Roda Rolet

Teknik yang paling popular untuk seleksi kromosom adalah seleksi roda rolet[9,10]. Gambar 2.4. menggambarkan roda rolet untuk populasi 7 kromosom. Masing-masing kromosom diperkenalkan dalam gambar 2.4. diberi suatu irisan dari suatu lingkaran roda rolet. Area lingkaran dalam roda adaiah sebanding dengan ratio fitness kromosom. Pemilihan kromosom untuk perkawinan, angka random dihasilkan dalam interval [0, 100], dan kromosom yang terpilih dengan segmen memutar angka random. Seperti memutar roda rolet dimana tiap kromosom mempunyai segmen pada roda proporsional terhadap fitnessnya. Roda rolet berputar dan ketika panah menunjukkan salah satu segment, kromosom yang bersangkutan terpilih.

Begitu sepasang kromosom dipilih, operator crossover diaplikasikan. Pertama, operator crossover memilih secara random titik crossover dimana dua induk

crossover "dipatahkan", dan kemudian merubah bagian kromosom setelah titik crossover. Sebagai hasilnya dua keturunan terbentuk. Sebagai contoh, cromosom X6 dan X2 dapat dicrossover-kan setelah gen kedua pada tiap kromosom memproduksi dua keturunan, seperti yang ditunjukkan pada gambar 2.5. Nilai 0.7 untuk probabilitas secara umum memproduksi hasil yang bagus.



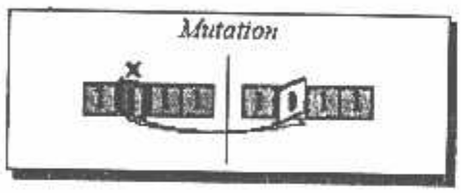
Gambar 2.5.
Operasi Crossover

Operasi genetik yang terakhir adalah mutasi. Mutasi sangat jarang terjadi di alam mewakili perubahan dalam gen. Holland mengenalkan mutasi sebagai operator background[8]. Peraturannya adalah menyediakan jaminan bahwa pencarian algoritma tidak terjebak pada optimum lokal. Serangkaian operasi seleksi dan crossover mungkin berhenti pada beberapa seperangkat solusi yang homogen. Dibawah kondisi seperti itu, semua kromosom adalah identik, dan kemudian rata-rata fitness populasi tidak dapat diperbaiki. Meskipun, solusi mungkin kelihatan optimal, atau agak optimal secara lokal. Hanya karena pencarian algoritma tidak mampu untuk memproses lebih lanjut. Mutasi ekuivalen dengan pencarian random, dan menolong kita untuk menghindari rugi keanekaragaman genetik.

Operator mutasi bekerja dengan memutar secara random gen yang terpilih dalam kromosom, seperti yang diperlihatkan pada gambar 2.6. Mutasi dapat terjadi pada

gen manapun dalam kromosom dengan probabilitas yang sama. Probabilitas mutasi adalah sangat kecil pada alam, dan tetap dijaga sangat rendah pada GA, cakupannya biasanya antara 0.001 dan 0.01.

Algoritma genetika menyakinkan perbaikan yang kontinu dari rata-rata fitness populasi, dan setelah sejumlah generasi tertentu (biasanya beberapa ratus) populasi meningkat sampai ke solusi optimal atau dekat dengan optimal.



Gambar 2.6.
Operasi Mutasi

2.2.2. Perkembangan Algoritma Genetika Untuk Penjadwalan Pemeliharaan

Penjawalan perawatan yang didiskusikan dalam makalah ini menentukan periode perawatan unit daya jadi pada level tertentu batas cadangan dapat dipertahankan. Proses perkembangan algoritma genetik dapat dijelaskan dalam lima langkah sebagai berikut :

Langkah 1 : menetapkan masalah, mendefinisikan kendala dan kriteria optimum. Komponen sistem daya dibuat untuk tetap dalam kondisi operasi selama periode tertentu dari life-timenya dengan perawatan yang bersifat preventif secara konstan. Tujuan penjadwalan perencanaan adalah untuk mencari serangkaian outage unit daya selama periode waktu yang telah ditentukan (normalnya satu tahun) sedemikian rupa sehingga tingkat keamanan dalam system maksimal.

Salah satu outage dalam sistem daya berhubungan dengan beberapa rugi dalam sekuritas. Batas sekuritas ditentukan oleli cadangan bersih sistem. Cadangan bersih, pada gilirannya , didefinisikan sebagai total kapasitas pembangkitan terpasang dikurangi rugi generasi karena outage sebelum terjadwal dan dikurangi ramalan beban maksimum selama periode perawatan. Sebagai contoh, jika kita mengasumsikan bahwa total kapasitas terpasang adalah 150 MW dan unit 20 MW adalah dijadwalkan untuk perawatan dalam periode waktu interval selama beban maksimum diprediksikan sebesar 100 MW, kemudian cadangan bersih akan sama dengan 30 MW. Penjadwalan perawatan harus memastikan bahwa cadangan bersih disediakan untuk menjamin suplai daya pada periode perawatan.

Diumpamakan, kebutuhan perawatan meliputi penjadwalan empat unit daya dalam 6 interval yang sama. Beban maksimum yang diharapkan selam interval ini berturut-turut adalah 70, 60, 85, 75, 60, dan 80 MW. Kapsitas unit dan kebutuhan perawatannya disajikan dalam tabel 1.

Tabel 1.
Data unit dan kebutuhan perawatan.

Nomer unit	Kapasitas unit, MW	Nomer interval yang dibutuhkan untuk perawatan
1	20	2
2	15	1
3	35	1
4	40	2

Kendala dalam masalah ini ditetapkan sebagai berikut :

- Unit 1 dipelihara pada dua interval konsekuen
- Unit 4 dipelihara pada dua interval yang terpisah dan tidak dirawat pada interval 6.
- Cadangan bersih sistem daya harus lebih besar atau sama dengan 0 pada beberapa interval.

Ukuran optimum di sini adalah maksimum cadangan bersih pada beberapa periode.

Secara matematika masalah ini dapat dijelaskan sebagai berikut :

Maximise min $\{R_i, i = 1,\text{K}, T\}$ (1)

dimana $R_i = S - P_i - \sum_{j=1}^{N} C_j X_{ij}$

S dalah kapasitas sistem, P_i adalah beban prediksi untuk interval i, C_j adalah kapasitas unit j,

$$\left(\sum_{j=1}^{N} C_j = S\right),$$

$X_{ij} = 1$, bila perawatan dilakukan pada unit j

0, sebaliknya

N adalah angka unit, dan T adalah angka interval perawatan.

Kelihatannya, total kapasitas unit berjalan pada beberapa interval harus kurang dari beban prediksi pada interval tersebut, hal ini berarti

$$\sum_{j=1}^{N} C_j X_{ij} \geq P_i \text{, untuk } i = 1,\ldots,\text{T} \qquad (2)$$

Langkah 2 : melambangkan domain masalah sebagai kromosom. Masalah penjadwalan kita menyajikan masalah penyusunan yang membutuhkan penempatan daftar tugas dalam urutan tertentu. Jadwal komplet mungkin terdiri dari sejumlah tugas yang *overlapping*, tetapi tidak semua penyusunan adalah legal, karena mungkin menyebabkan pelanggaran batas. Tugas kita adalah menyajikan jadwal komplit sebagai kromosom dengan panjang tetap.

Skema coding otomatis yang menjaga akan menugaskan tiap angka biner dan untuk membiarkan kromosom menjadi serangkaian angka biner. Meskipun, pemesanan unit dalam rangkaian tidak secara langsung mewakili jadwal. Beberapa unit dapat dirawat secara bersamaan, dan kami harus juga menyertakan waktu yang dibutuhkan untuk perawatan unit kedalam jadwal. Kemudian, jika dibandingkan dengan unit pemesanan dalam rangkaian, kami mungkin membuat serangkaian jadwal perawatan unit individu. Jadwal unit dapat dengan mudah diwakili seperti string 6-bit, dimana tiap bit adalah interval perawatan. Jika unit dirawat pada interval tertentu, bit yang bersangkutan diasumsikan bernilai 1, atau 0. jadijadwal perawatan komplit untuk masalah kami dapat diwakili dengan kromosom 24-bit.

Meskipun operator crossover dan mutasi dengan mudah dapat membentuk string biner yang memanggil untuk perawatan beberapa unit lebih dari sekali dan yang lainnya tidak semua. Sebagai tambahan, kami dapat memanggil untuk periode perawatan yang akan melebihi jumlah interval yang betul-betul diperlukan untuk perawatan unit.

Pendekatan yang lebih baik adalah untuk merubah sintak kromosom. Seperti yang telah didiskusikan, kromosom adalah kumpulan bagian dasar yang disebut gen. menurut kebiasaan, tiap gen diwakili satu bit dan tidak dapat dipecah menjadi elemen yang lebih kecil. Pada masalah kami, kami dapat mengambil konsep yang sama, tetapi gen diwakili oleh 4 bit. Dengan kata lain, bagian yang tidak dapat dipisahkan yang paling kecil dari kromosom kami adalh string 14-bit. Penyajian ini mengijinkan operator crossover dan mutasi beraksi berdasarkan pondasi teori algoritma genetika. Yang sisa untuk dilakukan adalah memproduksi kumpulan gen untuk tiap unit, seperti yang ditunjukkan pada tabel 2.

Tabel 2. Unit kumpulan gen.

Unit	Gene pool	Unit	Gene pool
1	110000 011000 001100 000110 000011	2	100000 010000 001000 000100 000010 000001
3	100000 010000 001000 000100 000010 000001	4	101000 100100 100010 010100 010010 010001

GA sekarang dapat membentuk populasi awal kromosom dengan pengisian kromosom 6 gen yang dipilih secara random dari kelompok yang bersangkutan.

Langkah 3 : definisikan fungsi fitness untuk mengevaluasi performa kromosom. Evaluasi kromosom adalah bagian rumit dari GA, karena kromosom yang dipilih untuk perkawinan berdasarkan kebugarannya. Fungsi fitness harus menangkap apa yang membuat jadwal perawatan baik atau jelek untuk user. Pada masalah kami, kami mengaplikasikan fungsi sederhana yang berkaitan dengan pelanggaran batas dan cadangan bersih pada tiap interval.

Fungsi fitnees kromosom *k* dapat didefinisikan sebagai

$$F_k = \min\{R_i, i = 1, K, T\} \qquad (3)$$

Jika cadangan bersih positif pada beberapa interval, kromosom tidak akan melanggar batas[2] dan fugsi F_k mewakili fitness kromosom. Dan bila cadangan bersih berada pada beberapa interval negatif, jadwal menjadi illegal, dan fungsi fitness kembali nol. Meskipun pada permulaan run, populasi awal yang dibuat secara random mungkin terdiri dari semua jadwal illegal. Pada kasus ini, nilaifitness kromosom tinggal tanpa perubahan, dan pemilihan terjadi sesuai dengan persetujuan nilai fitness aktual.

Kemudian, fungsi fitness kromosom *k* dapat didefinisikan kembali sebagai

$\Phi_k(F_k) = 0$, bila $F_k < 0$ dan terdiri dari *m* dan *n* maka $F_m \geq 0$ dan $F_n \leq 0$;

F_k, sebaliknya.

(4)

dimana $F_k = \min\{R_i, i = 1, K, T\}$

Langkah 4 : Bangunlah operator genetik. Pembangunan operator genetic adalah proses yang menantang, dimana eksperimen dibutuhkan untuk membuat crossover dan mutasi bekerja dengan tepat. Kromosom yang telah dipatahkan dengan cara yang legal untuk masalah yang ada. Karena kami sudah merubah sintak kromosom untuk masalah kami, kami dapat menggunakan operator genetik dalam bentuk klasiknya. Tiap gen dalam kromosom disajikan dengan string 6-bit tak terpisahkan yang terdiri dari jadwal perawatan yang tepat untuk unit tertentu. Kemudian mutasi random gen atau rekombinasi beberapa gen dari dua kromosom induk mungkin hanya menghasilkan dalam perubahan jadwal perawatan untuk unit individu, tetapi tidak dapat membuat kromosom "buatan".

Langkah 5 : jalankan GA dan setel parameternya. Sekarang waktunya menjalankan GA. Pertama kita harus memilih ukuran populasi dan jumlah generasi yang akan dijalankan. Akal sehat menyarankan kita untuk populasi yang lebih besar dapat mencapai solusi yang lebih baik dari pada populasi yang lebih kecil, tetapi kerjanya lebih lambat. Sesungguhnya, bagaimanapun, ukuran populasi yang paling efektif tergantung pada masalah yang akan diselesaikan, khususnya, pada masalah skema coding[11]. GA dapat berjalan hanya dengan jumlah generasi yang terbatas untuk menghasilkan solusi. Kelihatannya, kami akan memilih populasi yang sangat besar dan menjalankannya hanya sekali. Atau, kami akan memilih populasi yang lebih kecil dan menjalankannya beberapa kali. Pada beberapa kasus, hanya eksperimen yang dapat memberikan jawaban.

BAB III

PLTGU UNIT PEMBANKITAN GRESIK

3.1. Sejarah PLTGU Gresik

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

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

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

3.2 Konfigurasi dan Kapasitas Unit Pembangkitan Gresik

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

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

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

Tabel 3.1

Tipe Pembangkit di Unit Pembangkitan Gresik

I	**Gas Turbin Power Plant**				
	GT 1	HSD/NG	20 MW	Alsthom-France	07-6-1978
	GT 2	HSD/NG	20 MW	Alsthom-France	09-6-1978
	GT 3	HSD	21 MW	Alsthom-France	20-8-1984
	GT 4	HSD/NG	20 MW	GE- USA	02-9-1994
	GT 5	HSD/NG	21 MW	GE- USA	24-2-1995
II	**Steam Turbin Power Plant**				
	ST 1	HSD/RO	100 MW	Toshiba-Japan	31-08-1981
	ST 2	HSD/RO	100 MW	Toshiba-Japan	14-11-1981
	ST 3	HSD/NG	200 MW	Toshiba-Japan	15-03-1988
	ST 4	HSD/NG	200 MW	Toshiba-Japan	01-07-1988

III	**Combined Cycle Power Plant**				
	Blok 1				
	GT 1.1	NG	112,45 MW	Mitsubishi	30-03-1992
		HSD	100,98 MW		24-04-1994
	GT 1.2	NG	112,45 MW	Mitsubishi	01-05-1992
		HSD	100,98 MW		02-05-1994
	GT 1.3	NG	112,45 MW	Mitsubishi	02-06-1992
		HSD	100,98 MW		01-05-1994
	GT 1.0	-	118,91 MW	Mitsubishi	10-04-1993
	Blok 2				
	GT 2.1	NG	112,45 MW	Mitsubishi	20-07-1992
		HSD	100,98 MW		01-05-1994
	GT 2.2	NG	112,45 MW	Mitsubishi	14-08-1992
		HSD	100,98 MW		04-02-1994
	GT 2.3	NG	112,45 MW	Mitsubishi	18-09-1992
		HSD	100,98 MW		03-05-1994
	GT 2.0	-	118,91 MW	Mitsubishi	05-08-1993
	Blok 3				
	GT 3.1	NG	112,45 MW	Mitsubishi	14-01-1993
	GT 3.2	NG	112,45 MW	Mitsubishi	19-01-1993
	GT 3.3	NG	112,45 MW	Mitsubishi	13-01-1993
	GT 3.0	-	118,91 MW	Mitsubishi	30-11-1993
	TOTAL KAPASITAS		**2280 MW**		

Sumber data operational PLTGU Gresik

Kapasitas terpasang PLTGU Unit Pembangkitan Gresik :

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

Turbin Uap : 188,92 MW (BBG) dan 157,79 MW (BBM)

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

Kapasitas total PLTGU (BBG) : 1578,78 MW

Kapasitas Minimum :

Kapasitas total per Blok, konfigurasi 3.3.1 : 125 MW

Kapasitas total PLTGU : 375 MW

Kapasitas Maksimum :

Turbin Gas : 115 MW (BBG), Konfigurasi 3.3.1

Turbin Uap : 189 MW (BBG), Konfigurasi 3.3.1

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

Kapasitas total PLTGU (BBG) : 1614,6 MW

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

- Blok pengolahan bahan bakar
- Blok Turbin Gas
- Blok Turbin Uap
- Blok kendali operasi

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

a) Blok pengolahan bahan bakar PLTGU Unit Pembangkitan Gresik.

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

Peralatan utama yang terdapat pada sirkit bahan bakar :

- Inlet Scrubber
- Condensat Drain Tank
- Cold Stack
- Filter Separator
- Pressure Control Valve

b) Blok Turbin Gas PLTGU Unit Pembangkitan Gresik.

Selain berfungsi sebagai peralatan konversi energi listrik dari bahan bakar, juga merupakan sumber energi pnas yang akan dimanfaatkan oleh sirkit HRSG. Peralatan utama yang terdapat pada sirkit turbin gas meliputi:

- Compressor
- Combustion chamber
- Gas turbin
- Turbo Generator

c) Blok turbin uap PLTGU Unit Pembangkitan Gresik.

Meliputi beberapa peralatan utama :

1. *Heat Recovery Steam Generator* (HRSG) terdiri,

- Exhaust damper
- By pass stack
- Silancer
- Preheater

- Economizer (low & high pressure)
- Evaporator (low & high pressure)
- Superheater (primary & secondary)
- Steam drum (primary & secondary)
- Water wall tube
- Deaerator

2. Steam turbin (dual pressure)
3. Generator turbo

Peralatan yang terdapat pada sistem air bersih dan sirkulasi meliputi :

1. Condesat Ejector pump
2. Codenser
3. Gland Leakage Condenser
4. Air Ejector
5. Boiler Feed Pump
6. Boiler Circulation Pump
7. Down Comer

Dan peralatan yang terdapat pada sistem air pendingin meliputi :

1. Feed Water Tank
2. Condenser Water Pump

d) **Blok kendali operasi PLTGU Unit Pembangkitan Gresik**

Berfungsi sebagai pusat sistem kontrol bagi seluruh Blok PLTGU. Pengendalian operasi PLTGU berlangsung secara otomatis yang dipantau melelaui terminal komputer yang terpasang di Control Centre Room (CCR). Didalam CCR terdapat empat buah Block Control Desk (BCD) yang pada prinsipnya merupakan terminal input-output dari komputer tersebut unutk memantau operasi PLTGU. Tugas utama komputer adalah untuk menyelenggarakan supervisi dan pengendalian operasi PLTGU. Untuk menyelenggarakan tugas supervisi dan pengendalian tersebut, komputer mengumpulkan data dan informasi dari sistem yang kemudian diolah menurut prosedur tertentu, dimana prosedur ini diatur oleh perangkat lunak (software) komputer. Informasi yang dikumpulkan komputer berasal dari peralatan-peralatan telemetri yang menghasilkan sinyal analog, melalui Direct Digital Controler (DDC) informasi tersebut diterima oleh komputer sebagai sinyal digital untuk kemudian diolah dan ditampilkan sebagai informasi real time yang perlu diketahui operator yang bertugas. Selain informasi real time, komputer juga menyimpan data-data masa lalu (history) yang direkam dalam suatu media penyimpan informasi tertentu. Fungsi komputer yang demikian ini disebut telah melakukan Superviosory Control And Data Acquisition (SCADA).

3.4. Data Jurnal Yang Dijadikan Acuan

Tabel 3.2.
Data Unit Dan Kebutuhan Perawatan Di Jurnal

Nomer Unit	Daya Terpasang (MW)	Periode
1,...,10	10	1
11,...,20	20	1
21,...30	30	1
31,...,35	40	2
36,...,40	40	2
41,...,45	50	2
46,...,50	55	3
51,...,55	60	4
56,...,60	70	3
61	85	4
62	90	4

ATURAN :

1. Perawatan pada berbagai unit dimulai pada saat sebuah interval dan berakhir pada interval yang sama. Pengecualian hanya untuk perawatan unit 62.
2. Unit 62 akan di rawat dalam empat interval dibagi ke dalam dua periode yang sama. Periode operasi minimum adalah 8 interval
3. Unit 61 dan 62 akan disediakan untuk periode musim dingin (interval 21-26), unit 56-60 akan disediakan dalam interval 23-26
4. Unit 1-5, 15-20 dan 25 akan disediakan dalam periode musim panas. (interval 7-13).

3.5. Data Input Real

Tabel 3.3.
Data Unit dan Kebutuhan Perawatan

NO	UNIT	DAYA TERPASANG MW	JENIS INSPEKSI
1	GT 1.1	112.45	CI
2	GT 1.2	112.45	CI
3	GT 1.3	112.45	CI
4	ST 1.0	118.91	SI
5	GT 2.1	112.45	TI dan CI
6	GT 2.2	112.45	CI
7	GT 2.3	112.45	CI dan MI
8	ST 2.0	188.91	SI
9	GT 3.1	112.45	TI
10	GT 3.2	112.45	MI
11	GT 3.3	112.45	MI
12	ST 3.0	188.91	ME
13	PLTU 1	100	SE
14	PLTU 2	100	SI
15	PLTU 3	200	SE
16	PLTU 4	200	SI+
17	PLTG GRESIK 1	20.1	CI
18	PLTG GRESIK 2	20.1	HGTI
19	PLTG GILITIMUR 1	21	CI + VS1

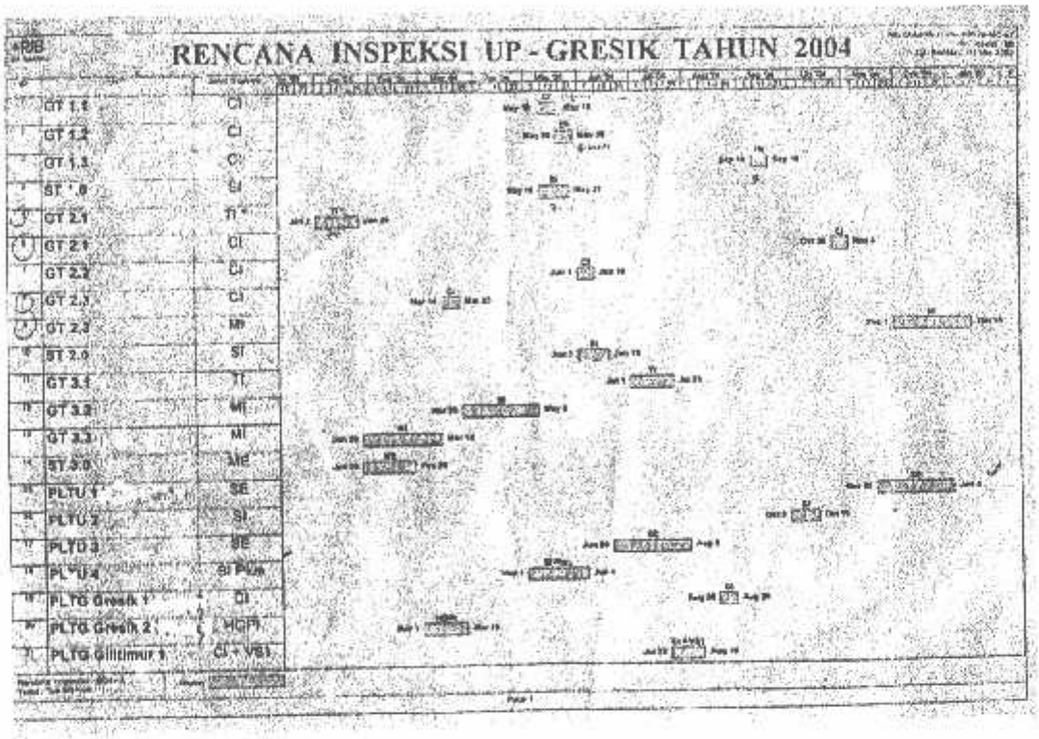
Jenis Inspeksi	Periode
MI	4
ME	3
CI	1
HGPI	2
SI	2
SI Plus	3
TI	3
SE	4
CI+VSI	2

ATURAN
1. Tidak ada pelayanan yang sama dalam 1 periode
2. Pelayanan yang boleh bersama-sama adalah :

MI – ME
CI - HGPI
MI - SI Plus
SI Plus - SI
SI – CI
TI – SE
CI+VSI - SE
MI – SE

CI : COMMON INSPECTION (10 hari)
TI : TURBIN INSPECTION (25 hari)
SI : SIMPLE INSPECTION (18 hari)
MI : MAJOR INSPECTION (45 hari)
HGTI : HOT GAS PATH INSPECTION (25 hari)
SE : SERIOUS EXPERT (45 hari)
ME : MEAN EXPERT (25 hari)
CI + VS1 (LEBIH DARI NORMAL)

DIMAINTENACE SETIAP KELIPATAN 8000 JAM



Gambar 3.1.
Rencana Perawatan UP Gresik Tahun 2004

BAB IV

ANALISA DATA DENGAN METODE GENETIC ALGORITHM PADA PT. PEMBANGKITAN JAWA ~ BALI

4.1 Program Komputer Metode Genetic Algorithm

Untuk pemecahan masalah penjadwalan perawatan pada sistem tenaga digunakan bantuan program komputer. Program komputer ini sangat berguna untuk mempercepat proses perhitungan yang membutuhkan ketelitian tinggi dan sering melibatkan iterasi yang memerlukan waktu yang lama bila dikerjakan secara manual.

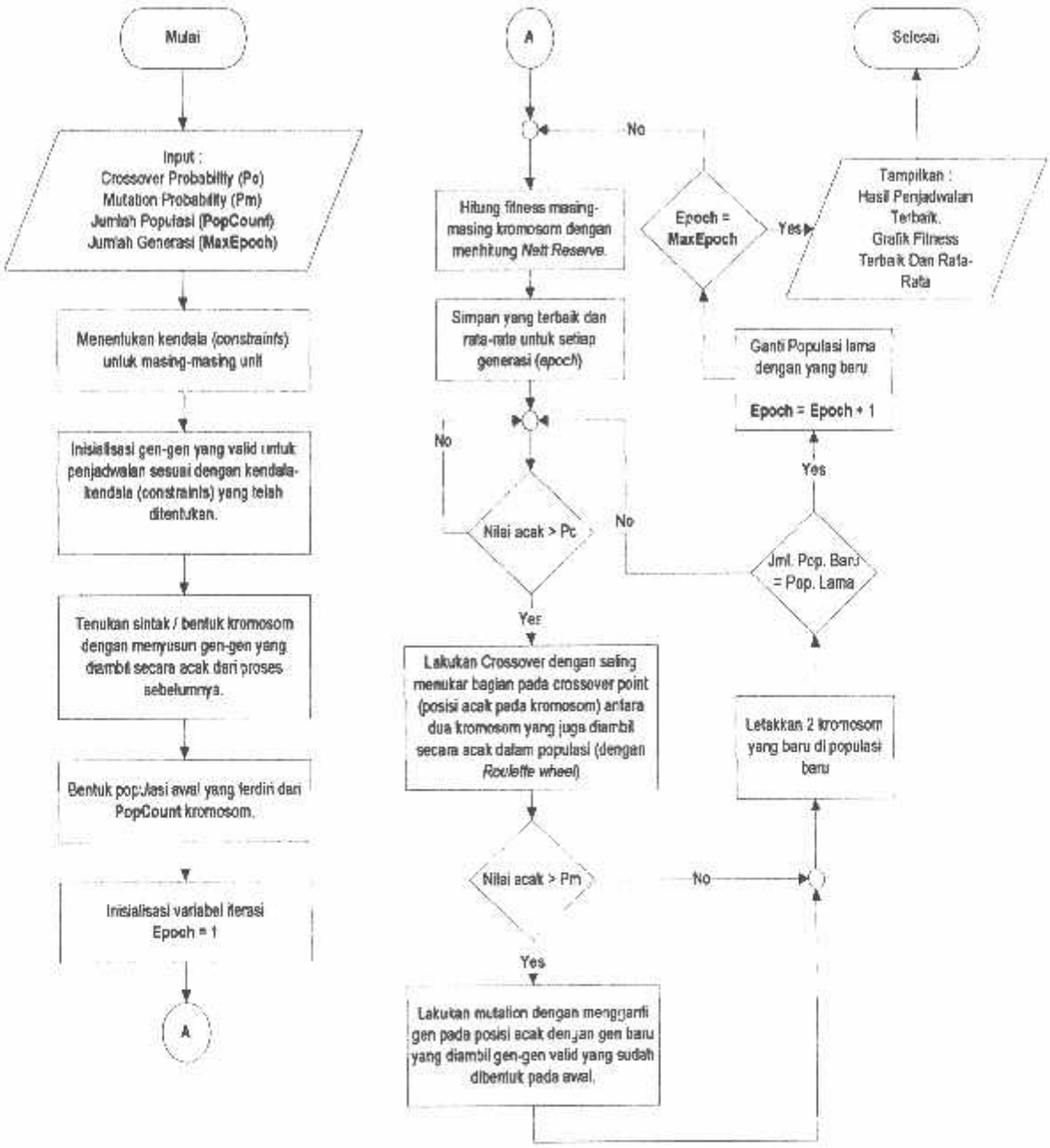
Program komputer ini menggunakan bahasa pemrograman Borland Delphi versi 7, yang merupakan bahasa pemrograman terstruktur yang relatif mudah untuk dipelajari dan mudah penggunaannya.

4.2 Algoritma Program

1. Masukan data parameter Crossover Probability (Pc), Mutation Probability (Pm), Jumlah Populasi (PopCount), Jumlah Generasi (MaxEpoch).
2. Menentukan kendala (constraints) untuk masing-masing unit.
3. Menginisialisasi gen-gen yang valid untuk penjadwalan sesuai dengan kendala-kendala (constraints) yang telah ditentukan.
4. Tentukan sintak/bentuk kromosom dengan menyusun gen-gen yang diambil secara acak dari proses sebelumnya.
5. Menginisialisasi variable iterasi Epoch = 1.

6. Menghitung fitness masing-masing kromosom dengan menghitung Nett Reserve.
7. Menyimpan yang terbaik dan rata-rata untuk setiap generasi (epoch).
8. Melakukan Crossover dengan saling menukar bagian pada crossover point (posisi acak pada kromosom) antara dua kromosom yang juga diambil secara acak dalam populasi (dengan Roulette wheel).
9. Melakukan mutation dengan mengganti gen pada posisi acak dengan gen baru yang diambil gen-gen valid yang sudah dibentuk pada awal.
10. Meletakkan 2 kromosom yang baru di populasi baru.
11. Menggənti Populasi lama dengan yang baru Epoch = Epoch + 1
12. Menampilkan Hasil Penjadwalan Terbaik Grafik Fitness Terbaik Dan Rata-Rata.

4.3 Flowchart program



Gambar 4.1.
Flowchart Program

4.4. Tampilan Program

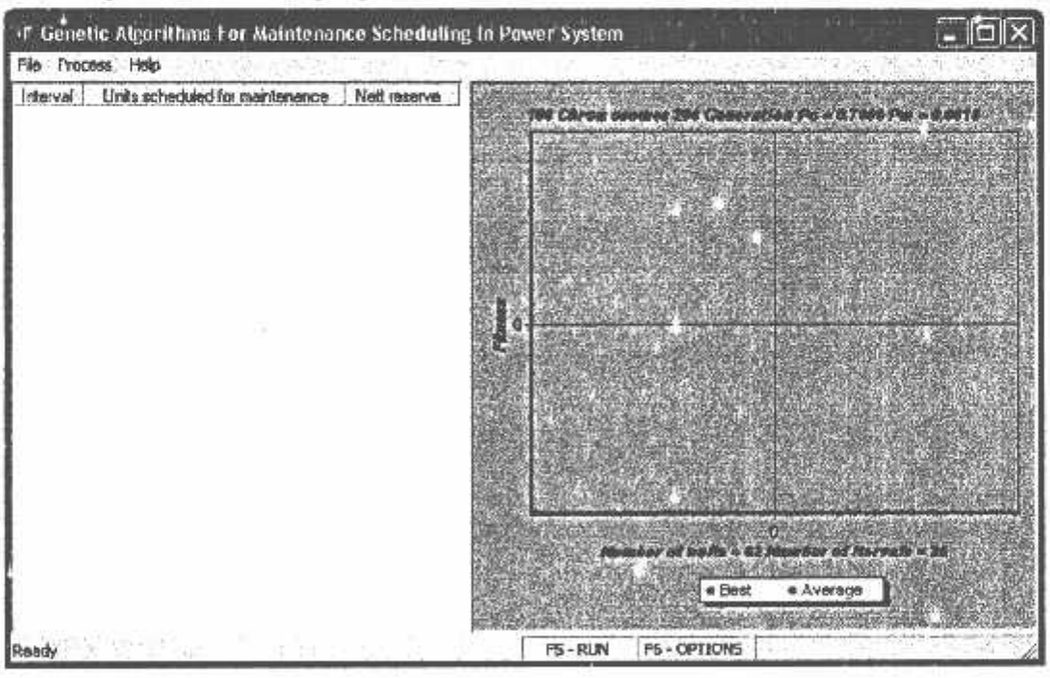
4.4.1. Program Validasi Program Komputer

Agar kinerja unit-unit pada PLTGU Gresik dapat bekerja secara maksimal, maka diperlukan perawatan pada komponen-komponen utama yang terdapat di Unit Pembangkitan Gresik secara teratur. Oleh karena diperlukan suatu jadwal perawatan yang tepat, dimana masalah penjadwalan perawatan dapat dipecahkan dengan menggunakan metode Genetik Algoritma.

Berikut ini adalah tampilan dari data validasi program komputer :

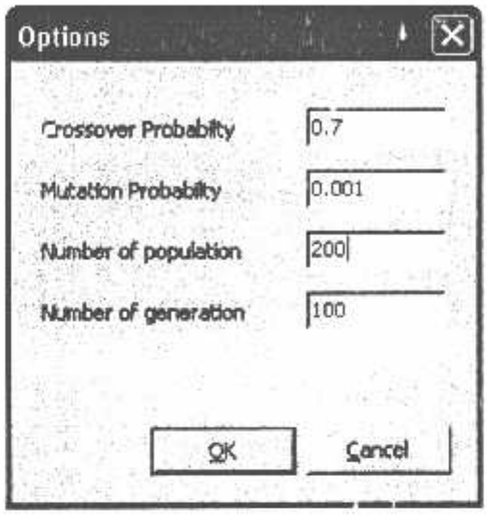
Program ini menggunakan bahasa pemrograman Bortland Delphi 7 kemudian dieksekusi menggunakan komputer dengan spesifikasi Prosesor AMD Athlon 1.7, RAM 256 Mb.

1. Tampilan utama dari program



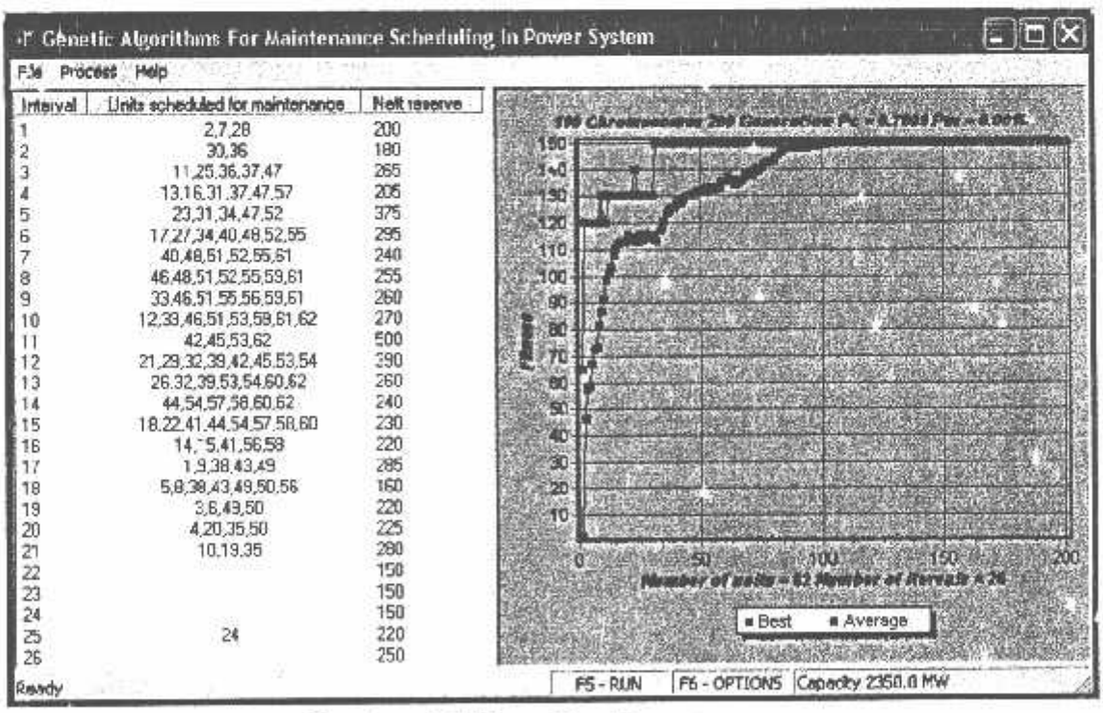
Gambar 4.2. Tampilan Utama

2. Tekan F6 untuk memasukan data



Gambar 4.3.
Menu Tampilan Data

3. Tekan F5 untuk menjalankan program

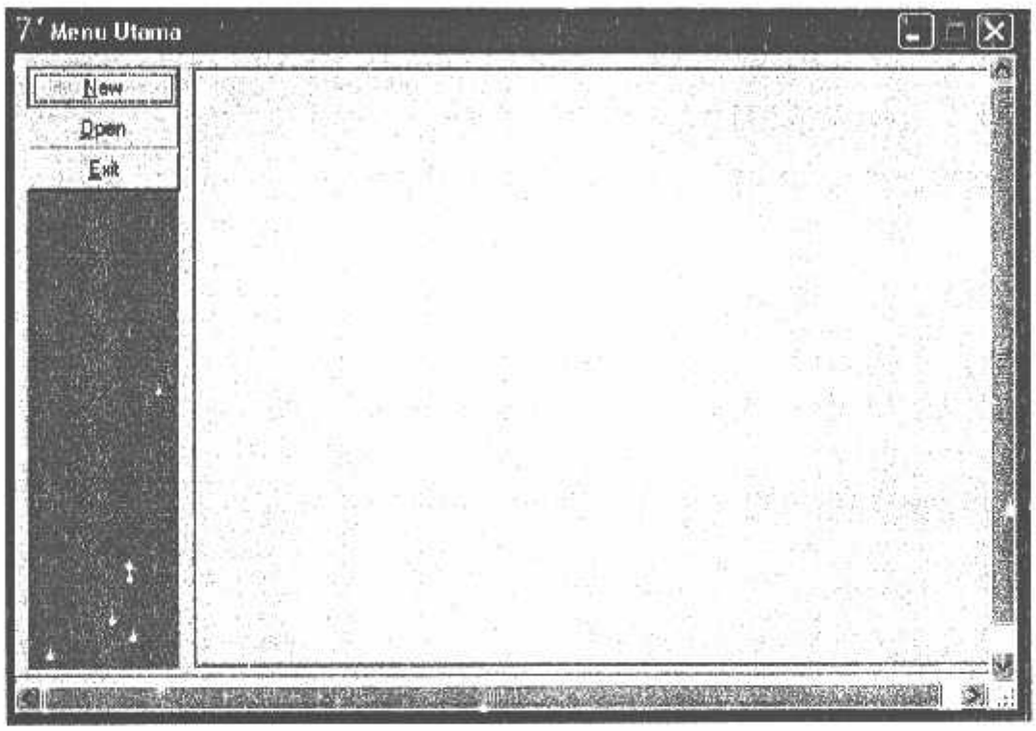


Gambar 4.4. Tampilan Program

4.4.1. Program Real

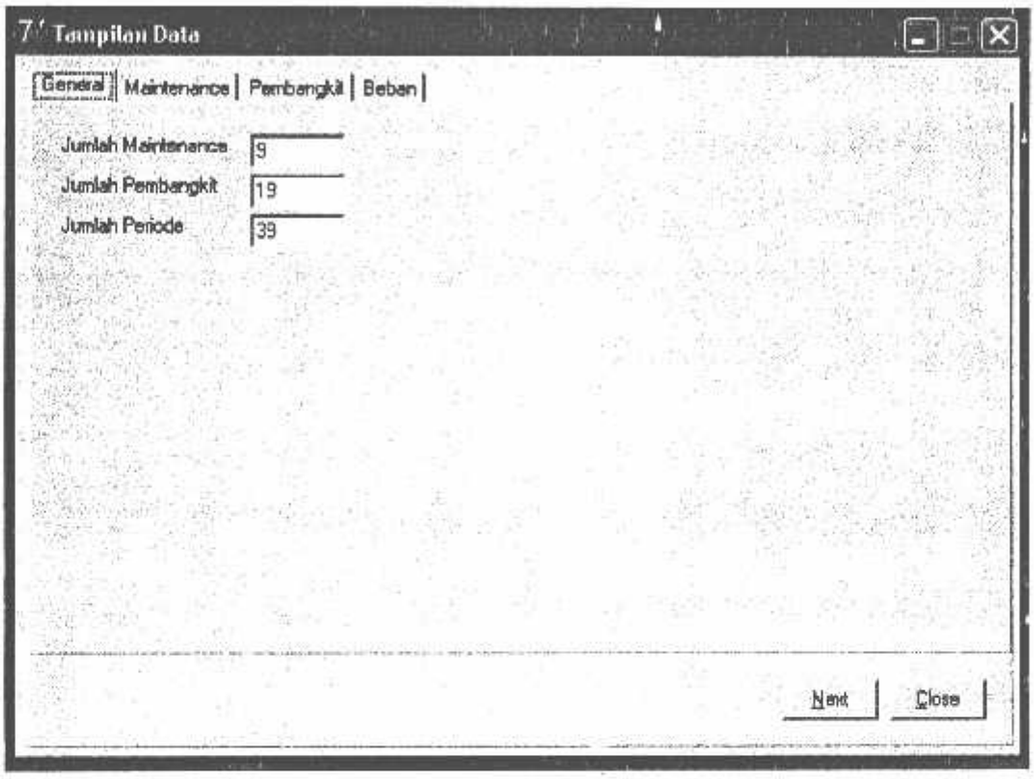
Program ini menggunakan bahasa pemrograman Bortland Delphi 7 kemudian dieksekusi menggunakan komputer dengan spesifikasi Prosesor AMD Athlon 1.7, RAM 256 MB.

1. Tampilan utama dari program



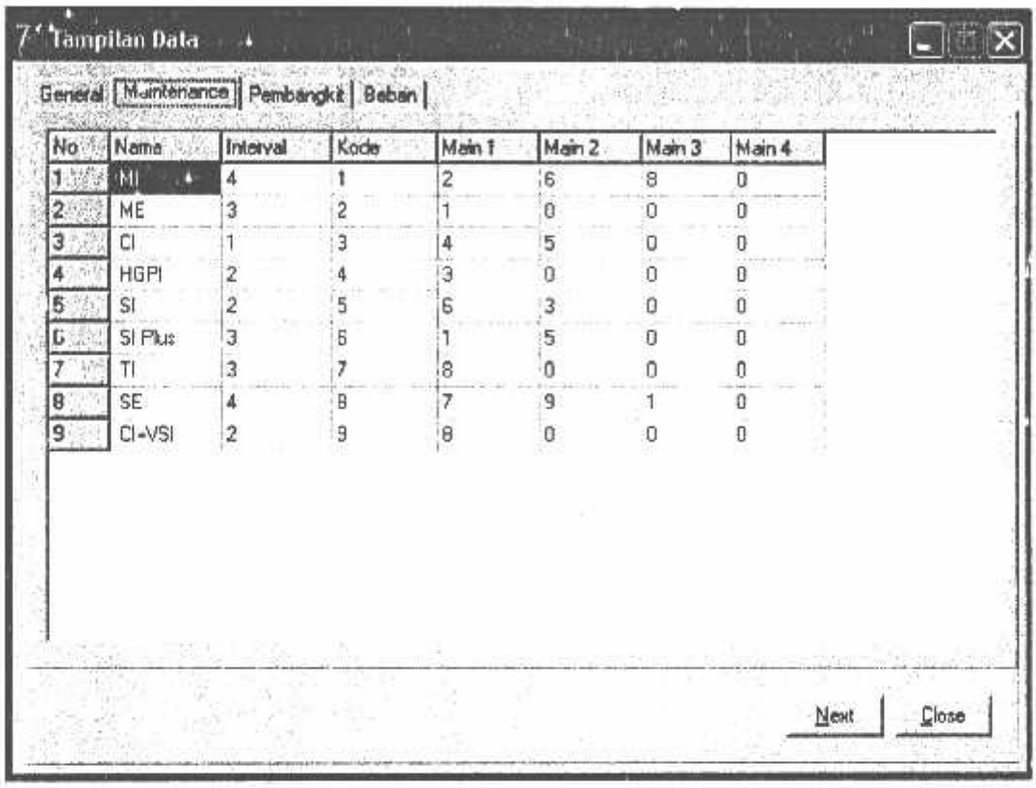
Gambar 4.5.
Tampilan Program Utama

2. Tekan tombol *General data* untuk memasukan data, berapa jumlah maintenance, jumlah pembangkitan, jumlah periode, dan jumlah beban untuk eksekusi penjadwalan maintenance.



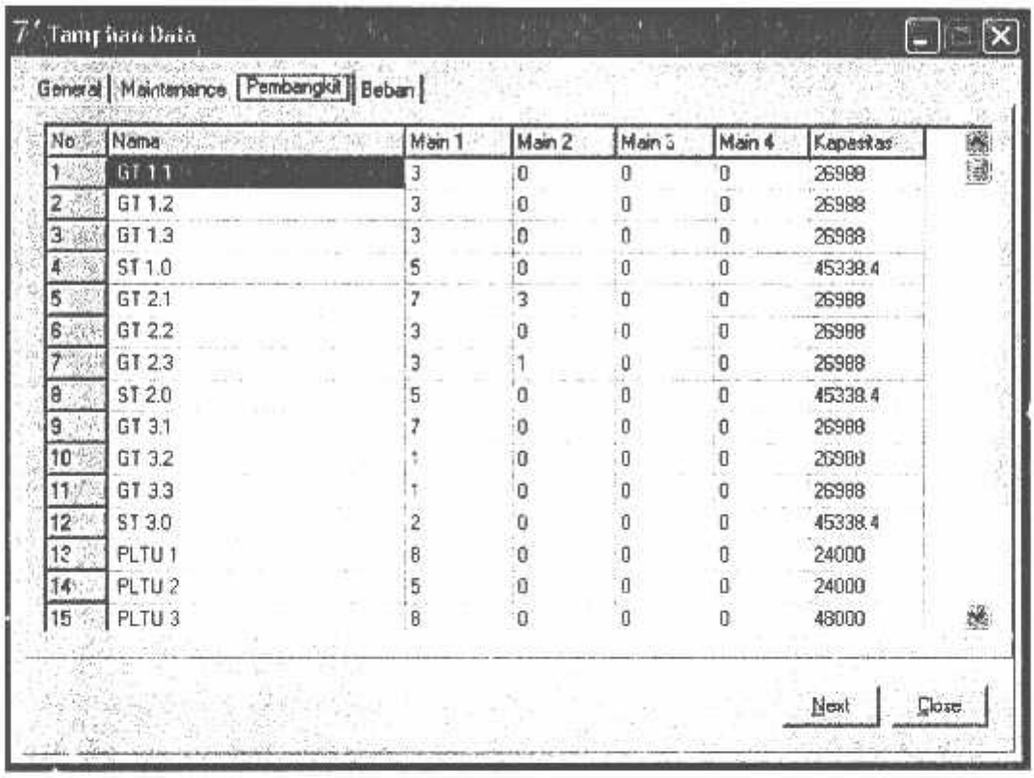
Gambar 4.6.
Menu Tampilan Data Pada Data Utama

3. Tekan tombol maintenance, masukan data nama maintenance, interval, kode, main 1, main 2, main 3, dan main 4.



Gambar 4.7.
Tampilan Data Maintenance

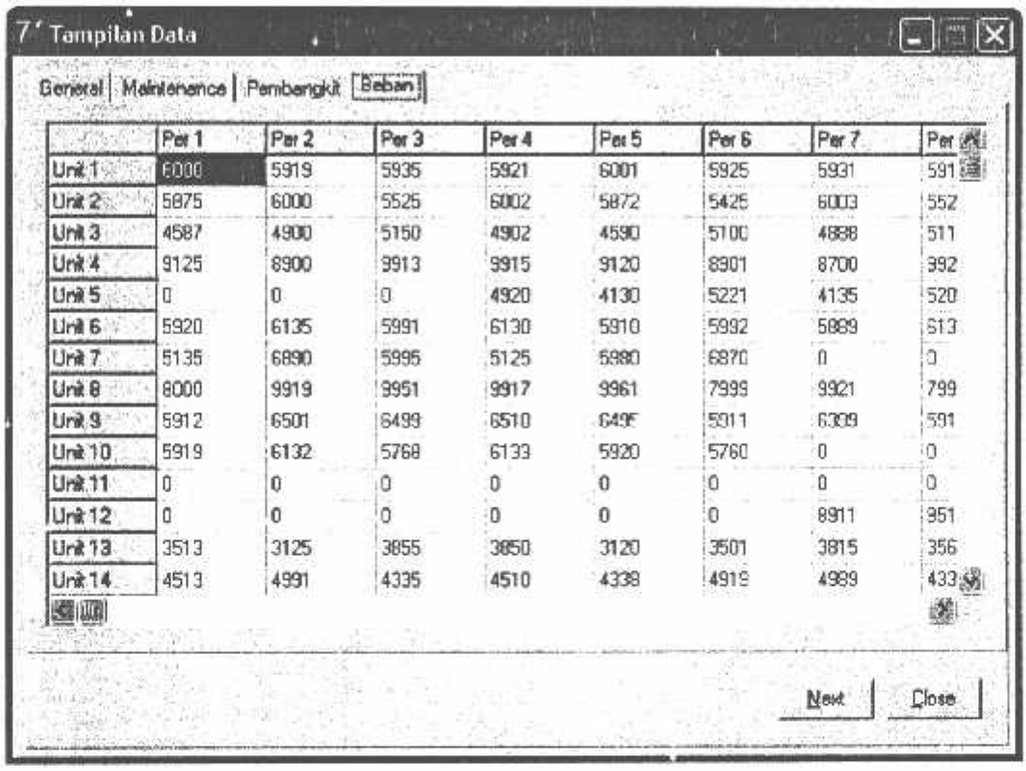
4. Tekan Tombol Pembangkit, masukan data nama unit pembangkit, main 1, main 2, main 3, main 4, dan kapasitas daya terpasang tiap unit



No	Nama	Main 1	Main 2	Main 3	Main 4	Kapasitas
1	GT 1.1	3	0	0	0	26988
2	GT 1.2	3	0	0	0	26988
3	GT 1.3	3	0	0	0	26988
4	ST 1.0	5	0	0	0	45338.4
5	GT 2.1	7	3	0	0	26988
6	GT 2.2	3	0	0	0	26988
7	GT 2.3	3	1	0	0	26988
8	ST 2.0	5	0	0	0	45338.4
9	GT 3.1	7	0	0	0	26988
10	GT 3.2	1	0	0	0	26988
11	GT 3.3	1	0	0	0	26988
12	ST 3.0	2	0	0	0	45338.4
13	PLTU 1	8	0	0	0	24000
14	PLTU 2	5	0	0	0	24000
15	PLTU 3	8	0	0	0	48000

Gambar 4.8.
Tampilan Data Pembangkit

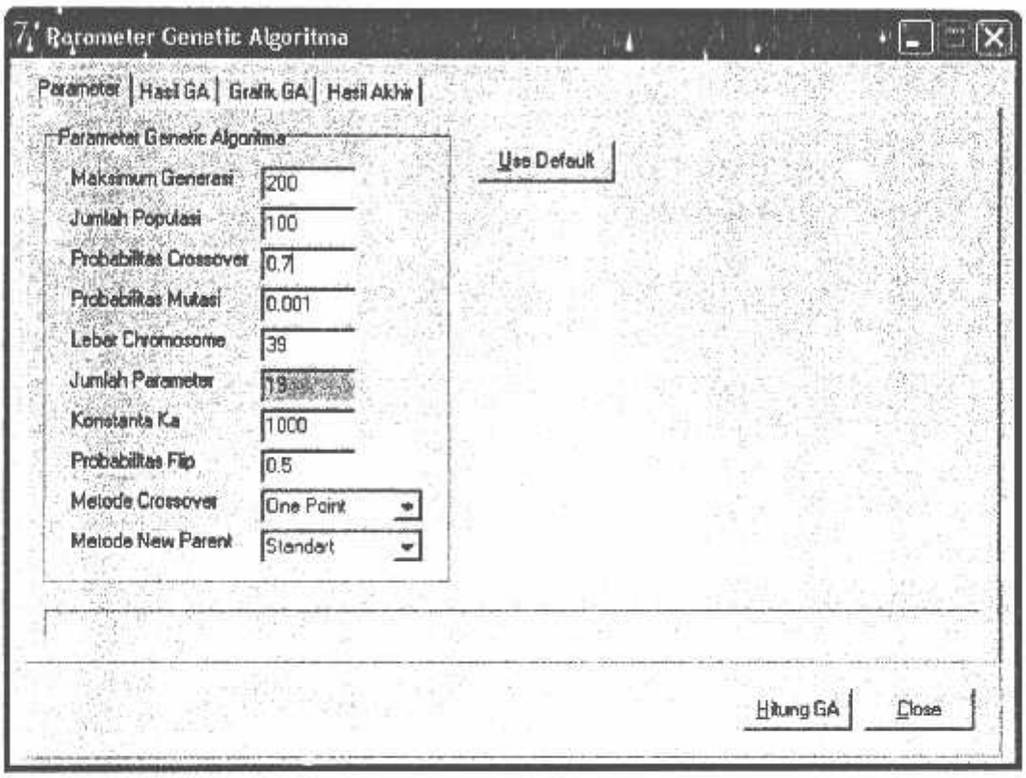
5. Masukan data beban yang dilayani tiap periode dengan menekan tombol beban.



	Per 1	Per 2	Per 3	Per 4	Per 5	Per 6	Per 7	Per
Unit 1	6000	5919	5935	5921	6001	5925	5931	591
Unit 2	5875	6000	5525	6002	5872	5425	6003	552
Unit 3	4587	4900	5150	4902	4590	5100	4898	511
Unit 4	9125	8900	9913	9915	9120	8901	8700	992
Unit 5	0	0	0	4920	4130	5221	4135	520
Unit 6	5920	6135	5991	6130	5910	5992	5899	613
Unit 7	5135	6890	5995	5125	5980	6870	0	0
Unit 8	8000	9919	9951	9917	9961	7999	9921	799
Unit 9	5912	6501	6499	6510	649F	5911	6309	591
Unit 10	5919	6132	5768	6133	5920	5760	0	0
Unit 11	0	0	0	0	0	0	0	0
Unit 12	0	0	0	0	0	0	8911	951
Unit 13	3513	3125	3855	3850	3120	3501	3815	356
Unit 14	4513	4991	4335	4510	4338	4919	4989	433

Gambar 4.9.
Tampilan Data Beban

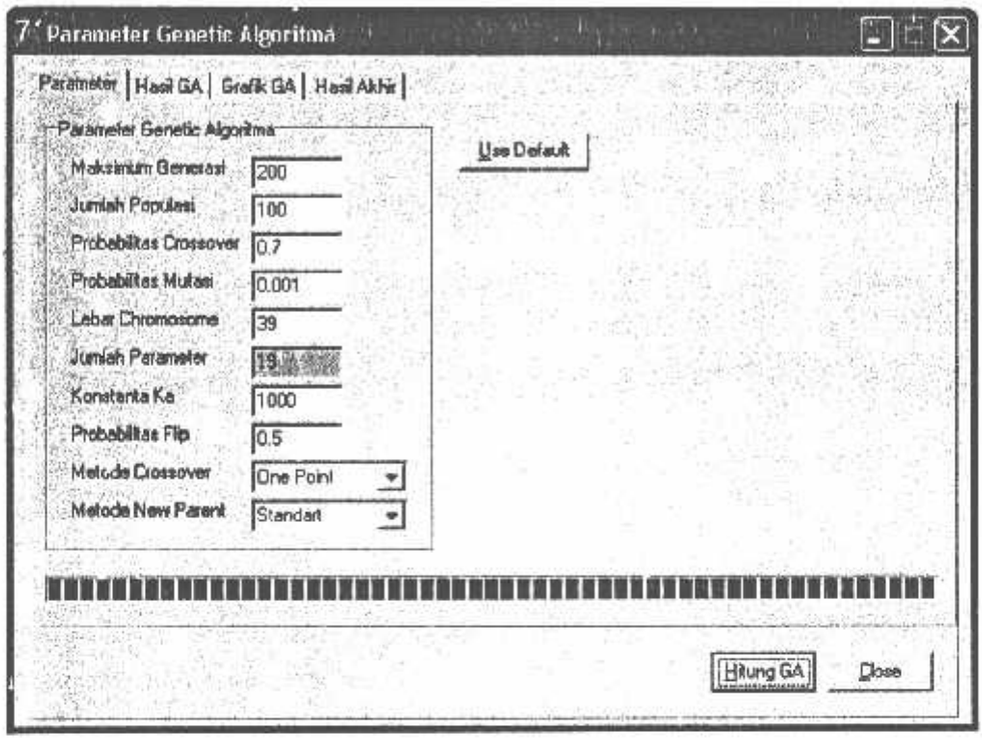
6. Tekan next dan masukkan data parameter.



Gambar 4.10.

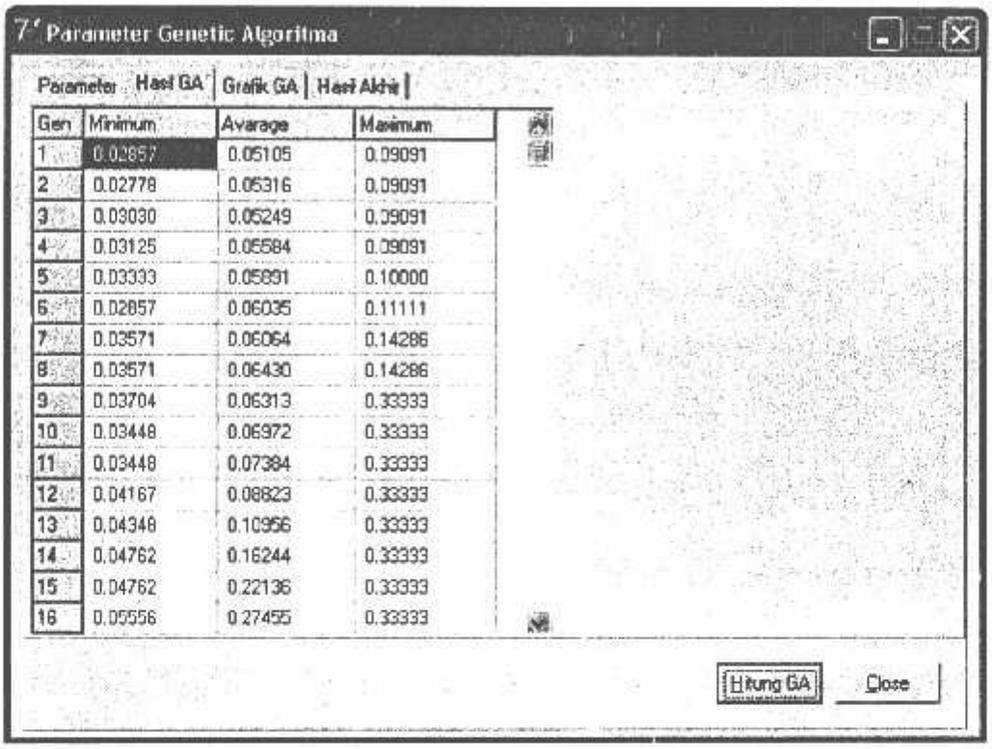
Tampilan Data Parameter *Genetic Algorithm*

7. Tekan Hitung GA



Gambar 4.11.
Proses GA

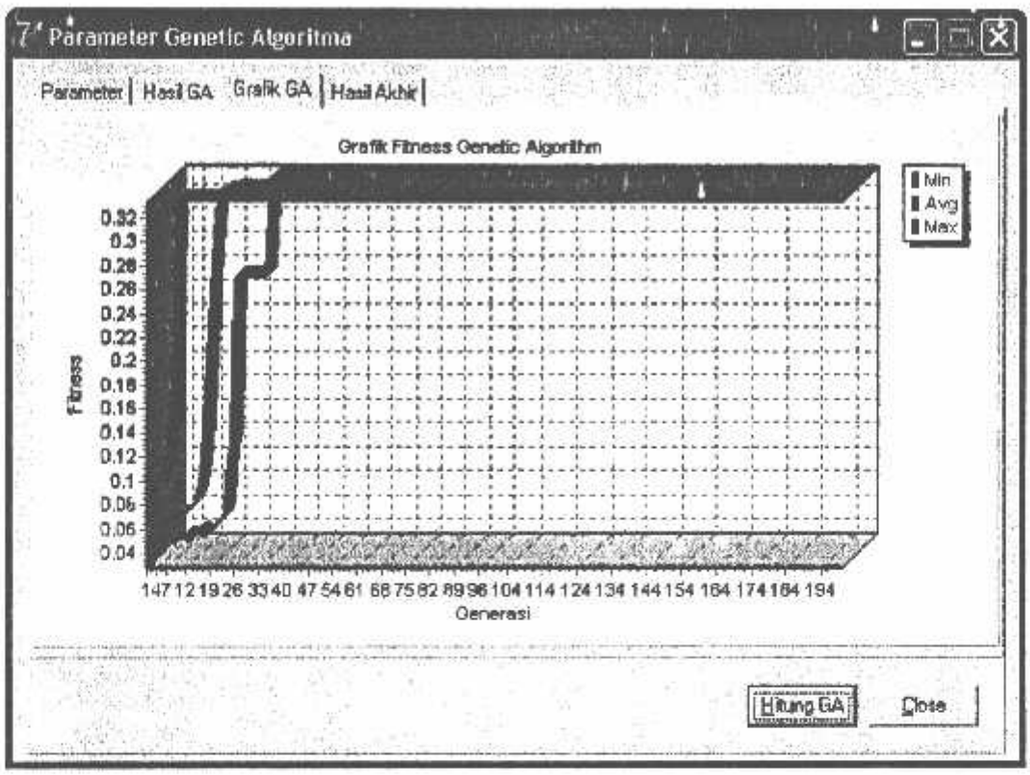
8. Tekan Hasil GA



Gen	Minimum	Avarage	Maximum
1	0.02857	0.05105	0.09091
2	0.02778	0.05316	0.09091
3	0.03030	0.05249	0.09091
4	0.03125	0.05584	0.09091
5	0.03333	0.05891	0.10000
6	0.02857	0.06035	0.11111
7	0.03571	0.06064	0.14286
8	0.03571	0.06430	0.14286
9	0.03704	0.06313	0.33333
10	0.03448	0.06972	0.33333
11	0.03448	0.07384	0.33333
12	0.04167	0.08823	0.33333
13	0.04348	0.10956	0.33333
14	0.04762	0.16244	0.33333
15	0.04762	0.22136	0.33333
16	0.05556	0.27455	0.33333

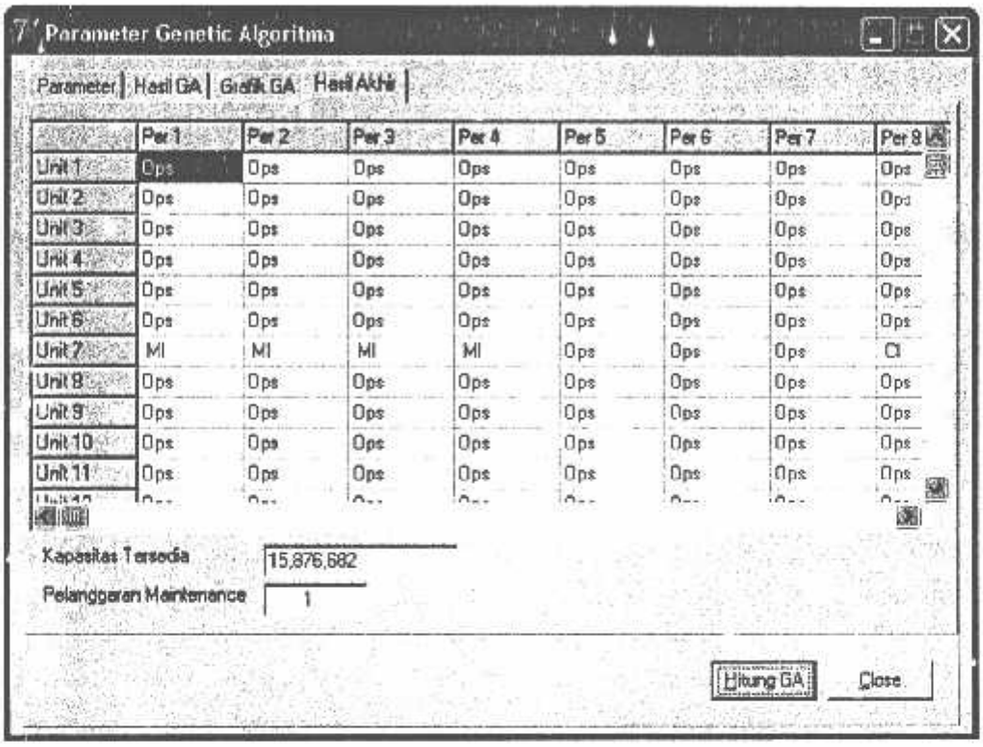
Gambar 4.12.
Fitness Minimum, Rata-rata, dan Maksimum

9. Tekan Grafik GA



Gambar 4.13.
Grafik Fitness Genetic Algorithm

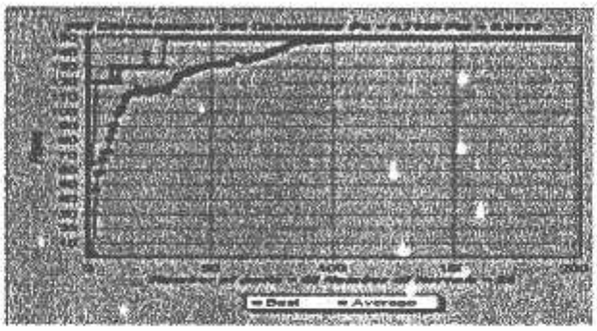
10. Kemudian Tekan Hasil Akhir



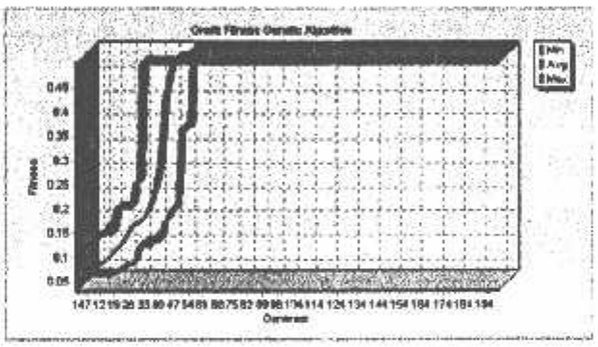
Gambar 4.14.

Jadwal Maintenance Tiap Unit, Kapasitas Yang Tersedia dan Pelanggaran Maintenance

4.4.3. PERBANDINGAN PROGRAM VALIDASI PROGRAM KOMPUTER DENGAN PROGRAM REAL



Gambar 4.15.
Grafik Fitness Genetic Algorithm (Validasi)



Gambar 4.16.
Grafik Fitness Genetic Algorithm (Real)

Pada gambar 4.15 dan 4.16 ditunjukkan grafik fitnees yang semakin tinggi, dimana grafik best dan average (validasi) / grafik min, avg, dan max (real) akhirnya sejajar secara konstan dan akhirnya membentuk garis lurus, dan itu menunjukkan hasil penjadwalan maintenance yang terbaik.

4.4.4. PERBANDINGAN ANTARA HASIL PROGRAM DENGAN DATA PLN

Tabel 4.1.
Perbandingan Kapasitas Tersedia PT. PJB Dengan Metode GA

Kapasitas Tersedia PT. PJB (MWH)	Kapasitas Tersedia GA (MWH)
11.645.010	15.876.682

Tabel 4.1 diatas menunjukan bahwa dengan menggunakan metode *Genetic algorithm* untuk penjadwalan perawatan diperoleh kapasitas tersedia sebesar 15.876.682 MWH .

4.4.5. VALIDASI PROGRAM KOMPUTER TERHADAP PROGRAM JURNAL

Interval	Units scheduled for maintenance	Nett reserve
1	2 7,28	200
2	30,16	180
3	11,25,36,37,47	265
4	13,16,31,37,47,57	295
5	23 31 34 47,52	375
6	17,27 38 40 48,52 55	295
7	40,48,51,52,55,61	240
8	45,48,51,52,55,59,61	255
9	33,46,51,55,56,59,61	260
10	12,33,46,51,53,59,61,62	270
11	42,45,53,62	500
12	21,29,32,39,42,45,53,54	380
13	26,32,39,53,54,60,62	260
14	44,54,57,58,60,62	240
15	18,22,41,44,54,57,58,60	230
16	14,15,41,56,58	220
17	1 9,38,43,49	285
18	5,8,38,43,49,50,56	160
19	3,6,49,50	220
20	4,20,35,50	225
21	10,19,35	290
22		150
23		150
24		150
25	24	220
26		250

Tabel 4.2. Validasi

Interval	Units scheduled for maintenance	Nett reserve
1	10,22	210
2	15,19	210
3	2 6,45,62	200
4	1 3,31,46,62	250
5	31,58	340
6	19,26,38,56,61	385
7	7,25,56,59,61	325
8	4,48,49,59,61	255
9	14,28,41,48,49,54,56,61	275
10	32,37,48,49,54	450
11	28,32,36,37,51,52,54	420
12	24,36,51,52,53,54,60	370
13	,30,42,51,52,53,60	410
14	35,39,42,51,52,53,55,60	210
15	8,9,12,25,34,36,42,53,55,62	235
16	29,30,34,47,55,59,62	275
17	4,42,47,50,55,59	150
18	22,40,43,46,50,59	150
19	2,40,46,50,57	220
20	16,44,46,57	155
21	11,27,44,57	190
22	12,20	310
23		150
24		150
25	,59	210
26	5,33	200

Tabel 4.3. Jurnal

Pada tabel 4.2. dan tabel 4.3 ditunjukkan unit penjadwalan perawatan unit beserta besar kapasitas tersedia, dimana menurut program skripsi ini, total kapasitas tersedia adalah sebesar 6.475 MW sedang pada jurnal total kapasitas tersedia adalah sebesar 6.775 MW, berarti jurnal lebih besar 300 MW, jadi program ini mempunyai error sebesar 4,6 %. Dengan demikian program skripsi ini dianggap valid.

BAB V

PENUTUP

5.1. Kesimpulan

Dari hasil penjadwalan perawatan pada sistem tenaga dengan menggunakan metode *Genetic Algorithm* di PT. Pembangkitan Jawa – Bali, maka dapat diambil kesimpulan sebagai berikut :

1. Dengan menggunakan metode GA kita dapat menentukan periode kerja perawatan pada unit tenaga di PLTGU Gresik secara tepat, sehingga dapat mempertahankan cadangan sebesar 15.876.682 MWH.
2. Dengan memakai penjadwalan perawatan menggunakan metode GA pada sistem tenaga, disamping dapat lebih memperpanjang umur ekonomis peralatan juga mempertinggi keandalan peralatan, karena perawatan peralatan secara teratur dapat memperkecil Forced Outage Hours yang berarti peralatan dapat lebih diandalkan bagi kepentingan operasi.

5.2. Saran

Perlu adanya kerjasama yang baik antara PT.PJB dengan kalangan Pendidikan agar data dan informasi yang dibutuhkan dapat diperoleh dengan mudah yang memungkinkan ilmu pengetahuan dapat berkembang dengan leluasa tanpa adanya batasan.

DAFTAR PUSTAKA

1. **Wood, Allen J.,** Power Generation, Operation, and Control, 2nd ed, **John Wiley & Sons, Inc, New York, 1996.**
2. **Marsudi, Djiteng,** Operasi Sistem Tenaga Listrik, **Balai Penerbit & Humas ISTN, Jakarta, 1990.**
3. Genetic Algorithm For Maintenance Scheduling In Power Systems, **Michael Negnevitsky and Galina Kalareva, School of Engineering University of Tasmania.**
4. **Srikusuma Dewi** "Artficial Intelegence" **Graha Ilmu, 2003.**
5. **Goldberg, D.E.** Genetic Algorithms in Search, Optimisation and Machine Learning. **Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.**

LAMPIRAN-LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama	: HENRY ARDHIAN AKLISGANDHI
NIM	: 99.12.082
Jurusan	: Teknik Elektro ST
Konsentras	: Teknik Energi Listrik
Judul Skripsi	: PENJADWALAN PERAWATAN PADA SISTEM TENAGA DI PLTUG SEKTOR GRESIK DENGAN MENGGUNAKAN METODE GENETIK ALGORITMA

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1_

Hari	: Rabu
Tanggal	: 30 Maret 2005
Dengan Nilai	: 69, 165 (enam puluh sembilan koma seratus enam)

Panitia Ujian Skripsi

Ketua Majelis Penguji	Sekretaris Majelis Penguji
(Ir. Mochtar Asroni, MSME.)	(Ir. F. Yudi Limpraptono, MT.)
NIP. Y. 101 810 0036	NIP. Y. 103 950 0274

Anggota Penguji

Penguji Pertama	Penguji Kedua
(Ir. Made Wartana, MT.)	(Ir. Junior Siahaan)
NIP. 130 991 182	NIP. 102 890 0202

INSTITUT TEKNOLIGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : Henry Ardhian Aklisgandhi
Nim : 99.12.082
Masa Bimbingan : 27 September 2004 sampai dengan 27 Maret 2005

Judul Skripsi : Penjadwalan Perawatan Pada Sistem Tenaga di PT Pembangkit Jawa Bali Menggunakan *Genetic Algorithm*

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1	18-10-04	Bab I : 1. Apa semua referensi dalam Journal dan di cantumkan juga dalam Daftar Pustaka? 2. Banyak kesalahan yg tidak disengaja, mana copy Journal yg anda pakai	
2	04-11-04	Bab I : Perbaikan redaksional Bab II : segera di maksimalkan	
3	08-12-04	Bab II : Lanjutkan Perbaikan Redaksional	
4	17-03-05	Bab III : yg dibandingkan di Bab III berupa validasi program, tetapi pada Journal yg dipakai sebagai rujukan validasi Validasi program dicantumkan di Bab IV	
5	28-03-05	1. Untuk Validasi mana bandingan antara hasil program & hasil Journal? 2. Untuk aplikasi Real mana bandingan antara hasil Program & Data PJB?	
6		3. Bahan penjelasan tentang kode-kode jenis pembangkit 4. Kesimpulan saran - harus dilengkapi beberapa alur dengan redaksional	
7			
8			
9			
10			

Malang 31-03-2005
Dosen Pembimbing

(Ir. H. Almizan Abdullah, MSEE)

From.S-4b

```
{**********************************************************************
**********
 Genetic Algorithms for Maintenance Scheduling In Power System

 Date: 25/02/2005
 Programmer: $
 Depedencies:
  - File LoadsData.txt,CapsData.txt

 309278E0-86D2-11D9-9395-9D46D9540D39
                 '
**********************************************************************
*********}

unit MainFrm;

interface

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

const
  NUM_INTERVAL = 26;
  NUM_UNIT = 62;

type

  TChromo = record
    Fitness: Double;
    Bits: String;
  end;

  TGenePool = record
    TotalGenes: Integer;
    Genes: array of String;
  end;

  TMain = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    Help1: TMenuItem;
    About1: TMenuItem;
    Process1: TMenuItem;
    Run1: TMenuItem;
    Chart1: TChart;
    Splitter1: TSplitter;
    Series1: TLineSeries;
```

```
    Series2: TLineSeries;
    sbMain: TStatusBar;
    lvMain: TListView;
    N1: TMenuItem;
    Options1: TMenuItem;
    pbEpoch: TProgressBar;
    SaveAs1: TMenuItem;
    Open1: TMenuItem;
    N2: TMenuItem;
    procedure FormDestroy(Sender: TObject);
    procedure Run1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Options1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure SaveAs1Click(Sender: TObject);
    procedure Open1Click(Sender: TObject);
  private
    FBestChromo: TChromo;
    FLoadsData: array[1..NUM_INTERVAL] of Double;
    FCapsData: array[1..NUM_UNIT] of Double;
    FPops,FNewPops: array of TChromo;
    FBest,FAverage: array of Double;
    FGenePools: array[1..NUM_UNIT] of TGenePool;
    FTotalCaps: Double;
    FParams: TOptionsParams;

    procedure LoadsDataFromFiles;
    procedure InitPopulation;
    procedure CleanUp;

    function RandInt(Range: Integer): Integer;
    function Fitness(Chromo: TChromo): Double;
    procedure CrossOver(Parent1,Parent2: TChromo;var Child1,Child2: TChromo);
    function Mutate(Chromo: TChromo): TChromo;

    function Roulette(const TotalFitness: Double): TChromo;
    function CalculatePopulationFitness: Double;
    function FindBestFitness: Double;
    procedure Epoch(const NumGeneration: Integer);
    procedure ReplacePopulation;
    procedure InitChart;
    procedure DisplaySchedule;
    procedure DisplayCharts;

    procedure LoadData(const FileName: String);
    procedure SaveData(const FileName: String);

    // helpers
    function MakeGenePool(const GeneLength,NumBits: Integer;
```

```
        var Buffer: TGenePool): Integer;
    function MakeAdvancedGenePool(const GeneLength: Integer; var Buffer:
  TGenePool): Integer;
    procedure CheckAvailabilityConstraints(var Buffer: TGenePool;
        const MustAvailable: array of Integer);
  public
    { Public declarations }
  end;

  resourcestring
    APP_TITLE = 'Genetic Algorithms For Maintenance Scheduling In Power System';
    APP_IDLE  = 'Ready';
    APP_BUSY  = 'Please wait..';

var
  Main: TMain;

implementation

{$R *.DFM}

uses Math, AboutFrm;

{ TMain }

procedure TMain.InitPopulation;
var
  i,j,index : Integer;
  chromo: TChromo;

begin
  LoadsDataFromFiles;

  FTotalCaps := Sum(FCapsData);
  sbMain.Panels[3].Text := Format('Capacity %5.1f MW',[FTotalCaps]);

  CleanUp;

  // general initialization
  for i := 1 to NUM_UNIT do
  begin
    case i of
    1..30: MakeGenePool(NUM_INTERVAL,1,FGenePools[i]);
    31..45: MakeGenePool(NUM_INTERVAL,2,FGenePools[i]);
    46..50,56..60: MakeGenePool(NUM_INTERVAL,3,FGenePools[i]);
    51..55,61: MakeGenePool(NUM_INTERVAL,4,FGenePools[i]);
    end;
  end;
  // #62 Unit specific
  MakeAdvancedGenePool(NUM_INTERVAL,FGenePools[NUM_UNIT]);
```

```
  // specific initialization
  for i := 1 to NUM_UNIT do
  begin
   case i of
   1..5,15..20,25:
    CheckAvailabilityConstraints(FGenePools[i],[7,8,9,10,11,12,13]);
   50..60:
    CheckAvailabilityConstraints(FGenePools[i],[23,24,25,26]);
   61..62:
    CheckAvailabilityConstraints(FGenePools[i],[21,22,23,24,25,26]);
   end;
  end;


  FPops := nil;
  FNewPops := nil;

  Setlength(FPops,FParams.PopCount);
  Setlength(FNewPops,FParams.PopCount);

  lvMain.Items.BeginUpdate;
  try
   lvMain.Items.Clear;
  finally
   lvMain.Items.EndUpdate;
  end;

  for i := Low(FPops) to High(FPops) do
  begin
   chromo.Fitness := 0;
   chromo.Bits := '';
   for j := 1 to NUM_UNIT do
   begin
    index := RandInt(FGenePools[j].TotalGenes)-1;
    chromo.Bits := chromo.Bits + FGenePools[j].Genes[index];
   end;
   FPops[i] := chromo;
  end;

  // init progressbar
  pbEpoch.Max := FParams.EpochCount;
end;

function TMain.RandInt(Range: Integer): Integer;
begin
  Result := 1 + Random(Range);
end;

procedure TMain.FormDestroy(Sender: TObject);
begin
```

```
    FPops := nil;
    FNewPops := nil;
    FBest := nil;
    FAverage := nil;
  end;

  // calculate fitness
  // sources: Paper e.q. (1,2,3,4)
  function TMain.Fitness(Chromo: TChromo): Double;
  var
    i,j : Integer;
    tmp : Double;
    bits: String;
    R : array[1..NUM_INTERVAL] of Double;
  begin
    for i := 1 to NUM_INTERVAL do
    begin
      tmp := 0.0;
      bits := Chromo.Bits;
      for j := 1 to NUM_UNIT do
      begin
        tmp := tmp + FCapsData[j]*(Ord(Chromo.Bits[(j-1)*NUM_INTERVAL + i])-
Ord('0'));
      end;
      R[i] := FTotalCaps - FLoadsData[i] - tmp;
    end;
    Result := Max(MinValue(R),0);
  end;

  procedure TMain.CrossOver(Parent1, Parent2: TChromo; var Child1,
    Child2: TChromo);
  var
    index: Integer;
  begin
    index := (RandInt(NUM_UNIT)-1)*NUM_INTERVAL + 1;
    Child1 := Parent1;
    Child2 := Parent2;

    if index > 1 then
    begin
      Child1.Bits := Copy(Parent1.Bits,1,index-1) +
                Copy(Parent2.Bits,index,NUM_INTERVAL*NUM_UNIT);
      Child2.Bits := Copy(Parent2.Bits,1,index-1) +
                Copy(Parent1.Bits,index,NUM_INTERVAL*NUM_UNIT);
    end;
  end;

  function TMain.Mutate(Chromo: TChromo): TChromo;
  var
    index: Integer;
```

```
  newgene: String;
  tmp: String;
 begin
  tmp := Chromo.Bits;
  index := RandInt(NUM_UNIT);
  newgene := FGenePools[index].genes[RandInt(FGenePools[index].TotalGenes)-1];
  index := (index-1)*NUM_INTERVAL + 1;
  Delete(tmp,index,NUM_INTERVAL);
  Insert(newgene,tmp,index);
  Result.Bits := tmp;
end;

procedure TMain.Epoch(const NumGeneration: Integer);
var
 i : Integer;
 n: Integer;
 AverageFitness,TotalFitness,BestFitness: Double;
 child1,child2,parent1,parent2 : TChromo;
begin
 InitPopulation;

 try
  pbEpoch.Visible := true;
  with lvMain.Items.Add do SubItems.Add(APP_BUSY);

  Application.ProcessMessages;
  for i := 1 to NumGeneration do
  begin
   TotalFitness := CalculatePopulationFitness;
   BestFitness := FindBestFitness;
   AverageFitness := TotalFitness/FParams.PopCount;
   FBest[i-1] := BestFitness;
   FAverage[i-1] := AverageFitness;

   pbEpoch.Position := i;

   n := Low(FPops);

   while n < High(FPops) do
   begin
    parent1 := Roulette(TotalFitness);
    parent2 := Roulette(TotalFitness);
    child1  := parent1;
    child2  := parent2;
    if Random < FParams.Pc then
     CrossOver(parent1,parent2,child1,child2);

    if Random < FParams.Pm then
    begin
     child1 := Mutate(child1);
```

```
          child2 := Mutate(child2);
        end;
        FNewPops[n] := child1;
        Inc(n);
        FNewPops[n] := child2;
        Inc(n);
      end;
      ReplacePopulation;
    end;

  finally
    pbEpoch.Visible := false;
    pbEpoch.Position := 0;
    sbMain.Panels[0].Text := APP_IDLE;
  end;

  DisplaySchedule;
  DisplayCharts;
end;

// return total population fitness
function TMain.CalculatePopulationFitness: Double;
var
  i: Integer;
begin
  Result := 0;
  for i := Low(FPops) to High(FPops) do
  begin
    FPops[i].Fitness := Fitness(FPops[i]);
    Result := Result + FPops[i].Fitness;
  end;
end;

function TMain.FindBestFitness: Double;
var
  i : Integer;
begin
  Result := 0;
  FBestChromo := FPops[Low(FPops)];
  for i := Low(FPops) to High(FPops) do
  begin
    if FPops[i].Fitness > Result then
    begin
      Result := FPops[i].Fitness;
      FBestChromo := FPops[i];
    end;
  end;
end;

procedure TMain.ReplacePopulation;
```

```
  var
    i : Integer;
  begin
    for i := Low(FPops) to High(FPops) do
      FPops[i] := FNewPops[i];
  end;

  function TMain.Roulette(const TotalFitness: Double): TChromo;
  var
    Fitness,slice: Double;
    i : Integer;
  begin
    slice := Random*TotalFitness;
    fitness := 0;
    Result := FPops[Low(FPops)];

    for i := Low(FPops) to High(FPops) do
    begin
      fitness := fitness + FPops[i].Fitness;
      if fitness >= slice then
      begin
        Result := FPops[i];
        Break;
      end;
    end;
  end;

  procedure TMain.Run1Click(Sender: TObject);
  begin
    FBest := nil;
    FAverage := nil;
    SetLength(FBest,FParams.EpochCount);
    SetLength(FAverage,FParams.EpochCount);
    Epoch(FParams.EpochCount);
  end;

  procedure TMain.FormCreate(Sender: TObject);
  begin
    pbEpoch.Visible := false;
    pbEpoch.Parent := sbMain;
    pbEpoch.SetBounds(2,2,345,16);
    Caption := APP_TITLE;
    sbMain.Panels[0].Text := APP_IDLE;
    Randomize;
    FParams.Pm := 0.01;
    FParams.Pc := 0.7;
    FParams.EpochCount := 100;
    FParams.PopCount := 100;

    InitChart;
```

```
end;

procedure TMain.InitChart;
begin
 with Chart1 do
 begin
  Title.Text.Text := Format('%d Chromosomes %d Generation Pc = %5.4f Pm =
%5.4f',

[FParams.PopCount,FParams.EpochCount,FParams.Pc,FParams.Pm]);
  BottomAxis.Title.Caption := Format('Number of units = %d Number of itervals =
%d',
                          [NUM_UNIT,NUM_INTERVAL]);
  LeftAxis.Title.Caption := 'Fitness';
 end;
end;

procedure TMain.DisplaySchedule;
var
 i,j,x,n : Integer;
 tmp : Double;
 bits: String;
 R : array[1..NUM_INTERVAL] of Double;
 S : String;
begin
 lvMain.Items.BeginUpdate;
 try
  lvMain.Items.Clear;
  for i := 1 to NUM_INTERVAL do
  begin
   tmp := 0.0;
   S := '';
   bits := FBestChromo.Bits;
   n := 1;
   for j := 1 to NUM_UNIT do
   begin
    x := (Ord(bits[(j-1)*NUM_INTERVAL + i])-Ord('0'));
    tmp := tmp + FCapsData[j]*x;
    if x = 1 then
    begin
     if n = 1 then S := IntToStr(j)
     else S := S + ',' + IntToStr(j);
     n := 0;
    end;
   end;
   R[i] := FTotalCaps - FLoadsData[i] - tmp;

   with lvMain.Items.Add do
   begin
    Caption := IntToStr(i);
```

```
        SubItems.Add(S);
        SubItems.Add(FloatToStr(R[i]));
       end;
     end;
    finally
     lvMain.Items.EndUpdate;
    end;
  end;

  procedure TMain.Options1Click(Sender: TObject);
  begin
   if ShowOptions(FParams) then
    InitChart;
  end;

  procedure TMain.Exit1Click(Sender: TObject);
  begin
   Close;
  end;

  procedure TMain.LoadsDataFromFiles;
  var
   SL: TStrings;
   i: Integer;
  begin
   SL := TStringList.Create;
   try
    SL.LoadFromFile(ExtractFilePath(Application.Exename)+ 'LoadsData.dat');
    if SL.Count <> NUM_INTERVAL then
     raise Exception.Create('Error in loads file (wrong number of data)');
    for i := 0 to SL.Count-1 do
     FLoadsData[i+1] := StrToInt(SL[i]);
    SL.Clear;
    SL.LoadFromFile(ExtractFilePath(Application.Exename)+ 'CapsData.dat');
    if SL.Count <> NUM_UNIT then
     raise Exception.Create('Error in capacity file (wrong number of data)');
    for i := 0 to SL.Count-1 do
     FCapsData[i+1] := StrToInt(SL[i]);

    FTotalCaps := Sum(FCapsData);
   finally
    SL.Free;
   end;
  end;

  // notes:
  // Each unit has a pool of genes that varies in size
  // due to validation contraints that genes must follow
  function TMain.MakeGenePool(const GeneLength, NumBits: Integer;
   var Buffer: TGenePool): Integer;
```

```
  var
   bits: String;
   i,N : Integer;

  begin
   N := GeneLength-NumBits + 1;
   Buffer.TotalGenes := N;
   SetLength(Buffer.Genes,N);
   SetLength(bits,GeneLength);
   FillMemory(PChar(bits),GeneLength,Ord('0'));
   FillMemory(PChar(bits),NumBits,Ord('1'));

   for i:=1 to N do
   begin
    Buffer.Genes[i-1] := Copy(bits,1,GeneLength);//dynamic array start from 0
    MoveMemory(PChar(bits)+1,PChar(bits),GeneLength-1);
    FillMemory(PChar(bits),1,Ord('0'));
   end;

   Result := N;
  end;

  procedure TMain.CleanUp;
  var
   i: Integer;
  begin
   for i := Low(FGenePools) to High(FGenePools) do
   begin
    FGenePools[i].Genes := nil;
    FGenePools[i].TotalGenes := 0;
   end;
  end;

  procedure TMain.About1Click(Sender: TObject);
  begin
   ShowAbout;
  end;


  // advanced constraints
  procedure TMain.CheckAvailabilityConstraints(var Buffer: TGenePool;
   const MustAvailable: array of Integer);
  var
   i,j,p,gen_len: Integer;
   bits: String;
   Done : Boolean;

   // Am I in an array ?
   function InArray(const Value: Integer;const Values: array of Integer): Boolean;
   var
```

```
    i : Integer;
   begin
    Result := false;
    for i := Low(Values) to High(Values) do
    begin
     Result := Value = Values[i];
     if Result then Break;
    end;
   end;

   // Am I maintained this time ?
   function Scheduled(const Interval: Integer;const Gene: String): Boolean;
   var
    i : Integer;
   begin
    Result := false;
    for i := 1 to Length(Gene) do
    begin
     // if scheduled at interval i
     Result := (Gene[i] = '1') and (Interval = i);
     if Result then Break;
    end;
   end;

begin
  // buffer is per unit's gene pool
  for i := Low(Buffer.Genes) to High(Buffer.Genes) do
  begin
   bits := Buffer.Genes[i];//it's a gene
   gen_len := Length(Bits);
   for j := Low(MustAvailable) to High(MustAvailable) do
   begin
    // Scheduled is true when it's scheduled for maintenance
    // if it's scheduled then move the schedule to another vacant interval
    // notes:
    // j starts with 0 where bits starts with 1 (reguler string)
    if Scheduled(MustAvailable[j],bits) then
    begin
     // set it to '0' -> operating for that interval
     bits[MustAvailable[j]] := '0';
     // then find another feasible interval for maintenance
     repeat
      p := RandInt(gen_len);
      Done := (bits[p] = '0') and (not InArray(p,MustAvailable));
      if Done then bits[p] := '1'; // ok, found ..scheduled it for maintenance
     until Done;
     Buffer.Genes[i] := bits;//re-assigned to gene pool
    end;
   end;
  end;
```

```
end;

function TMain.MakeAdvancedGenePool(const GeneLength: Integer; var Buffer:
TGenePool): Integer;
var
 bits: String;
 i,N : Integer;

begin
 N := GeneLength-6;
 SetLength(bits,GeneLength);
 Buffer.TotalGenes := N-4;
 SetLength(Buffer.Genes,N-4);
 FillMemory(PChar(bits),GeneLength,Ord('0'));

 // interval 21-26 must be reserved (in operating state)
 i := 0;

 while i < N-4 do
 begin
  FillMemory(PChar(bits),N,Ord('0'));
  FillMemory(PChar(bits) + i,2,Ord('1'));
  FillMemory(PChar(bits) + i + 3,2,Ord('1'));
  Buffer.Genes[i] := Copy(bits,1,GeneLength);
  Inc(i);
 end;

 Result := N;
end;

procedure TMain.DisplayCharts;
var
 i : Integer;
begin
 Series1.Clear;// best
 Series2.Clear;// average

 for i := 1 to FParams.EpochCount-1 do
 begin
  Series1.AddXY(i,FBest[i-1]);
  Series2.AddXY(i,FAverage[i-1]);
 end;
end;

procedure TMain.LoadData(const FileName: String);
var
 i: Integer;
 best,avg: Double;
 F: TextFile;
begin
```

```
  AssignFile(F,FileName);
  Reset(F);

  try
   try
    ReadLn(F,FParams.Pm,FParams.Pc);// Pm, Pc
    Readln(F,FParams.EpochCount,FParams.PopCount); // epoch count
    ReadLn(F,FBestChromo.Bits,FBestChromo.Fitness);
    if FParams.EpochCount > 0 then
    begin
     FBest := nil;
     FAverage := nil;
     SetLength(FBest,FParams.EpochCount);
     SetLength(FAverage,FParams.EpochCount);
     for i := 0 to FParams.EpochCount-1 do
     begin
      ReadLn(F,best,avg);
      FBest[i] := best;
      FAverage[i] := avg;
     end;

     InitChart;
     DisplaySchedule;
     DisplayCharts;
    end;

   except
    FParams.Pm := 0.7;
    FParams.Pc := 0.001;
    FParams.EpochCount := 1;
    FParams.PopCount := 1;
   end;
  finally
   CloseFile(F);
  end;
end;

procedure TMain.SaveData(const FileName: String);
var
 i: Integer;
 F: TextFile;
begin
 AssignFile(F,FileName);
 Rewrite(F);

 try
  WriteLn(F,FParams.Pm,FParams.Pc);// Pm, Pc
  Writeln(F,FParams.EpochCount,' ',FParams.PopCount); // epoch count
  WriteLn(F,FBestChromo.Bits,FBestChromo.Fitness);
  if FParams.EpochCount > 0 then
```

```
    begin
      for i := 0 to FParams.EpochCount-1 do
      begin
        WriteLn(F,FBest[i],FAverage[i]);
      end;
    end;
  finally
    CloseFile(F);
  end;
end;

procedure TMain.SaveAs1Click(Sender: TObject);
var
  dlg: TSaveDialog;
begin
  dlg := TSaveDialog.Create(Self);
  try
    dlg.Title := 'Save data to a file';
    dlg.DefaultExt := 'txt';
    dlg.Filter := 'Text Files|*.txt|All Files|*.*';
    if dlg.Execute then
      SaveData(dlg.FileName);
  finally
    dlg.Free;
  end;
end;

procedure TMain.Open1Click(Sender: TObject);
var
  dlg: TOpenDialog;
begin
  dlg := TOpenDialog.Create(Self);
  try
    LoadsDataFromFiles;
    dlg.Title := 'Open Data';
    dlg.DefaultExt := 'txt';
    dlg.Filter := 'Text Files|*.txt|All Files|*.*';
    if dlg.Execute then
      LoadData(dlg.FileName);
  finally
    dlg.Free;
  end;
end;

end.
```

```
unit OptionsFrm;

interface

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

type
 TOptionsParams = record
  Pm,Pc: Double;
  EpochCount,PopCount: Integer;
 end;

 TOptions = class(TForm)
  Button1: TButton;
  Button2: TButton;
  txtPC: TEdit;
  Label1: TLabel;
  Label2: TLabel;
  txtPM: TEdit;
  Label3: TLabel;
  txtPop: TEdit;
  Label4: TLabel;
  txtGener: TEdit;
 private
  function GetParams: TOptionsParams;
  procedure SetParams(const Value: TOptionsParams);
  { Private declarations }
 public
  property Params: TOptionsParams read GetParams write SetParams;
 end;


 function ShowOptions(var AParams: TOptionsParams): Boolean;

implementation

{$R *.DFM}


function ShowOptions(var AParams: TOptionsParams): Boolean;
begin
 with TOptions.Create(Application) do
 try
  Params := AParams;
  Result := false;
  if ShowModal = mrOK then
  begin
   Result := true;
```

```
      AParams := Params;
    end;
   finally
    Free;
   end;
  end;

 { TOptions }

 function TOptions.GetParams: TOptionsParams;
 begin
  with Result do
  begin
   Pm := StrToFloat(txtPm.Text);
   Pc := StrToFloat(txtPC.Text);
   EpochCount := StrToInt(txtGener.Text);
   PopCount := StrToInt(txtPop.Text);
  end;
 end;

 procedure TOptions.SetParams(const Value: TOptionsParams);
 begin
  with Value do
  begin
   txtPC.Text := FloatToStr(Pc);
   txtPM.Text := FloatToStr(Pm);
   txtGener.Text := IntToStr(EpochCount);
   txtPop.Text := IntToStr(PopCount);
  end;
 end;

 end.
```

```
unit AboutFrm;

interface

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

type
  TAbout = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Bevel1: TBevel;
    Label2: TLabel;
    Label3: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;


  procedure ShowAbout;

implementation

{$R *.DFM}

procedure ShowAbout;
begin
  with TAbout.Create(Application) do
  try
    ShowModal;
  finally
    Free;
  end;
end;

end.
```

```
unit uUtils;

interface

uses SysUtils;

type
 TSort=(asc,dec);

 TBatas=record
  min,max:double;
 end;

 TBatasArr1=array of TBatas;
 TBatasArr2=array of array of TBatas;

 dArr1=array of double;
 dArr2=array of array of double;
 iArr1=array of integer;
 iArr2=array of array of integer;
 bArr1=array of boolean;
 bArr2=array of array of boolean;
 sArr1=array of String;

 TAlleleTCSC=record
  Lokasi,TypeAlat,Setting:double;
 end;

 TChromTCSC1=array of TAlleleTCSC;

 TAlleleUpfc=record
  Lokasi,TypeAlat,Tap,Sudut:double;
 end;

 TChromUpfc1=array of TAlleleUpfc;

 TAlleleMain=record
  main:integer;
  stat:boolean;
 end;

 TChromMain1=array of TAlleleMain;
 TChromMain2=array of array of TAlleleMain;

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

 procedure Swap(var X,Y:byte);overload;
 procedure Swap(var X,Y:integer);overload;
```

```
  procedure Swap(var X,Y:word);overload;
  procedure Swap(var X,Y:double);overload;
  procedure Swap(var X,Y:extended);overload;
  procedure Swap(var X,Y:string);overload;
  procedure Swap(var X,Y:boolean);overload;

  procedure BubleSort(var aData:dArr1;const aType:TSort);overload;
  procedure BubleSort(var aData:iArr1;const aType:TSort);overload;
  procedure BubleSort(var aData:sArr1;const aType:TSort);overload;

  function DecodeBinToFloat1(const aData:bArr1):double;
  function DecodeBinToFloat2(const aData:bArr2):dArr1;

  function GetBatas(const aValue,aMin,aMax:double):double;
  function GetFlip(const aFlip:double):boolean;

  function GetBatasToReal(const aValue,aMin,aMax:double):double;
  function GetRealToBatas(const aValue,aMin,aMax:double):double;

implementation

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

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

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

procedure Swap(var X,Y:byte);
var tmp:byte;
begin
 tmp:=X;
 X:=Y;
 Y:=tmp;
end;
```

```
procedure Swap(var X,Y:integer);
var tmp:integer;
begin
 tmp:=X;
 X:=Y;
 Y:=tmp;
end;

procedure Swap(var X,Y:word);
var tmp:word;
begin
 tmp:=X;
 X:=Y;
 Y:=tmp;
end;

procedure Swap(var X,Y:double);
var tmp:double;
begin
 tmp:=X;
 X:=Y;
 Y:=tmp;
end;

procedure Swap(var X,Y:extended);
var tmp:extended;
begin
 tmp:=X;
 X:=Y;
 Y:=tmp;
end;

procedure Swap(var X,Y:string);
var tmp:string;
begin
 tmp:=X;
 X:=Y;
 Y:=tmp;
end;

procedure Swap(var X,Y:boolean);
var tmp:boolean;
begin
 tmp:=X;
 X:=Y;
 Y:=tmp;
end;

procedure BubleSort(var aData:dArr1;const aType:TSort);
```

```
  var i,j:integer;
  begin
   for i:=1 to (high(aData)-1) do
   begin
    for j:=i to high(aData) do
    begin
     if aType=asc then
     begin
      if aData[i]>aData[j] then
      begin
       Swap(aData[i],aData[jJ);
      end;
     end
     else if aType=dec then
     begin
      if aData[i]<aData[j] then
      begin
       Swap(aData[i],aData[j]);
      end;
     end;
    end;
   end;
  end;

  procedure BubleSort(var aData:iArr1;const aType:TSort);
  var i,j:integer;
  begin
   for i:=1 to (high(aData)-1) do
   begin
    for j:=i to high(aData) do
    begin
     if aType=asc then
     begin
      if aData[i]>aData[j] then
      begin
       Swap(aData[i],aData[j]);
      end;
     end
     else if aType=dec then
     begin
      if aData[i]<aData[j] then
      begin
       Swap(aData[i],aData[j]);
      end;
     end;
    end;
   end;
  end;

  procedure BubleSort(var aData:sArr1;const aType:TSort);
```

```
  var i,j:integer;
  begin
   for i:=1 to (high(aData)-1) do
   begin
    for j:=i to high(aData) do
    begin
     if aType=asc then
     begin
      if aData[i]>aData[j] then
      begin
       Swap(aData[i],aData[j]);
      end;
     end
     else if aType=dec then
     begin
      if aData[i]<aData[j] then
      begin
       Swap(aData[i],aData[j]);
      end;
     end;
    end;
   end;
  end;

  function DecodeBinToFloat1(const aData:bArr1):double;
  var i:integer;
     powerof2,sa:double;
  begin
   result:=0;
   powerof2:=1;
   sa:=pangkat(2,high(aData))-1;
   for i:=high(aData) downto 1 do
   begin
    if aData[i]=true then
    begin
     result:=result+powerof2;
    end;
    powerof2:=powerof2*2;
   end;
   result:=result/sa;
  end;

  function DecodeBinToFloat2(const aData:bArr2):dArr1;
  var i,j:integer;
    Data:bArr1;
    nnData:bArr2;
    NData,NLength:integer;
  begin
   NData:=high(aData)+1;
   NLength:=high(aData[0])+1;
```

```
  SetLength(Data,NLength+1);
  SetLength(result,NData+1);
  SetLength(nnData,NData+1,NLength+1);
  for i:=0 to NData-1 do
  begin
   for j:=0 to NLength-1 do
   begin
    nnData[i+1,j+1]:=aData[i,j];
   end;
  end;
  for i:=1 to NData do
  begin
   for j:=1 to NLength do
   begin
    Data[j]:=nnData[i,j];
   end;
   result[i]:=DecodeBinToFloat1(Data);
  end;
end;

function GetBatas(const aValue,aMin,aMax:double):double;
begin
 if aValue>1.0 then raise Exception.Create('Value tidak boleh lebih dari 1');
 if aValue<0.0 then raise Exception.Create('Value tidak boleh kurang dari 0');
 result:=aMin+aValue*(aMax-aMin);
end;

function GetFlip(const aFlip:double):boolean;
begin
 result:=false;
 if random<=aFlip then result:=true;
end;

function GetBatasToReal(const aValue,aMin,aMax:double):double;
begin
 result:=aMin+aValue*(aMax-aMin);
end;

function GetRealToBatas(const aValue,aMin,aMax:double):double;
begin
 result:=(aValue-aMin)/(aMax-aMin);
end;

end.
```

```
unit uInput;

interface

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

type
  TfrmInput = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    Panel1: TPanel;
    TabSheet2: TTabSheet;
    btnClose: TButton;
    btnNext: TButton;
    Label1: TLabel;
    edtNMain: TEdit;
    Label2: TLabel;
    edtNGen: TEdit;
    Label3: TLabel;
    edtNjam: TEdit;
    SaveDialog1: TSaveDialog;
    fgMaintenance: TStringGrid;
    TabSheet3: TTabSheet;
    TabSheet4: TTabSheet;
    fgBeban: TStringGrid;
    fgGen: TStringGrid;
    procedure btnNextClick(Sender: TObject);
    procedure edtNMainChange(Sender: TObject);
    procedure edtNGenChange(Sender: TObject);
    procedure edtNjamChange(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmInput: TfrmInput;

implementation

uses uParamGA;

{$R *.dfm}

procedure TfrmInput.btnNextClick(Sender: TObject);
```

```
var input:TextFile;
   NamaFile,Nama:string;
   i,j,NMain,kode,NPeriode,NGen,interval,main1,main2,main3,main4:integer;
   load,kap:double;
begin
 if btnNext.Caption='&Save' then
 begin
  if SaveDialog1.Execute then
  begin
   NamaFile:=SaveDialog1.FileName;
   AssignFile(input,NamaFile+'.txt');
   Rewrite(input);
   NMain:=StrToInt(edtNMain.Text);
   NGen:=StrToInt(edtNGen.Text);
   NPeriode:=StrToInt(edtNjam.Text);
   Writeln(input,NMain);
   Writeln(input,NGen);
   Writeln(input,NPeriode);
   for i:=1 to NMain do
   begin
    Nama:=fgMaintenance.Cells[1,i];
    interval:=StrToInt(fgMaintenance.Cells[2,i]);
    kode:=StrToInt(fgMaintenance.Cells[3,i]);
    main1:=StrToInt(fgMaintenance.Cells[4,i]);
    main2:=StrToInt(fgMaintenance.Cells[5,i]);
    main3:=StrToInt(fgMaintenance.Cells[6,i]);
    main4:=StrToInt(fgMaintenance.Cells[7,i]);
    Writeln(input,interval,' ',kode,' ',main1,' ',main2,' ',
    main3,' ',main4,' ',Nama);
   end;
   for i:=1 to Ngen do
   begin
    Nama:=fgGen.Cells[1,i];
    main1:=StrToInt(fgGen.Cells[2,i]);
    main2:=StrToInt(fgGen.Cells[3,i]);
    main3:=StrToInt(fgGen.Cells[4,i]);
    main4:=StrToInt(fgGen.Cells[5,i]);
    kap:=StrToFloat(fgGen.Cells[6,i]);
    Writeln(input,main1,' ',main2,' ',main3,' ',main4,' ',
    kap:9:1,' ',Nama);
   end;
   for i:=1 to Ngen do
   begin
    for j:=1 to NPeriode do
    begin
     Load:=StrToFloat(fgBeban.Cells[j,i]);
     Write(input,Load:9:2,' ');
    end;
    Writeln(input,'');
   end;
```

```
      CloseFile(input);
      MessageDlg('File berhasil disimpan!',mtInformation,[mbOK],0);
     end;
    end
    else if btnNext.Caption='&Next' then
    begin
     frmParamGA.edtNParam.Text:=edtNGen.Text;
     frmParamGA.edtLengthChrom.Text:=edtNjam.Text;
     frmParamGA.Show;
    end;
  end;

  procedure TfrmInput.edtNMainChange(Sender: TObject);
  var i,N:integer;
  begin
   if edtNMain.Text='' then
   begin
    fgMaintenance.RowCount:=2;
   end
   else
   begin
    N:=StrToInt(edtNMain.Text);
    fgMaintenance.RowCount:=N+1;
    for i:=1 to N do
    begin
     fgMaintenance.Cells[0,i]:=IntToStr(i);
    end;
   end;
  end;

  procedure TfrmInput.edtNGenChange(Sender: TObject);
  var i,j,N:integer;
  begin
   if edtNGen.Text='' then
   begin
    fgGen.RowCount:=2;
   end
   else
   begin
    N:=StrToInt(edtNGen.Text);
    fgGen.RowCount:=N+1;
    fgBeban.RowCount:=N+1;
    for i:=1 to N do
    begin
     fgGen.Cells[0,i]:=IntToStr(i);
     fgBeban.Cells[0,i]:='Unit '+IntToStr(i);
    end;
    if edtNjam.Text<>'' then
    begin
     for i:=1 to N do
```

```
      begin
       for j:=1 to StrToInt(edtNjam.Text) do
       begin
        fgBeban.Cells[j,i]:='0';
       end;
      end;
    end;
  end;
end;

procedure TfrmInput.edtNjamChange(Sender: TObject);
var i,j,N:integer;
begin
 if edtNjam.Text='' then
 begin
  fgBeban.ColCount:=2;
 end
 else
 begin
  N:=StrToInt(edtNjam.Text);
  fgBeban.ColCount:=N+1;
  for i:=1 to N do
  begin
   fgBeban.Cells[i,0]:='Per '+IntToStr(i);
  end;
  if edtNGen.Text<>'' then
  begin
   for i:=1 to StrToInt(edtNgen.Text) do
   begin
    for j:=1 to N do
    begin
     fgBeban.Cells[j,i]:='0';
    end;
   end;
  end;
 end;
end;

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

procedure TfrmInput.FormCreate(Sender: TObject);
begin
 fgMaintenance.Cells[0,00]:='No';
 fgMaintenance.Cells[1,0]:='Nama';
 fgMaintenance.Cells[2,0]:='Interval';
 fgMaintenance.Cells[3,0]:='Kode';
 fgMaintenance.Cells[4,0]:='Main 1';
```

```
  fgMaintenance.Cells[5,0]:='Main 2';
  fgMaintenance.Cells[6,0]:='Main 3';
  fgMaintenance.Cells[7,0]:='Main 4';
  fgGen.Cells[0,0]:='No';
  fgGen.Cells[1,0]:='Nama';
  fgGen.Cells[2,0]:='Main 1';
  fgGen.Cells[3,0]:='Main 2';
  fgGen.Cells[4,0]:='Main 3';
  fgGen.Cells[5,0]:='Main 4';
  fgGen.Cells[6,0]:='Kapasitas';
end;

end.
```

```
unit uMenu;

interface

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

type
  TfrmMenu = class(TForm)
    Panel1: TPanel;
    btnNew: TButton;
    btnOpen: TButton;
    btnExit: TButton;
    StatusBar1: TStatusBar;
    Panel2: TPanel;
    OpenDialog1: TOpenDialog;
    procedure btnExitClick(Sender: TObject);
    procedure btnOpenClick(Sender: TObject);
    procedure btnNewClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmMenu: TfrmMenu;

implementation

uses uUtils, uInput, uObjFunc;

{$R *.dfm}

procedure TfrmMenu.btnExitClick(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TfrmMenu.btnOpenClick(Sender: TObject);
var NamaFile,Nama:string;
    output:TextFile;
    kap,load:double;
    i,j,NMain,Ngen,NPeriode,interval,kode,main1,main2,main3,main4:integer;
begin
  try
    if OpenDialog1.Execute then
    begin
      NamaFile:=OpenDialog1.FileName;
```

```
  AssignFile(output,NamaFile);
  Reset(output);
  Readln(output,NMain);
  Readln(output,Ngen);
  Readln(output,NPeriode);
  frmInput.edtNMain.Text:=IntToStr(NMain);
  frmInput.edtNGen.Text:=IntToStr(Ngen);
  frmInput.edtNjam.Text:=IntToStr(NPeriode);
  frmInput.fgMaintenance.RowCount:=NMain+1;
  SetLength(gMain,NMain+1);
  for i:=1 to NMain do
  begin
   Readln(output,interval,kode,main1,main2,main3,main4,Nama);
   frmInput.fgMaintenance.Cells[0,i]:=IntToStr(i);
   frmInput.fgMaintenance.Cells[1,i]:=Nama;
   frmInput.fgMaintenance.Cells[2,i]:=IntToStr(interval);
   frmInput.fgMaintenance.Cells[3,i]:=IntToStr(kode);
   frmInput.fgMaintenance.Cells[4,i]:=IntToStr(main1);
   frmInput.fgMaintenance.Cells[5,i]:=IntToStr(main2);
   frmInput.fgMaintenance.Cells[6,i]:=IntToStr(main3);
   frmInput.fgMaintenance.Cells[7,i]:=IntToStr(main4);
   gMain[i].nama:=Nama;
   gMain[i].interval:=interval;
   gMain[i].kode:=kode;
   gMain[i].main1:=main1;
   gMain[i].main2:=main2;
   gMain[i].main3:=main3;
   gMain[i].main4:=main4;
  end;
  frmInput.fgGen.RowCount:=Ngen+1;
  SetLength(gGen,Ngen+1);
  for i:=1 to Ngen do
  begin
   Readln(output,main1,main2,main3,main4,kap,Nama);
   frmInput.fgGen.Cells[0,i]:=IntToStr(i);
   frmInput.fgGen.Cells[1,i]:=Nama;
   frmInput.fgGen.Cells[2,i]:=IntToStr(main1);
   frmInput.fgGen.Cells[3,i]:=IntToStr(main2);
   frmInput.fgGen.Cells[4,i]:=IntToStr(main3);
   frmInput.fgGen.Cells[5,i]:=IntToStr(main4);
   frmInput.fgGen.Cells[6,i]:=FloatToStr(kap);
   gGen[i].nama:=Nama;
   gGen[i].main1:=main1;
   gGen[i].main2:=main2;
   gGen[i].main3:=main3;
   gGen[i].main4:=main4;
   gGen[i].kap:=kap;
  end;
  frmInput.fgBeban.RowCount:=Ngen+1;
  frmInput.fgBeban.ColCount:=NPeriode+1;
```

```
unit uAbout;

interface

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

type
  TfrmAbout = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmAbout: TfrmAbout;

implementation

{$R *.dfm}

end.
```