APLIKASI *DYNAMIC PROGRAMMING* UNTUK PENJADWALAN PEMBANGKIT TENAGA LISTRIK

SKRIPSI

KONSENTRASI TEKNIK ENERGI ELEKTRIK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh:

GIGIH SANJAYA S D

NIM. 0510630044 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MALANG

2012

LEMBAR PERSETUJUAN

APLIKASI *DYNAMIC PROGRAMMING* UNTUK PENJADWALAN PEMBANGKIT TENAGA LISTRIK

SKRIPSI

KONSENTRASI TEKNIK ENERGI ELEKTRIK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh:

GIGIH SANJAYA S D

NIM. 0510630044 - 63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Hadi Suyono, S.T., M.T., Ph.D.	**Rini Nur Hasanah, Dr., S.T., M.Sc.**
NIP. 19730520 200801 1 013	**NIP. 19680122 199512 2 001**

LEMBAR PENGESAHAN

APLIKASI *DYNAMIC PROGRAMMING* UNTUK PENJADWALAN PEMBANGKIT TENAGA LISTRIK

Disusun Oleh:
GIGIH SANJAYA S D
NIM. 0510630044 - 63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 10 Agustus 2012

Dosen Penguji

Ir. Teguh Utomo, MT.
NIP. 19650913 199103 1 003

Ir. Hari Santoso, MS.
NIP. 19531205 198503 1 001

Ir. Unggul Wibawa, M.Sc.
NIP. 19630106 198802 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Dr. Ir. Sholeh Hadi Pramono, MS.
NIP. 19580728 198701 1 001

PENGANTAR

Puji syukur dipanjatkan kehadirat Tuhan Yang Maha Esa atas segala berkat dan rahmatnya sehingga skripsi yang berjudul **"**Aplikasi *Dynamic Programming* Untuk Penjadwalan Pembangkit Tenaga Listrik**"** dapat diselesaikan.

Ucapan terima kasih disampaikan kepada semua pihak yang telah membantu hingga dapat diselesaikannya skripsi ini. Khususnya diucapkan kepada:

1. Bapak Dr. Ir. Sholeh Hadi Pramono, M.S., selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
2. Bapak M. Azis Muslim, S.T., M.T., Ph.D., selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
3. Bapak Moch. Rifan, S.T., M.T., selaku Ketua Program Studi Jurusan Teknik Elektro Fakultas Teknik, Universitas Brawijaya.
4. Bapak Hadi Suyono, S.T., M.T., Ph.D., selaku dosen pembimbing I dan Ibu Dr. Rini Nur Hasanah, S.T., M.Sc., selaku dosen pembimbing II.
5. Bapak dan Ibu Dosen serta Karyawan Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
6. Ibuku tersayang atas segala bentuk cinta kasihnya kepadaku yang terwujud dalam bentuk dukungan materiil, doa, dan semuanya.
7. Ayahku atas segala jerih payahnya di dalam membesarkan dan mendidikku.
8. Seluruh mahasiswa Jurusan Teknik Elektro khususnya dari Konsentrasi Teknik Energi Elektrik yang namanya tidak dapat disebutkan satu per satu.

Sebuah penantian panjang telah berakhir, menjadi sebuah titik perjalanan yang baru. Tiada yang sempurna di dunia ini, tersadar bahwa skripsi ini sangat jauh dari kesempurnaan. Karenanya, segala kritik dan saran yang sifatnya membangun dari pembaca tentang isi skripsi ini akan diterima dengan senang hati. Akhir kata, penulis berharap, semoga skripsi ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Malang, 27 Juli 2012

Penulis

DAFTAR ISI

PENGANTAR..... i
DAFTAR ISI..... ii
DAFTAR GAMBAR..... iv
DAFTAR TABEL..... v
DAFTAR LAMPIRAN..... vi
RINGKASAN..... vii
BAB I. Pendahuluan
1.1 Latar Belakang 1
1.2 Rumusan Masalah..... 2
1.3 Batasan Masalah..... 2
1.4 Tujuan 3
1.5 Sistematika Penulisan..... 3
BAB II. Tinjauan Pustaka
2.1 *Unit Commitment* dan *Economic Dispatch*..... 4
2.2 Konfigurasi Sistem Pembangkit pada Jaringan..... 5
2.3 Kendala-kendala dalam *Unit Commitment*..... 6
2.3.1 Gambaran umum mengenai kendala..... 6
2.3.2 Kesetimbangan daya..... 7
2.3.3 Cadangan berputar (*spinning reserve*)..... 7
2.3.4 Kendala biaya *start up*..... 8
2.4 Kurva Laju Panas (*Heat Rate*) dan Kurva Laju Biaya (*Cost Rate*)..... 10
2.5 Pemrograman Dinamis (*Dynamic Programming*/DP)..... 11
BAB III. Metodologi
3.1 Studi Literatur..... 14
3.2 Pengumpulan Data..... 14
3.3 Penyusunan Program Aplikasi..... 14
3.4 Analisa Data..... 18
3.5 Kesimpulan..... 18
BAB IV. Hasil Penelitian Dan Pembahasan
4.1 Pengolahan Data..... 19
4.2 Rancangan Program *Software* Aplikasi..... 21

4.2.1 Diagram alir program aplikasi..... 21
4.2.2 Fungsi-fungsi yang dirancang untuk program aplikasi..... 23
4.2.3 Validasi program..... 26
4.3 Hasil Pengujian dengan Program Aplikasi..... 26
4.3.1 Hasil pengujian dengan metode *dynamic programming*..... 27
BAB V. Kesimpulan Dan Saran
5.1 Kesimpulan..... 30
5.2 Saran..... 30
DAFTAR PUSTAKA..... 31
LAMPIRAN..... 32

DAFTAR GAMBAR

Gambar 2.1	Gambaran permasalahan *economic dispatch* dan *unit commitment*……..	5
Gambar 2.2	Sistem unit pembangkit dalam jaringan dengan tanpa memperhitungkan rugi-rugi (*losses*) jaringan…….....	6
Gambar 2.3	Biaya *start* dingin/*cooling* dan *start* panas/*banking*…….....	10
Gambar 2.4	Kurva *input-output* generator uap……	10
Gambar 2.5	Kurva *heat rate* dan *cost rate*……..	11
Gambar 3.1	Diagram alir metode *dynamic programming*……..	17
Gambar 4.1	Diagram alir *unit commitment* dengan metode *dynamic programming*……	23

DAFTAR TABEL

1.	Tabel 4.1	Nilai yang dibutuhkan dalam persamaan regresi dengan model polinomial…….	20
2.	Tabel 4.2	Perbandingan hasil program aplikasi DP dengan standar IEEE……	26
3.	Tabel 4.3	Jumlah state feasible setiap jam……..	27
4.	Tabel 4.4	Biaya minimum setiap jam…….	28
5.	Tabel 4.5	Hasil pengujian menggunakan metode DP (*dynamic programming*)…….	28

DAFTAR LAMPIRAN

Lampiran	Judul	Halaman
Lampiran 1:	Data unit pembangkit dengan daya maksimum (MW) dan daya minimum (MW)	32
Lampiran 2:	Biaya *start up* untuk masing-masing unit pembangkit dan biaya pelumas	33
Lampiran 3:	Konstanta persamaan biaya untuk masing-masing unit pembangkit	34
Lampiran 4:	*Average Full Load Cost*/AFLC (Rp/MWh), dan probabilitas *forced outage rate* (FOR)	35
Lampiran 5:	Contoh data beban harian di PLN-Jaya wilayah operasi area I (MW)	36
Lampiran 6:	Penjadwalan 26 Unit Pembangkit Termal selama 24 jam	37
Lampiran 7:	Pembebanan 26 Unit Pembangkit Termal selama 24 jam (dalam pu)	38
Lampiran 8:	*Listing* Program Aplikasi dengan Metode *Dynamic Programming*	39

RINGKASAN

Gigih Sanjaya S D, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Agustus 2012, Aplikasi Dynamic Programming untuk Penjadwalan Unit Pembangkit Tenaga Listrik. Dosen Pembimbing:
Hadi Suyono, S.T., M.T., Ph.D. dan Dr. Rini Nur Hasanah, S.T., M.Sc.

Energi listrik menjadi salah satu prioritas kebutuhan yang harus terpenuhi dalam kehidupan masyarakat. Karena peranan energi listrik yang begitu besar, penyedia energi listrik harus mampu memenuhi kebutuhan tersebut dengan faktor ketersediaan, kesinambungan, kuantitas, dan kualitas yang tinggi, serta harga yang wajar. Permasalahan muncul karena energi listrik tidak dapat disimpan dalam jumlah dan waktu banyak sehingga harus tersedia pada saat konsumen membutuhkan. Hal ini menyebabkan beban sistem berubah-ubah dari waktu ke waktu.

Pada pengoperasian sistem tenaga listrik, komponen biaya operasi terbesar adalah biaya bahan bakar. Unit-unit pembangkit termal yang menggunakan bahan bakar seperti, batu bara, gas alam dan minyak, memiliki karakteristik, biaya produksi, dan kapasitas daya yang berbeda-beda. Biaya total pembangkitan dapat diturunkan dengan mengoptimalkan energi yang diproduksi sesuai dengan kebutuhan konsumen. Optimasi yang efektif untuk kondisi beban seperti ini adalah optimisasi jangka pendek. Permasalahan yang terkait dengan ini adalah *unit commitment* (UC) dan *economic dispatch*.

Satu cara untuk menyelesaikan permasalahan *unit commitment* dan *economic dispatch* adalah dengan *dynamic programming* (DP). *Dynamic programming* merupakan sebuah teknik untuk memecahkan masalah dengan cara membagi masalah menjadi beberapa sub masalah yang saling tumpang-tindih (*overlapping*). Metode ini menunjukkan pemecahan dari tiap sub-sub masalah yang lebih kecil hanya dilakukan satu kali kemudian hasilnya disimpan untuk mendapatkan penyelesaian akhir dari permasalahan yang sebenarnya.

Prosedur penjadwalan dan optimasi dengan metode *dynamic programming* (DP) didapatkan dengan menguji sejumlah keputusan yang mungkin dalam banyak tahap. Sasaran utamanya adalah membuat suatu keputusan pada setiap sub masalah dengan meminimalkan biaya untuk semua keputusan yang dibuat. Dari hasil aplikasi dengan metode DP didapatkan nilai optimal biaya pembangkitan unit pembankit yang *feasible* setiap jamnya selama 24 jam. Kemudian, nilai optimal biaya pembangkitan untuk tiap jam dijumlahkan, sehingga didapatkan nilai optimal dari biaya total bahan bakar minimum sebesar Rp 6.091.513.380,58 dan besar pembebanan dari unit pembangkit yang *feasible* serta penjadwalannya ditunjukkan pada **Lampiran**.

Kata Kunci: *Dynamic programming*, *unit commitment*, penjadwalan pembangkit

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini, energi listrik menjadi salah satu kebutuhan utama yang harus tersedia dalam kehidupan masyarakat. Karena peranannya yang begitu besar, penyedia energi listrik harus mampu memenuhi faktor ketersediaan, kesinambungan, kuantitas, dan kualitas yang tinggi dengan harga yang wajar. Permasalahan muncul karena energi listrik tidak dapat disimpan dalam jumlah banyak dan waktu yang lama sehingga harus tersedia pada saat konsumen membutuhkan. Hal ini menyebabkan menjadi berubah dari waktu ke waktu. Apabila daya yang dikirim dari bus-bus pembangkit lebih besar dari kebutuhan daya pada bus-bus beban, maka akan timbul pemborosan daya. Ketika daya yang dibangkitkan lebih rendah dari kebutuhan, maka harus ada pemadaman lokal pada bus-bus beban, yang akan mengakibatkan kerugian pada konsumen. Untuk itu, perlu adanya perencanaan dalam pengoperasian dan pengendalian sistem tenaga listrik.

Pada pengoperasian sistem tenaga listrik, komponen biaya operasi terbesar adalah biaya bahan bakar. Unit-unit pembangkit termal yang menggunakan bahan bakar seperti, batu bara, gas alam dan minyak, memiliki karakteristik, biaya produksi, dan kapasitas daya yang berbeda-beda. Biaya total pembangkitan dapat diturunkan dengan mengoptimalkan energi yang diproduksi sesuai dengan kebutuhan konsumen. Optimasi yang efektif untuk kondisi beban seperti ini adalah optimasi jangka pendek. Permasalahan lain yang tidak terpisahkan, yaitu *unit commitment* (UC) dan *economic dispatch*. *Unit commitment* merupakan permasalahan untuk menentukan penjadwalan unit pembangkit yang harus dioperasikan setelah diketahui ramalan profil beban dan unit-unit pembangkit yang siap dioperasikan, sedangkan *economic dispatch* adalah permasalahan untuk menentukan pembebanan unit pembangkit setelah diketahui beban yang harus ditanggung unit-unit pembangkit tersebut agar biaya dapat diturunkan serendah-rendahnya. (Zhu, 2007: 214)

Salah satu cara untuk menyelesaikan permasalahan *unit commitment* dan *economic dispatch* adalah dengan metode *dynamic programming* (DP). Menurut Levitin (2011: 20), *dynamic programming* merupakan sebuah teknik untuk memecahkan masalah dengan cara membagi masalah menjadi beberapa sub masalah yang saling tumpang-tindih (*overlapping*). Metode ini menunjukkan pemecahan dari tiap sub-sub

masalah yang lebih kecil hanya dilakukan satu kali kemudian hasilnya disimpan untuk mendapatkan penyelesaian akhir dari permasalahan yang sebenarnya.

Terkait dengan permasalahan *unit commitment* dan *economic dispatch* terdapat sejumlah biaya yang mempengaruhi proses pengambilan keputusan dan biaya-biaya ini dipengaruhi oleh tahap perhitungan sebelumnya dalam metode *dynamic programming*. Biaya yang harus dikeluarkan untuk berpindah dari sebuah keputusan (tahap) ke keputusan (tahap) berikutnya disebut sebagai biaya transisi. Sasaran utamanya adalah membuat suatu keputusan pada setiap tahap permasalahan dengan meminimalkan biaya untuk semua keputusan yang dibuat dengan memperhatikan kendala-kendala yang ada.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dijelaskan sebelumnya, permasalahan dalam skripsi ini dapat dirumuskan sebagai berikut:

- Bagaimana menjadwalkan unit-unit pembangkit untuk memenuhi kebutuhan beban sistem dengan biaya minimum dan memenuhi batasan-batasan teknis pengoperasian sistem pembangkit menggunakan metode optimasi *dynamic programming*.

1.3 Batasan Masalah

Adapun batasan-batasan masalah dalam penyusunan skripsi ini adalah sebagai berikut:

- Penjadwalan unit pembangkit hanya dilakukan untuk unit pembangkit termal.
  Dalam skripsi ini penjadwalan dibatasi hanya pada pembangkit termal, dengan mempertimbangkan biaya produksi energi listrik yang jauh lebih besar daripada unit pembangkit jenis yang lain dari sisi biaya bahan bakar.
- Menggunakan metode *dynamic programming* (DP).
  Metode DP adalah metode sistematik yang menguji sejumlah keputusan yang mungkin dengan banyak tahap.
- Tidak membahas kendala ketersediaan bahan bakar.
  Bahan bakar diasumsikan selalu tersedia untuk pembangkitan.
- Tidak membahas kendala ketersediaan petugas operator unit pembangkit.
  Petugas operator unit pembangkit diasumsikan selalu tersedia dan siap untuk mengoperasikan pembangkit untuk menghasilkan energi listrik yang dibutuhkan.

- Tidak membahas polusi yang ditimbulkan unit pembangkit termal.
  Dalam skripsi ini tidak diperhitungkan polusi yang dihasilkan oleh unit pembangkit termal selama proses pembangkitan.
- Tidak membahas kendala transmisi daya listrik.
  Jaringan transmisi diasumsikan dapat menyalurkan semua daya listrik yang dihasilkan oleh unit pembangkit.

1.4 Tujuan

Tujuan dari penulisan skripsi ini adalah mengaplikasikan *dynamic programming* sebagai metode optimasi dalam penyelesaian masalah penjadwalan pembangkit tenaga listrik. Untuk menghasilkan penjadwalan unit-unit pembangkit dalam memenuhi beban sistem dengan biaya minimum dan tetap memenuhi batasan-batasan teknis pengoperasian sistem pembangkit.

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan skripsi ini adalah sebagai berikut:

Bab I. Pendahuluan, membahas tentang latar belakang, rumusan masalah, ruang lingkup, tujuan, batasan masalah, dan sistematika penulisan.

Bab II. Tinjauan Pustaka, mengenai *unit commitment*, *economic dispatch*, karakteristik unit pembangkit, dan *dynamic programming*.

Bab III. Metodologi, berisi tentang metode pengambilan data dan metode analisis data.

Bab IV. Analisa Data dan Pembahasan, perhitungan biaya total pembangkitan setelah dilakukan optimisasi dengan *dynamic programming*.

Bab V. Penutup, berisi tentang kesimpulan dan saran.

BAB II
TINJAUAN PUSTAKA

2.1 *Unit Commitment* dan *Economic Dispatch*

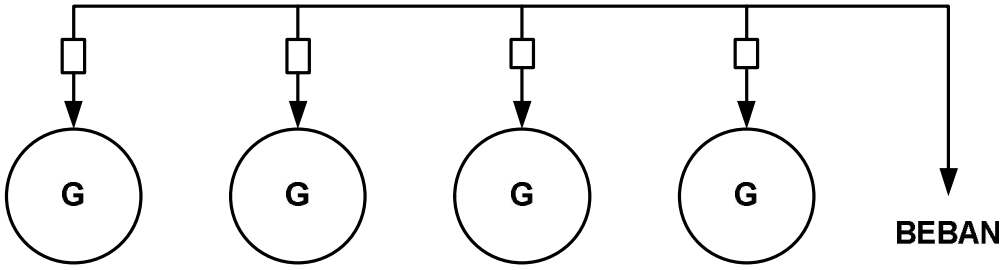
Aktivitas manusia yang terus berputar mengakibatkan sistem pelayanan yang harus disediakan untuk penduduk yang luas akan mengalami perputaran juga. Termasuk di dalamnya sistem transportasi, sistem komunikasi, demikian juga pada sistem tenaga listrik. Karakteristik beban dari waktu ke waktu berubah dengan jangkauan yang luas. Misalnya, beban puncak siang hari biasanya dua kali lipat beban minimum di malam hari. Beban puncak tahunan sekitar tiga kali beban minimum tahunannya karena variasi musiman. (Gonen, 1986: 39)

Penyaluran tenaga listrik kepada pelanggan tidak lepas dari permasalahan penjadwalan pembangkit yang beroperasi (*online*) untuk menyuplai beban dan rugi-rugi pada saluran serta biaya total pembangkitan. Menurut Stanton, dkk. (2000: 1555), karena semua unit pembangkit yang *online* memiliki biaya pembangkitan yang berbeda-beda, perlu untuk menemukan tingkat pembangkitan masing-masing unit yang akan memenuhi beban dengan biaya minimum. Ini harus memperhitungkan menjelaskan fakta bahwa biaya pembangkitan dalam satu generator adalah tidak sebanding dengan tingkat pembangkitannya, tetapi memiliki hubungan fungsi nonlinier. Selain itu, karena sistem ini secara geografis tersebar, kerugian transmisi tergantung pada pola pembangkitan dan harus dipertimbangkan dalam memperoleh pola optimal.

Membawa unit pembangkit pada kondisi *commit* artinya membuat suatu unit pembangkit bekerja dan dalam kondisi sinkron dengan sistem dan menghubungkan daya ke jaringan. Membuat bekerja sejumlah unit pembangkit yang mencukupi dan membiarkannya tetap bekerja menyangkut perhitungan ekonomis. Sangat mahal biaya untuk menjalankan terlalu banyak unit pembangkit jika unit-unit pembangkit tersebut tidak dibutuhkan seluruhnya. Dana yang dapat dihemat akan besar jika hanya unit pembangkit yang dibutuhkan untuk menyuplai beban pada saat tertentu saja yang dijalankan. (Wood, 1984: 131).

Permasalahan yang tidak kalah pentingnya adalah *economic dispatch*. *Economic dispatch* erat hubungannya dengan permasalahan *unit commitment*. *Economic dispatch* adalah cara untuk menentukan besarnya daya yang harus dibangkitkan oleh tiap unit pembangkit ketika sudah didapatkan besarnya beban yang harus disuplai dan unit-unit pembangkit yang akan beroperasi agar biaya pembangkitan dapat ditekan

serendah-rendahnya. Sementara itu, *unit commitment* (UC) merupakan cara untuk menentukan unit-unit pembangkit yang harus beroperasi. Gambar 2.1 akan membantu menjelaskan tentang permasalahan *economic dispatch* dan *unit commitment*.



Gambar 2.1 Gambaran Permasalahan *Economic Dispatch* dan *Unit Commitment*
Sumber: Wood (1984: 57)

Tujuan dari UC adalah penjadwalan unit pembangkit termal yang dioperasikan untuk memenuhi beban sistem, dengan sejumlah keputusan menghidupkan atau mematikan unit-unit pembangkit sesuai keperluan dengan memperhatikan beban sistem dan cadangan berputar yang diperlukan tanpa melebihi kemampuan dari kapasistas pembangkitan yang ada.

2.2 Konfigurasi Sistem Pembangkit pada Jaringan

Pada Gambar 2.2 diperlihatkan sejumlah N sistem pembangkit yang dihubungkan dengan bus bar untuk melayani beban P_R. Masukan pada masing-masing unit, dilambangkan sebagai F_i, yang menyatakan laju biaya pada masing-masing unit pembangkit. Keluaran pada masing-masing unit adalah P_i, yang menyatakan jumlah energi listrik yang dihasilkan pada bagian-bagian unit tersebut. Sedangkan laju biaya totalnya merupakan jumlah keseluruhan dari masing-masing biaya per-unit. Kendala utama (*main constraint*) pada sistem operasinya adalah jumlah *output* energi harus sama dengan kebutuhan beban.

Konfigurasi pada Gambar 2.2, secara matematis dapat dijelaskan dengan suatu fungsi objektif (F_T) yang menunjukkan biaya total dalam memenuhi kebutuhan beban. Permasalahannya adalah untuk meminimalkan fungsi F_T, tergantung pada batas/syarat bahwa jumlah energi yang dihasilkan harus sama dengan jumlah bebannya. Dengan tidak memasukkan rugi-rugi transmisi secara eksplisit, maka persamaan matematisnya adalah:

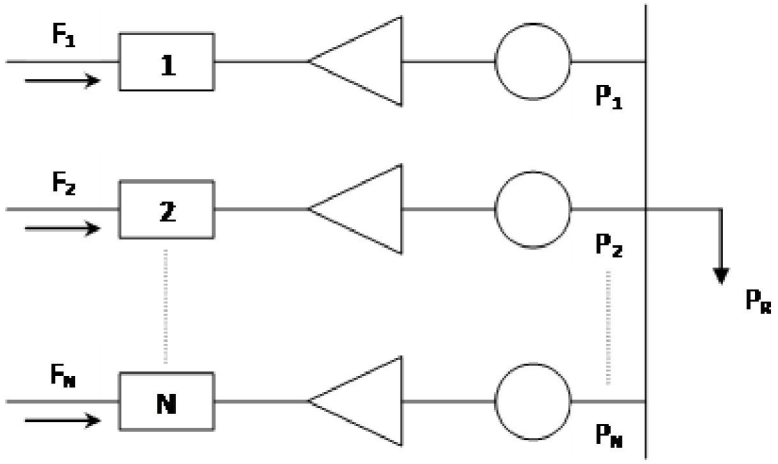
$$F_T = F_1 + F_2 + F_3 + \cdots + F_N$$
$$= \sum_{i=1}^{N} F_i(P_i) \qquad (2\text{-}1)$$

dengan:

P_i = daya nyata *output* dari generator ke-i (MW)

$F_i(P_i)$ = biaya pengoperasian dari unit ke-i (Rp/jam)

N = total jumlah dari units pembangkit



Gambar 2.2 Sistem unit pembangkit dalam jaringan
dengan tanpa memperhitungkan rugi-rugi (*losses*) jaringan
Sumber: Wood (1984: 30)

2.3 Kendala-Kendala dalam *Unit Commitment*

2.3.1 Gambaran umum mengenai kendala

Dalam proses optimasi pada umumnya, dan khususnya pada optimasi operasi sistem tenaga listrik, selalu ada kendala-kendala (*constraints*). Secara matematis untuk sistem yang terdiri dari N variable dapat dijelaskan sebagai berikut: (Sivanagaraju, 2009: 35)

- Fungsi (x_1, x_2, x_3,…, x_n) yang disebut dengan fungsi objektif (*objective function*) adalah fungsi yang akan dioptimalkan, misalnya akan dicari nilai maksimum atau nilai minimumnya. Dalam hal optimasi operasi sistem tenaga listrik F(x_1, x_2, x_3,…, x_n) adalah fungsi biaya operasi (contohnya, bahan bakar) yang perlu dicari nilai minimumnya. Pada kasus ini x_1, x_2, x_3,…, x_n, adalah daya yang dibangkitkan oleh unit pembangkit ke-1, ke-2, ke-3 sampai ke-n dalam sistem.
- Kendala-kendala yang harus diatasi, secara matematis digambarkan oleh pertidaksamaan atau persamaan yang harus dipenuhi, misalnya pada operasi sistem tenaga tenaga listrik digambarkan oleh pertidaksamaan dan persamaan sebagai berikut:

$$K_1 \leq F(x_1, x_2, x_3, \ldots\ldots\ldots, x_n) \leq K_2 \qquad (2\text{-}2)$$

$$F(x_1, x_2, x_3, \ldots\ldots\ldots, x_n) \leq K_3 \quad (2\text{-}3)$$

$$F(x_1, x_2, x_3, \ldots\ldots\ldots, x_n) - B - L = 0 \quad (2\text{-}4)$$

dengan,

K_1 = batas pembangkitan daya minimum

K_2 = batas pembangkitan daya maksimum

K_3 = batas pembangkitan daya maksimum untuk sekelompok unit pembangkit tertentu ($x_1, x_2, x_3, \ldots, x_n$), misalnya karena pembatasan aliran daya.

B = daya beban (yang dibutuhkan konsumen)

L = rugi-rugi daya dalam sistem.

2.3.2 Kesetimbangan daya

Kendala utama *unit commitment* yang harus dipecahkan adalah kesetimbangan daya, yaitu total energi listrik yang dibangkitkan oleh unit-unit pembangkit yang beroperasi untuk mencukupi kebutuhan beban sistem. Persamaan matematis kesetimbangan daya ditulis sebagai berikut: (Wood, 1984: 132)

$$\sum_{i=1}^{N} P_i(H) = P_L(H) \quad (2\text{-}5)$$

dengan:

H = Jumlah waktu studi (1,2,3…….M)

$P_i(H)$ = Daya keluaran unit i pada jam ke-H (MW)

$P_L(H)$ = Beban sistem pada jam ke-H (MW)

N = Jumlah unit pembangkit yang beroperasi (*online*)

Besarnya daya pembangkitan yang harus dikeluarkan oleh masing-masing unit pembangkit dapat dilakukan dengan pembebanan ekonomis, yang dalam penelitian ini didasarkan pada daftar prioritas dengan memperhatikan *average full load cost* (AFLC) pada setiap pembangkit.

2.3.3 Cadangan berputar (*spinning reserve*)

Cadangan berputar dalam UC merupakan jumlah total daya pembangkitan dari semua unit yang beroperasi pada sistem dikurangi beban dan rugi-rugi dari sistem. Artinya, cadangan berputar merupakan banyaknya daya yang tersisa untuk menyediakan daya apabila ada unit pembangkit yang tiba-tiba keluar dari sistem dengan alasan teknis tertentu. Secara sederhana dapat dijelaskan bahwa apabila ada unit pembangkit yang keluar dari sistem, unit pembangkit yang lain harus mampu menyediakan cukup daya sehingga keluarnya unit pembangkit tersebut tidak menyebabkan penurunan frekuensi

dan tegangan yang terlalu besar dan dapat menyediakan cukup daya untuk suatu periode tertentu. Secara matematis cadangan berputar dapat dinyatakan dalam persamaan berikut: (Wood, 1984: 135)

$$\sum_{i=1}^{N} SR_i(H) \geq SR_{min}(H) \qquad (2\text{-}6)$$

dengan:

H = waktu studi (1,2,3…….M)

SR_i (H) = daya cadangan berputar pada jam ke-H (MW)

SR_{min} (H) = cadangan berputar minimal yang harus diberikan pada jam ke-H

N = jumlah unit pembangkit yang beroperasi.

Ada bebarapa aturan untuk mengalokasikan cadangan berputar, salah satu yang umum digunakan adalah mengalokasikan cadangan berputar sebesar daya pembangkitan maksimal diantara unit-unit yang beroperasi pada waktu tersebut. Sehingga apabila unit dengan daya pembangkitan paling besar terpaksa keluar (*forced outage*), sistem masih mampu melayani beban yang ada.

Aturan lain yang dapat digunakan untuk mengalokasikan cadangan berputar adalah berdasarkan beban yang sedang ditanggung. Dalam penelitian ini cadangan berputar sebesar 10% dari beban puncak.

2.3.4 Kendala biaya *start up*

Biaya *start up* adalah biaya yang diperlukan oleh unit pembangkit untuk beroperasi dari keadaan tidak beroperasi atau tidak terhubung pada sistem. Total dari biaya *start-up* sistem dapat dinyatakan dengan persamaan berikut ini: (Dasgupta, 1993: 5)

$$StartCost(t) = \sum_{i=1}^{N} u_i^t (1 - u_i^{t-1}) S_i^t \qquad (2\text{-}7)$$

dengan:

StartCost (t) = biaya *start up* pada waktu t

u_i^t = Keadaan unit pembangkit ke-i untuk waktu sekarang (0 dan 1)

u_i^{t-1} = Keadaan unit pembangkit ke-i untuk waktu sebelumnya (0 dan 1)

S_i^t = Biaya *start up* yang ditetapkan untuk unit ke-i

N = Jumlah unit pembangkit yang *online*

Pada tiap unit pembangkit termal ada dua macam biaya *start up*, yaitu:

- *Start* dingin/*cooling*, yaitu kondisi saat pembangkit dilepas dari sistem, temperatur boiler dibiarkan turun dari temperatur kerjanya, sehingga saat akan dioperasikan dilakukan pemanasan lagi.
- *Start* panas/*banking*, yaitu saat pembangkit dilepas dari sistem, temperatur boiler dipertahankan pada suhu kerjanya, sehingga memerlukan energi untuk mempertahankan temperatur tersebut.

Kedua macam *start* tersebut memerlukan biaya yang berbeda, dan memiliki kelebihan dan kekurangan masing-masing. Secara matematis, kedua *start* tersebut dinyatakan pada persamaan berikut: (Wood, 1984: 137)

Biaya *start* dingin = $C_c\left(1 - e^{-t/\alpha}\right)xF + C_f$ (2-8)

dengan,

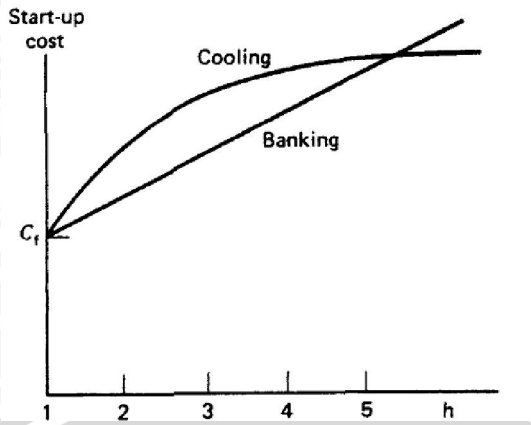
C_c = Koefisien biaya *start* dingin (Mbtu)
F = Biaya bahan bakar (Rp/Mbtu)
α = Konstanta termal unit pembangkit
C_f = Fixed cost (biaya tetap unit pembangkit seperti biaya perawatan unit) (Rp)
t = Lama waktu unit didinginkan (jam)

Biaya *start* panas = $C_t \; x \; t \; x \; F \; + \; C_f$ (2-9)

dengan,

C_t = Biaya untuk menjaga temperatur boiler (Mbtu/h)
F = Biaya bahan bakar (Rp/Mbtu)
α = Konstanta termal unit pembangkit
C_f = Fixed cost (biaya tetap unit pembangkit seperti biaya perawatan unit) (Rp)
t = Lama waktu unit didinginkan (jam)

Gambar 2.3 akan membantu menggambarkan kelebihan dan kekurangan kedua tipe *start* tersebut. Dapat dilihat pada Gambar 2.3 jika start dingin/*cooling* akan menguntungkan apabila jeda waktu ketika unit pembangkit (*decommit*) dilepaskan dari sistem dan ketika akan dihubungkan kembali (*recommit*) dengan sistem relatif lama. Dan jika jeda waktu antara *decommit* dan *recommit* dari unit tersebut cukup dekat, maka akan lebih menguntungkan jika digunakan *start* panas.



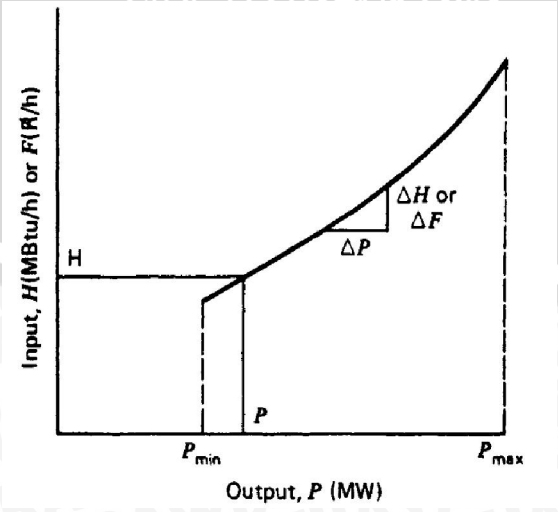
Gambar 2.3 Biaya *Start* Dingin/*Cooling* dan *Start* Panas/*Banking*

Sumber: Wood (1984: 138)

2.4 Kurva Laju Panas (*Heat Rate*) Dan Kurva Laju Biaya (*Cost Rate*)

Salah satu karakteristik penting pada unit pembangkit termal adalah karakteristik *input-output*. Karakteristik ini direpresentasikan dalam kurva *input-output*, yaitu kurva yang menghubungkan antara daya panas yang diperlukan sebagai *input* dan daya listrik yang dihasilkan sebagai *output*. *Input* dapat juga berupa total biaya yang diperlukan per jam. Gambar 2.4 menunjukkan karakteristik *input-output* ideal.

Berdasarkan kurva karakteristik *input-output* tersebut, bisa didapatkan karakteristik lain yang tidak kalah penting, yaitu karakteristik laju panas (*heat rate*). Karakteristik laju panas direpresentasikan dengan $(\Delta H/\Delta P)$ sebagai *input* dan P sebagai *output*. $(\Delta H/\Delta P)$ merupakan turunan pertama dari kurva *input-output* yang dinyatakan dalam Mbtu/kWh atau Rp/kWh, yaitu pemakaian energi panaas per kWh energi listrik yang dihasilkan atau jumlah biaya yang diperlukan untuk menghasilkan per kWh listrik.



Gambar 2.4 Kurva *Input-Output* Generator Uap

Sumber: Wood (1984: 9)

Gambar 2.5 menunjukkan kurva laju panas dan laju biaya. Kurva *input-output* ataupun kurva laju panas bisa diperoleh dengan perhitungan dari data rancangan unit pembangkit atau dengan pengujian laju panas. Pengujian laju panas rata-rata untuk beban 50%, 75%, dan 100% beban penuh dilakukan untuk membentuk persamaan konsumsi bahan bakar *HR*, dari pengujian tersebut didapatkan konstanta-konstanta *a*, *b*, dan *c*.

Berdasarkan kurva tersebut dapat direperesentasikan dalam bentuk polinomial kuadratik, yaitu: (Wood, 1984: 10)

$$HR(P_i) = c + bP_i + aP_i^2 \qquad (2\text{-}10)$$

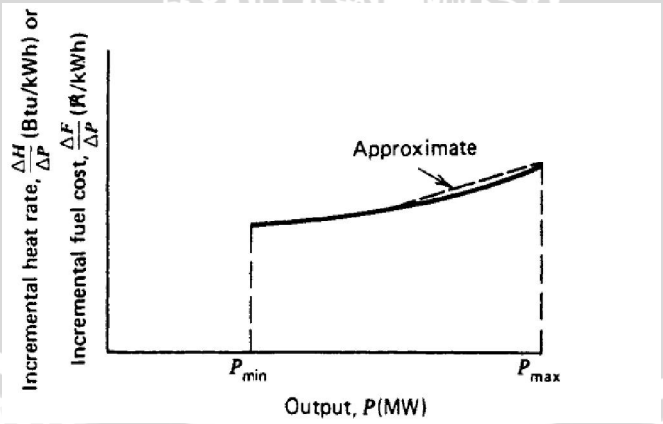
Dengan *a*, *b*, *c* merupakan koefisien konstanta dan P_i adalah output MW (per unit) dengan batasan sebagai berikut:

$$P_{i,min} \leq P_i \leq P_{i,max} \qquad (2\text{-}11)$$

dengan:

$P_{i,min}$, $P_{i,max}$ = daya minimal dan daya maksimal dari unit ke-i (MW) dan a, b, c, adalah koefisien konstanta.

Karakteristik *heat rate* digunakan untuk menggambarkan efisiensi dari unit pembangkit. Pada pembangkitan ekonomis, salah satu penggunaannya adalah pada biaya bahan bakar yang diperlukan untuk menghasilkan daya yang dibutuhkan. Dengan mengetahui jenis bahan bakar yang digunakan bersama dengan nilai kalor dan biayanya, maka kurva *heat rate* dapat ditransformasikan sebagai laju pembiayaan (*cost rate*).



Gambar 2.5 Kurva *Heat Rate* dan *Cost Rate*

Sumber: Wood (1984: 10)

2.5 Pemrograman Dinamis (*Dynamic Programming/DP*)

Pemrograman dinamis (*dynamic programming*) memiliki banyak keuntungan dibandingkan dengan skema pencacahan, keuntungan utama ini dapat mengurangi

besarnya suatu permasalahan. Misalnya, pada suatu sistem terdapat unit-unit dengan setiap kombinasi dari unit-unit tersebut mampu menyuplai beban. Sehingga kombinasi maksimal pengujian sebanyak 2^4 - 1 = 15. Namun, jika suatu urutan prioritas yang tepat dikenakan, hanya ada empat kombinasi:

Prioritas 1 unit + Prioritas 2 unit

Prioritas 1 unit + Prioritas 2 unit + Prioritas 3 unit

Prioritas 1 unit + Prioritas 2 unit + Prioritas 3 unit + Prioritas 4 unit

Pengenaan daftar prioritas yang diatur dalam urutan tingkat biaya rata-rata beban penuh akan menghasilkan penyaluran yang benar secara teoritis dan keikatan hanya jika:

a. Biaya tanpa beban adalah nol.
b. Karakteristik unit *input-output* linier antara keluaran nol dan beban penuh.
c. Tidak ada batasan lainnya.
d. Biaya *start up* nilainya ditetapkan.

Dalam pendekatan secara pemrograman dinamis berikut ini, diasumsikan bahwa:

a. Suatu kondisi terdiri atas berbagai unit dengan unit tertentu beroperasi dan sisanya *off line.*
b. Biaya *start up* dari suatu unit tidak tergantung dari waktu selama unit tersebut *off line*.
c. Tidak ada biaya untuk men-*shut down* suatu unit.
d. Ada urutan prioritas yang ketat, dan dalam setiap interval, jumlah minimal tertentu dari suatu kapasitas harus beroperasi.

Suatu kondisi yang layak adalah suatu kondisi pada saat unit yang beroperasi dapat menyediakan beban yang diperlukan dan yang memenuhi jumlah minimal suatu kapasitas setiap periode.

Perlu diketahui bahwa dalam metode ini setiap kombinasi dari unit-unit pembangkit tidak diharuskan untuk menghasilkan suatu penghematan selama proses komputasi. Proses ini akan diulang sampai semua unit yang tersedia telah habis. Keuntungan dari pendekatan ini adalah bahwa setelah memperoleh nilai optimal dari k unit, akan sangat mudah untuk menentukan secara optimal pembebanan (k +1) unit.

Persamaan metode pemrograman dinamis (DP) memberikan hubungan rekursif berikut ini: (Nagrath, 2008: 346)

$$F_n(x) = \min_y\{f_N(y) + F_{N-1}(x-y)\} \qquad (2\text{-}12)$$

Dengan fungsi biaya $F_n(x)$ didefinisikan sebagai berikut,

$F_N(x)$ = biaya minimum untuk membangkitkan x MW dengan N unit pembangkit,

$f_N(y)$ = biaya pembangkitan y MW oleh unit ke-N

$F_{N-1}(x-y)$ = biaya minimum untuk membangkitkan (x-y) MW dari sisa (N-1) unit

BAB III
METODOLOGI

Metode penelitian dalam skripsi ini menggunakan metode simulasi dengan pemrograman dinamis (DP), yang digunakan untuk menganalisis data karakteristik unit pembangkit dan data beban dalam proses iterasi algoritma pemrograman dinamis pada program aplikasi yang akan dirancang untuk dioptimasi dan dihitung biaya total pembangkitannya. Adapun langkah-langkah metode ini adalah sebagai berikut:

3.1. Studi Literatur

Yaitu mempelajari dari berbagai sumber buku referensi yang ada dan berkaitan dengan tujuan penulisan skripsi:

a. Mempelajari tentang *unit commitment* dan *economic dispatch*
b. Mempelajari tentang unit pembangkit tenaga listrik
c. Mempelajari tentang pemrograman dinamis (*dynamic programming*)

3.2. Pengumpulan Data

Data dalam skripsi ini menggunakan data sekunder dari unit-unit pembangkit pada PT. PLN Pembangkitan Jawa-Bali I (PJB I) Area Operasi I. Data-data yang diuji-cobakan adalah data-data peramalan beban dan karakteristik pembangkit selama 24 jam.

Data yang diperlukan dalam penelitian ini, antara lain:

a. Daya maksimum dan minimum dari unit pembangkit
b. *Heat rate* dari masing-masing unit pembangkit
c. Jenis dan harga bahan bakar yang digunakan
d. Biaya konsumsi pelumas
e. Beban harian untuk wilayah operasi area 1

Data detail yang diperlukan dalam penelitian ini terdapat dalam **Lampiran A**.

3.3. Penyusunan Program Aplikasi

Setelah melakukan pengumpulan data, tahap berikutnya adalah menyusun program aplikasi untuk memecahkan masalah optimisasi *unit commitment* yang diterjemahkan ke dalam bahasa pemrograman C++, berdasarkan algoritma *dynamic programming*. Langkah-langkah yang akan dilakukan adalah sebagai berikut:

a. Menentukan model persamaan biaya konsumsi bahan bakar

Persamaan biaya konsumsi bahan bakar setiap jam merupakan bentuk lain dari persamaan *heat rate* dari masing-masing pembangkit. Hal ini bisa dilakukan dengan mengolah data yang ada, yaitu *heat rate* rata-rata untuk beban 50%, 75% dan 100% beban penuh. Persamaan $HR(P)$, yang merupakan persamaan konsumsi bahan bakar yang dibuat dengan metode persamaan interpolasi model polinomial derajat dua. Setelah mendapatkan persamaan konsumsi bahan bakar untuk setiap jamnya $HR(P)$, maka akan didapatkan persamaan biaya total setiap jam,

$$F(P) = HR(P) * \text{Biaya\_Bakar/kg} \quad (3\text{-}1)$$

b. Menentukan biaya produksi rata-rata beban penuh

Setelah didapatkan persamaan biaya konsumsi bahan bakar untuk masing-masing unit kemudian dicari biaya produksi rata-rata beban penuh (*average full load production cost*) untuk menentukan daftar prioritas (*priority list*). Unit yang mempunyai biaya produksi per-kWh yang paling rendah akan diprioritaskan untuk *on line*, daripada unit-unit lain yang mempunyai biaya produksi per-kWh lebih besar.

c. Menyusun diagram alir

Metode *dynamic programming* (DP) merupakan suatu metode sistematik yang menguji sejumlah keputusan yang mungkin dengan banyak tahap. Terdapat sejumlah biaya yang terkait dengan keputusan yang mungkin dan biaya ini dipengaruhi oleh tahap sebelumnya. Biaya transisi adalah biaya yang harus dikeluarkan untuk berpindah dari sebuah keputusan ke keputusan berikutnya. Sasaran utamanya adalah membuat suatu keputusan pada setiap tahap permasalahan dengan meminimalkan biaya untuk semua keputusan yang dibuat dan tetap memperhatikan kendala-kendala yang ada.

Permasalahan UC dapat diselesaikan dengan metode DP, yang menggunakan langkah maju (*forward*) untuk menganalisis beban yang selalu berubah-ubah, melalui langkah-langkah sebagai berikut:

- Menentukan kombinasi dari unit-unit pembangkit yang akan diselidiki. Suatu kombinsi menunjukkan sederetan status pembangkit, dimana suatu unit on line ditandai dengan angka 1 dan untuk kondisi off line ditandai dengan angka 0. Banyaknya kombinasi yang mungkin dari sejumlah N pembangkit adalah sebanyak $(2^N - 1)$. Kombinasi ke-J pada jam ke-H dinyatakan sebagai *state* (H,J).

- Dapatkan kombinasi yang layak pada tahap (jam) tertentu. Dimana syarat bahwa suatu kombinasi layak adalah dengan mempertimbangkan kendala-kendala yang ada, yaitu bahwa :
- Kesetimbangan daya pembangkitan dengan beban sebagaimana pada rumusan (2-2), dimana daya beban berada diantara pembangkitan daya minimum dan makisimum unit on line. Kondisi tidak layak apabila tidak memenuhi persamaan di atas.
- Kendala cadangan berputar yaitu sebagaimana pada persamaan (2-5)
- Untuk *state* yang layak tersebut pada tiap jamnya lakukan pembebanan ekonomis untuk setiap unit pembangkit *on line*. Sehingga didapatkan biaya operasi pembangkitan untuk tiap-tiap *state* tersebut untuk melayani beban yang ada. Pada penelitian ini pembebanan ekonomis berdasarkan *average full load cost* (AFLC).
- Menentukan keputusan optimal secara rekursif pada tiap jamnya, berdasarkan pada persamaan (2-12), yang dijabarkan menjadi,

  $$F_{\cos t}(H,J) = \min_{\{K\}}[P_{\cos t}(H,J) + S_{\cos t}(H-1,K:H,J) + F_{\cos t}(H-1,K) \quad (3\text{-}2)$$

  dengan,

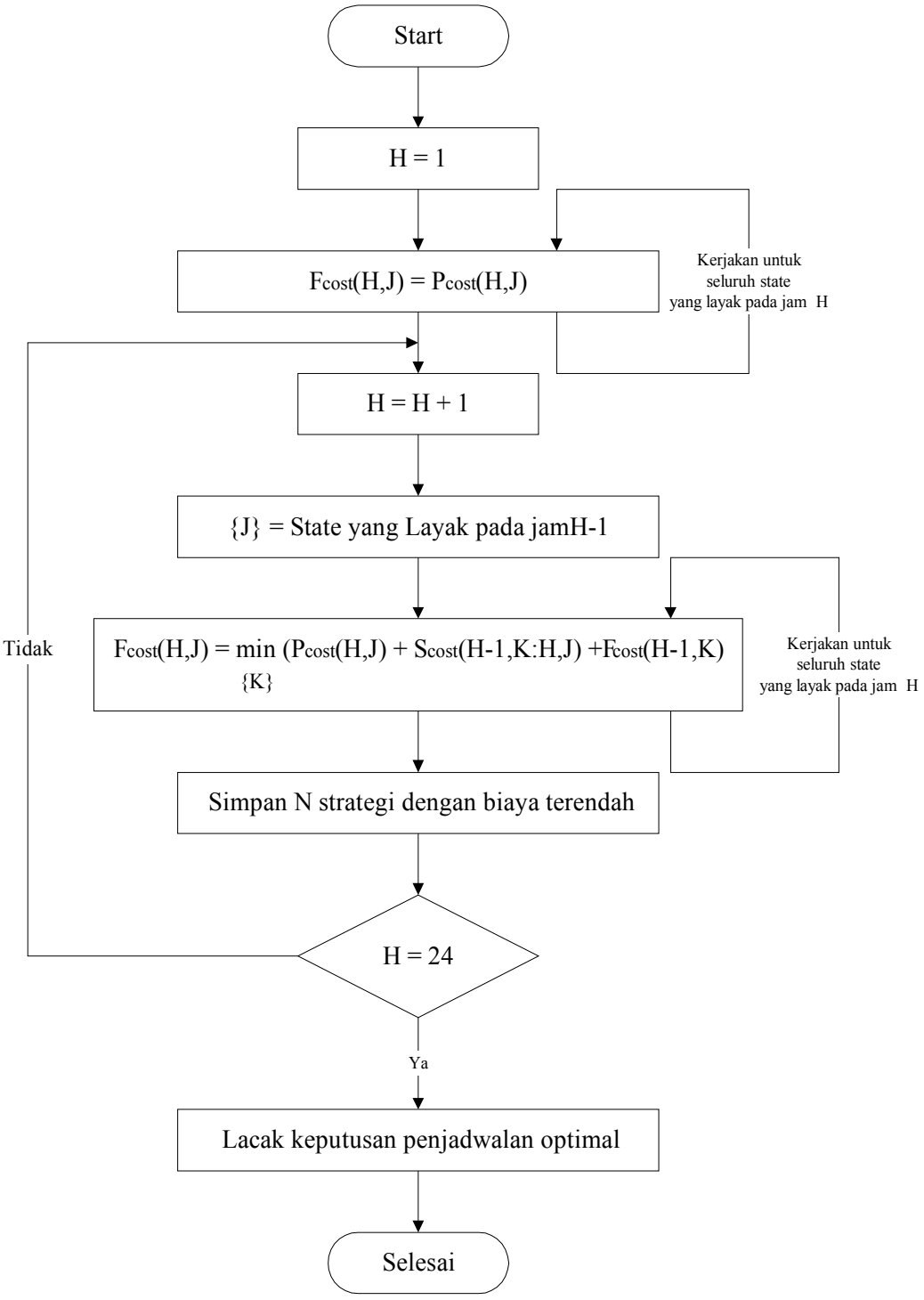
  $F_{cost}(H,J)$: Biaya total terendah untuk sampai state (H,J)

  $S_{cost}(H\text{-}1,L : H,J)$: Biaya transisi dari state $(H\text{-}1,K)$ ke state (H,J)

  $P_{cost}(H,J)$: Biaya produksi untuk state (H,J).

  {K} : Kombinasi yang layak pada jam H-1
- Setelah mendapatkan keputusan optimal sampai jam terakhir, status pembangkitan unit tiap jamnya dari awal sampai akhir kurun studi, dapat ditentukan dengan melakukan pelacakan jejak kembali melalui jalur transisi optimal.

Dari uraian di atas algoritma dari DP dapat digambarkan dalam diagram alir berikut ini:



Gambar 3.1 Diagram alir metode *dynamic programming*

d. Menyusun program

Alat yang dibutuhkan dalam penelitian ini adalah satu set PC dengan kemampuan cukup untuk mengoperasikan perangkat lunak Borland C++ dan Program Visual Borland C++ Builder. Spesifikasi PC minimum memiliki RAM 32MB dan processor P100 ke atas, untuk kelancaran pengolahan data.

3.4. Analisis Data

Setelah dilakukan optimasi dengan program aplikasi yang telah dirancang, keluaran program aplikasi tersebut merupakan hasil analisis tentang biaya total pembangkitan terbaik, yaitu biaya total pembangkitan yang paling kecil, waktu eksekusinya dan kelayakan solusinya berdasarkan batasan-batasan yang telah ditetapkan.

3.5. Kesimpulan

Dari analisis yang telah dilakukan akan diambil kesimpulan tentang kelayakan penerapan aplikasi algoritma *dynamic programming* untuk menyelesaikan permasalahan *unit commitment* pembangkit tenaga listrik.

BAB IV
ANALISIS DATA DAN PEMBAHASAN

Pada bagian ini akan dilakukan analisis data-data unit pembangkit dan karakteristiknya, kemudian dilakukan optimasi untuk menentukan penjadwalan yang optimal, yaitu pengoperasian pembangkit yang memenuhi batasan teknik dan memerlukan biaya serendah-rendahnya. Metode yang diterapkan dalam proses optimasi ini adalah metode *dynamic programming*.

4.1. Pengolahan Data

Pembangkit termal merupakan pembangkit yang mempunyai biaya konsumsi bahan bakar paling besar dari seluruh pembiayaan yang ada, selain biaya pelumas dan biaya *start up*. Persamaan bahan bakar dari masing-masing pembangkit dapat didekati dengan persamaan linear atau dengan persamaan polinomial derajat dua. Pendekatan dengan persamaan polinomial derajat dua lebih mendekati kenyataan, seperti ditunjukkan oleh persamaan (2.4). Dalam penentuan konstanta tersebut di atas data yang dibutuhkan adalah daya maksimum pembangkit, data *heat rate* rata-rata, harga bahan bakar, dan nilai kalor. Sebagai contoh perhitungan untuk unit 1 PLTU Suralaya 75, dengan data sebagai berikut:

- Daya maksimum pembangkit = 600 MW
- *Heat rate* rata-rata pada beban 50 % = x_1 = 2.667 (kCal/kWh)
- *Heat rate* rata-rata pada beban 75 % = x_1 = 2.450 (kCal/kWh)
- *Heat rate* rata-rata pada beban 100 % = x_1 = 2.410 (kCal/kWh)
- Harga bahan bakar batu bara = 63,50 (Rp/kg)
- Nilai Kalor = 3.000 (kCal/kg)

Dari data di atas dapat disusun persamaan regresi dengan model polinomial sebagai berikut:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} * \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i.y_i \\ \sum x_i^2.y_i \end{bmatrix}$$

- Jumlah data n = 3,
- Bahan bakar yang dibutuhkan adalah: $y_i = \frac{P_i * HR_i}{NK}$

dengan, P_i = daya ke-i; HR_i = *Heat rate* rata-rata pada daya ke-i, dan NK = Nilai Kalor,

- Pada beban 50 % = 300 MW ; y_1 = 266.700 kg/h,
- Pada beban 75 % = 450 MW ; y_1 = 367.500 kg/h,
- Pada beban 100 % = 600 MW ; y_1 = 482.000 kg/h.

Selanjutnya dicari jumlah kumulatif sebagaimana yang dibutuhkan pada persamaan matrik tersebut, sebagai berikut:

Tabel 4.1 Nilai yang dibutuhkan dalam persamaan regresi dengan model polinomial

%Beban	x_1	y_1	x_1^2	x_1^3	x_1^4	$x_1 * y_1$	$x_1^2 * y_1$
50%	300	266700	90000	27000000	8100000000	80010000	24003000000
75%	450	367500	202500	91125000	41006250000	165375000	74418750000
100%	600	482000	360000	216000000	129600000000	289200000	173520000000
Jumlah	1350	1116200	652500	334125000	178706250000	534585000	271941750000

Matrik yang dapat dibuat adalah:

$$A = \begin{bmatrix} 3 & 1350 & 652500 \\ 1350 & 652500 & 334125000 \\ 652500 & 334125000 & 178706250000 \end{bmatrix}$$

dan matrik y adalah ;

$$y = \begin{bmatrix} 1116200 \\ 534585000 \\ 271941750000 \end{bmatrix}$$

Nilai konstanta a, b, c didapatkan dari penyelesaian kedua matrik tersebut dengan persamaan A*c = y, akan didapatkan nilai konstanta, a = 0,30444, b = 443,667, c = 106.200. Dari hasil ini maka persamaan konsumsi bahan bakar tiap jamnya adalah:

$$HR_1(P_1) = 0{,}30444 * P_1^2 + 443{,}667 * P_1 + 106.200 \quad kg/h$$

sedangkan persamaan biaya bahan bakar merupakan hasil kali banyaknya konsumsi bahan bakar yang dibutuhkan tiap jam dan harga bahan bakar per kg, sehingga didapatkan persamaan biaya bahan bakar sebagai berikut:

$F_1(P_1)$ = (Harga_Bahan_Bakar)*$HR_1(P_1)$

Untuk unit 1 ini harga bahan bakar batu bara = 63,50 (Rp/kg), sehingga persamaan biaya bahan bakarnya menjadi:

$$F_1(P_1) = 63{,}50 * [0{,}30444 * P_1^2 + 443{,}667 * P_1 + 106.200]$$
$$F_1(P_1) = 19{,}33 * P_1^2 + 28172{,}83 * P_1 + 6.743.700{,}00 \quad Rp/h$$

Dengan cara yang sama didapatkan persamaan biaya bahan bakar untuk masing-masing unit pembangkit ditunjukkan pada **Lampiran 5**.

Dari persamaan biaya bahan bakar yang sudah didapakan tersebut kemudian dilakukan daftar prioritas untuk menentukan jumlah biaya yang dikeluarkan per-kWh pada setiap unit pembangkit. Daftar prioritas dapat dilakukan dengan menghitung biaya produksi rata-rata beban penuh (AFLC), yaitu:

$$AFLC = \frac{F(P_{mak})}{P_{mak}} \quad \text{Rp/kWh}$$

Untuk unit 1, PLTU Suralaya 75, yang mempunyai P_{mak} = 600 MW, biaya produksi rata-rata beban penuh (AFLC) adalah sebagai berikut:

$$AFLC = \frac{(19{,}33 * 600^2 + 28172{,}83 * 600 + 6.743.700{,}00)}{600}$$
$$= 51.010{,}00 \ Rp/MWh$$

Biaya produksi rata-rata beban penuh untuk masing-masing unit terdapat dalam **Lampiran 4**.

Dari daftar ini akan menjadi urutan prioritas dalam proses UC, maksudnya adalah bahwa unit 1 PLTU Suralaya 75 akan menjadi prioritas utama daripada unit berikutnya yang mempunyai nilai AFLC yang lebih besar.

4.2. Rancangan Program Aplikasi

Untuk menyelesaikan skripsi ini diperlukan program aplikasi perangkat lunak yang ditulis dalam bahasa program C++. Alat yang dibutuhkan dalam skripsi ini, antara lain: satu set PC, perangkat lunak Borland C++, dan Program Visual Borland C++ Builder. Spesifikasi PC minimum memiliki RAM 32MB dan processor P100 ke atas, untuk kelancaran pengolahan data.

4.2.1. Diagram alir program aplikasi

Permasalahan UC dapat diselesaikan dengan metode DP, yang menggunakan langkah maju (*forward*) untuk menganalisis beban yang selalu berubah-ubah, melalui langkah-langkah sebagai berikut:

- Menentukan kombinasi dari unit-unit pembangkit yang akan diselidiki. Suatu kombinsi menunjukkan sederetan status pembangkit, dimana suatu unit *on line*

ditandai dengan angka 1 dan untuk kondisi off line ditandai dengan angka 0. Banyaknya kombinasi yang mungkin dari sejumlah N pembangkit adalah sebanyak (2^N – 1). Kombinasi ke-J pada jam ke-H dinyatakan sebagai state (H,J).

- Dapatkan kombinasi yang layak pada tahap (jam) tertentu. Dimana syarat bahwa suatu kombinasi layak adalah dengan mempertimbangkan kendala-kendala yang ada, yaitu bahwa :
- Kesetimbangan daya pembangkitan dengan beban sebagaimana pada rumusan (2.2), dimana daya beban berada diantara pembangkitan daya minimum dan maksimum unit *on line*. Kondisi tidak layak apabila tidak memenuhi persamaan di atas.
- Kendala cadangan berputar yaitu sebagaimana pada persamaan (2.3)
- Untuk state yang layak tersebut pada tiap jamnya lakukan pembebanan ekonomis untuk setiap unit pembangkit *on line*. Sehingga didapatkan biaya operasi pembangkitan untuk tiap-tiap state tersebut untuk melayani beban yang ada. Pada skripsi ini pembebanan ekonomis berdasarkan *average full load cost* (AFLC).
- Menentukan keputusan optimal secara rekursif pada tiap jamnya, dengan rumusan matematis sebagai berikut :

$$F_{\cos t}(H,J) = \min_{\{K\}}[P_{\cos t}(H,J) + S_{\cos t}(H-1,K:H,J)] + F_{\cos t}(H-1,K)$$

dengan,

$F_{cost}(H,J)$: Biaya total terendah untuk sampai state (H,J)

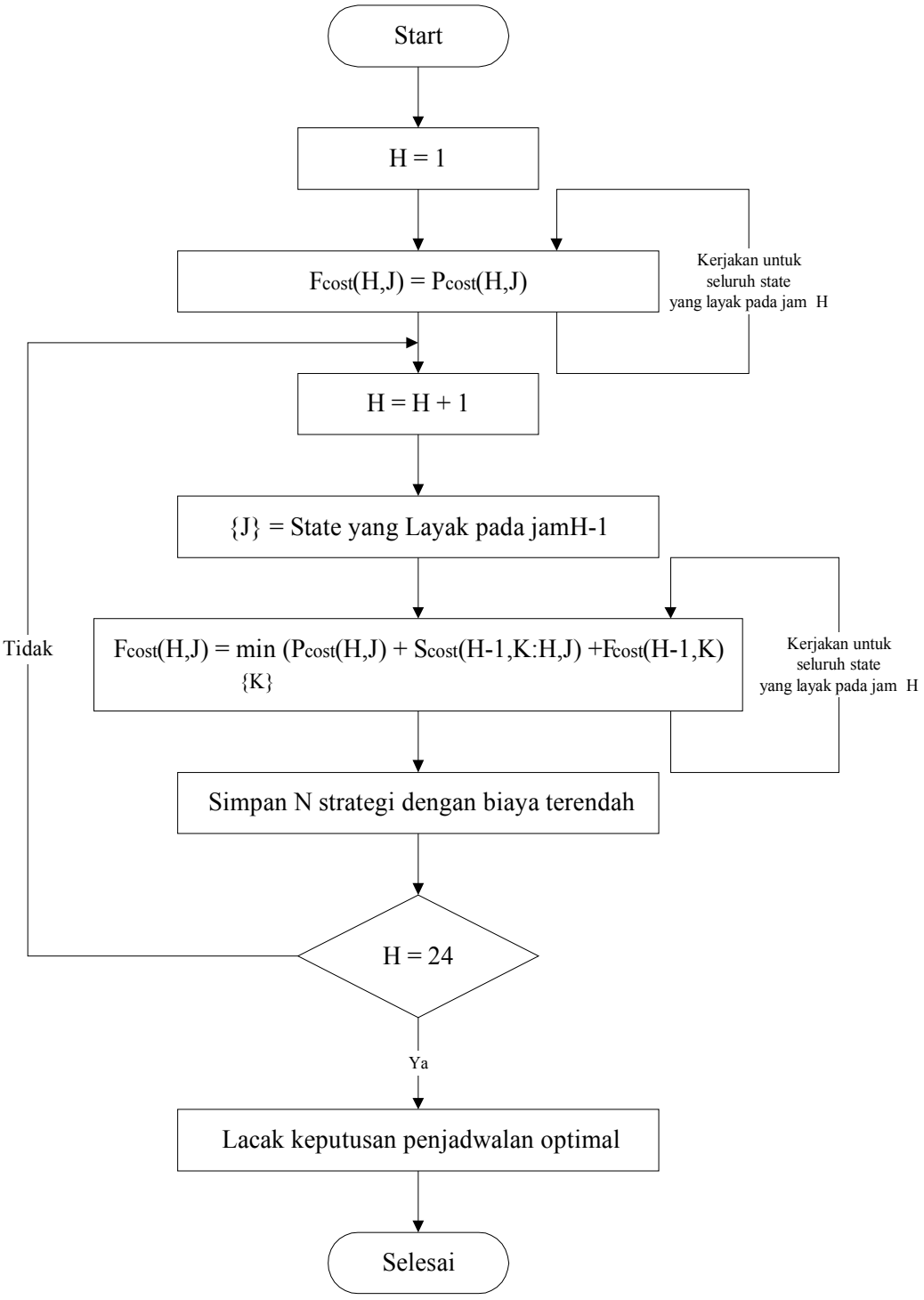
$S_{cost}(H\text{-}1,L : H,J)$: Biaya transisi dari state $(H\text{-}1,K)$ ke state (H,J)

$P_{cost}(H,J)$: Biaya produksi untuk state (H,J).

{K} : Kombinasi yang layak pada jam H-1

- Setelah mendapatkan keputusan optimal sampai jam terakhir, status pembangkitan unit tiap jamnya dari awal sampai akhir kurun studi, dapat ditentukan dengan melakukan pelacakan jejak kembali melalui jalur transisi optimal.

Dari uraian di atas algoritma dari DP dapat digambarkan dalam diagram alir berikut ini:



Gambar 4.1 Diagram alir *unit commitment* dengan metode *dynamic programming*

4.2.2. Fungsi-fungsi yang dirancang untuk program aplikasi

Program aplikasi DP, terdiri dari 10 fungsi dan satu fungsi utama. Berikut ini adalah daftar fungsi-fungsi tersebut dan penjelasannya:

- void InisialDP(void)

Fungsi ini digunakan untuk menghasilkan kondisi awal secara acak dari unit-unit pembangkit yang bernilai satu (1) apabila unit pembangkit tersebut terhubung/*commit*, dan bernilai nol (0) apabila unit pembangkit tersebut *offline*/*decommit*. Bilangan 0 dan 1 dihasilkan oleh fungsi secara acak. Karena merepresentasikan keadaan seluruh unit pembangkit setiap jam-nya dibutuhkan kombinasi bilangan biner sejumlah *array* [NOGEN], sepanjang *array* [WAKTU], dan sebanyak *array* [STATE]. NOGEN adalah jumlah unit pembangkit yang akan dioptimasi. WAKTU adalah rentang waktu optimasi, dan STATE adalah kondisi yang *feasible*/mungkin dari pembangkit untuk menyuplai beban.

- void LimPowMinDP(void)

  Fungsi ini digunakan untuk menghitung limit daya minimum dari unit-unit pembangkit yang beroperasi pada setiap jam, dengan menjumlahkan daya minimum *Pmin* semua pembangkit yang beroperasi. Hasil perhitungan ini akan digunakan untuk mengevaluasi kelayakan kapasitas daya terhadap beban yang harus dipenuhi setiap jamnya.

- void LimPowMaxDP(void)

  Fungsi ini digunakan untuk menghitung limit daya maksimum dari unit-unit pembangkit yang beroperasi pada setiap jam, dengan menjumlahkan daya minimum *Pmaks* semua pembangkit yang beroperasi. Hasil perhitungan ini akan digunakan untuk mengevaluasi kelayakan kapasitas daya terhadap beban yang harus dipenuhi setiap jamnya.

- void GenOLFeasible(void)

  Pada tahap ini telah diketahui total daya maksimum dan daya minimum unit pembangkit yang beroperasi setiap jam-nya. Fungsi ini digunakan untuk mengevaluasi kelayakan tiap unit pembangkit untuk menyuplai beban. Apabila tidak layak, yaitu apabila daya beban PB > dari daya maksimum yang dapat dibangkitkan, maka harus mengoperasikan unit pembangkit tambahan. Penambahan unit pembangkit ini disesuaikan dengan daftar prioritas. Pada fungsi ini kelayakan juga didasarkan pada kendala-kendala yang harus dipenuhi,

seperti perputaran cadangan dan kendala keamanan. Pada fungsi ini juga menetukan pembebanan optimum untuk unit pembangkit yang beroperasi.

- void OptPowerDP(void)
  Fungsi ini digunakan untuk menentukan nilai daya optimum berdasarkan *Pmin* dan *Pmaks* yang dialokasikan terlebih dahulu pada *array*, selanjutnya menentukan daya maksimum dan daya minimum pada keadaan *state feasible* tiap waktu berdasarkan daya beban yang berlaku pada waktu tersebut.

- double PowerCostDP(int i,int j,int k)
  Fungsi ini digunakan untuk menentukan biaya bahan bakar dan pelumas yang harus dikeluarkan setiap MWh energi listrik yang dihasilkan sesuai dengan persamaan biaya pada tiap unit pembangkit yang beroperasi.

- void EvalCostDP(void)
  Fungsi ini merupakan kelanjutan dari fungsi PowerCostDP (int i, int j, int k), fungsi ini digunakan untuk menjumlahkan biaya pembangkitan tiap unit generator yang sedang beroperasi setiap jamnya. Sehingga dihasilkan variabel yang menyimpan nilai dari jumlah biaya pembangkitan setiap jam. Biaya tersebut adalah biaya bahan bakar dan biaya pelumas.

- void BubleSort(void)
  Fungsi ini digunakan untuk mensortir kondisi pembangkit pada tiap jam-nya, kemudian menyimpan hasil kondisi tersebut, sehingga dapat digunakan untuk mengambil keputusan pada langkah selanjutnya tanpa mengulang langkah yang sama.

- void MinUpDown( )
  Fungsi ini digunakan untuk memeriksa kendala waktu minimum *up* dan *down*. Tiap unit pembangkit termal memiliki nilai waktu minimum *up* dan *down* yang berbeda-beda, sehingga harus dilakukan pemeriksaan kondisi pembangkit yang *feasible* tiap unitnya pada waktu tertentu.

- void StartUpCostDP( )

Fungsi ini digunakan untuk menghitung biaya start up unit pembangkit secara keseluruhan tiap unitnya. Biaya start up hanya diperlukan untuk unit-unit pembangkit yang sebelumnya tidak beroperasi apabila akan dioperasikan pada jam berikutnya. Unit pembangkit yang pada saat atau jam sebelumnya beroperasi tidak memerlukan biaya start up apabila akan dioperasikan lagi pada jam berikutnya.

- void main(void)

  Fungsi main ( ) adalah fungsi utama dalam pemrograman C++, fungsi ini digunakan untuk mengakses keseluruhan prosedur. Kemudian pada fungsi ini dilakukan proses iterasi sampai mendapatkan nilai yang paling optimal.

4.2.3. Validasi program

Validasi program dalam skripsi ini bertujuan untuk meyakinkan bahwa perhitungan dan program aplikasi yang digunakan adalah valid atau sahih. Hal ini dapat dilakukan dengan membandingkan hasil perhitungan program dengan penelitian lain sebelumnya yang telah diketahui hasilnya dengan data yang sama. Pada skripsi ini validasi program dilakukan dengan membandingkan pada penelitian yang dilakukan oleh A.G Bakirtzis dkk. (1996).

Tabel 4.2 Perbandingan hasil program aplikasi DP dengan standart IEEE

Standart IEEE ($) [(a)]	Hasil dg. DP ($)	% Beda dg. (a)
565.825,00	567.306,00	0,262

Perbedaan nilai optimum yang dihasilkan dari program aplikasi dengan metode DP berbeda sebesar 0,262% lebih besar dari Standard IEEE. Hal ini mungkin disebabkan oleh penggunaan satu strategi untuk mendapatkan nilai total optimum pada program aplikasi, sedangkan pada standart IEEE tidak dijelaskan berapa jumlah strategi yang digunakan, sehingga terdapat kemungkinan digunakan lebih dari satu strategi. Kemungkinan lainnya adalah status unit-unit *on-off* dan cadangan perputaran tidak dilaporkan sehingga tidak dapat dibandingkan secara langsung.

4.3. Hasil Pengujian dengan Program Aplikasi

Objek penelitian yang dilakukan pada skripsi ini dan data-data untuk karakteristik unit pembangkit dan peramalan beban diperoleh dari data sekunder untuk

semua pembangkit termal yang ada di wilayah PT. PLN Pembangkitan Jawa Bali I (PJBI) Area Operasi I. jumlah pembangkit termal sebanyak 26 pembangkit, yaitu 13 unit pembangkit PLTU, 1 unit pembangkit PLTP, 8 unit pembangkit PLTGU, dan 4 unit pembangkit PLTG. Seluruh data dari 26 pembangkit terdapat di **Lampiran A**.

4.3.1. Hasil pengujian dengan metode *Dynamic Programming*

Berdasarkan hasil pengujian menggunakan program aplikasi dengan metode *dynamic programming*, setelah melakukan 10 kali pengujian, didapatkan jumlah *state* yang memenuhi kebutuhan beban setiap jamnya seperti ditunjukkan oleh Tabel 4.3 berikut:

Tabel 4.3 Jumlah *state feasible* setiap jam

Waktu ke-	Jumlah *State Feasible*
0	4085
1	4048
2	3965
3	3953
4	4091
5	4096
6	4091
7	3646
8	2750
9	2463
10	2710
11	3277
12	2499
13	1928
14	2463
15	1928
16	1507
17	972
18	621
19	825
20	1302
21	2794
22	3893
23	4085

Hasil pengujian untuk biaya minimum setiap jamnya dengan jumlah state yang memenuhi berdasarkan karakteristik beban, ditunjukkan pada Tabel 4.4.

Tabel 4.4 Biaya minimum setiap jam

Jam ke-	State	Biaya Minimum (Rp)
0	53	231.358.275,80
1	164	235.981.755,60
2	374	240.693.865.20
3	139	242.268.582,10
4	59	227.120.586,40
5	0	216.340.971,00
6	59	228.491.292,00
7	1645	249.193.005,20
8	2702	259.960.758,00
9	978	262.693.469,70
10	2662	260.346.756,30
11	1427	254.248.511,80
12	1008	261.920.895,70
13	1912	266.247.584,50
14	2447	263.258.391,30
15	1912	267.020.158,50
16	1491	271.704.112,80
17	956	278.912.265,80
18	605	284.709.168,40
19	809	280.541.728,90
20	1286	275.279.757,50
21	2746	259.278.546,30
22	832	243.171.457,10
23	53	230.771.484,70

Berdasarkan hasil pengujian didapatkan ketentuan penjadwalan untuk 26 unit pembangkit termal selama 24 jam sesuai dengan karakteristik beban setiap jam, ditunjukkan pada **Lampiran 6** dan nilai pembebanan setiap unit pembangkitnya ditunjukkan pada **Lampiran 7**.

Hasil akhir dari pengujian ini, didapatkan nilai biaya total bahan bakar yang optimal, seperti ditunjukkan pada Tabel 4.5.

Tabel 4.5 Hasil pengujian menggunakan metode DP (*dynamic programming*)

Hasil Pengujian menggunakan Metode DP	
Biaya Optimal	Rp 6.091.513.380,60
Rerata Waktu	0.1 detik
Flop	4.182.600

Perputaran cadangan minimum yang disyaratkan dalam skripsi ini adalah sebesar 600 MW, hal ini disebabkan unit dengan pembangkitan terbesar dipergunakan pada setiap waktu studi adalah 600 MW. Sehingga apabila terjadi *forced outage* untuk unit dengan kapasitas 600 MW, pembangkitan masih mampu melayani beban.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian yang telah dilakukan, dapat disimpulkan antara lain sebagai berikut:

1. Berdasarkan pengujian yang sudah dilakukan dari program aplikasi menggunakan metode *dynamic programming* telah dapat menyelesaikan penjadwalan pembangkit tenaga listrik. Adapun penjadwalan dan pembebanan ekonomis yang optimal dari pembangkit tenaga listrik untuk sistem 26 unit pembangkit termal telah diberikan pada **Lampiran 6 dan 7**.
2. Untuk menyelesaikan masalah *unit commitment* dengan metode *dynamic programming*, disyaratkan untuk menerjemahkan permasalahan yang ada ke dalam sub-sub masalah. Setiap sub masalah dicari nilai optimalnya berdasarkan fungsi objektif yang telah ditentukan sebelumya. Solusi terbaik didapatkan dengan menjumlahkan setiap nilai optimal dari masing-masing sub masalah.
3. Dari hasil aplikasi dengan metode DP didapatkan nilai optimal biaya pembangkitan dari unit pembangkit yang *feasible* setiap jamnya selama 24 jam. Selanjutnya, nilai optimal biaya pembangkitan untuk tiap jam dijumlahkan sehingga didapatkan nilai optimal dari biaya total bahan bakar minimum sebesar Rp 6.091.513.380,58 dan besar pembebanan dari unit pembangkit yang *feasible* serta penjadwalannya ditunjukkan pada **Lampiran**.

5.2 Saran

1. Perlu dikembangkan algoritma yang menampung lebih banyak kendala (*constraint*) sehingga hasil yang didapatkan lebih mendekati keadaan sesungguhnya, seperti kendala transmisi, ketersediaan tenaga operator, suplai bahan bakar dan lain-lain.
2. Perlu dikembangkan algoritma dengan metode pemrograman dinamis yang lebih baik, dengan menggabungkan beberapa metode untuk mendapatkan hasil uji yang mendekati kenyataan di lapangan.

DAFTAR PUSTAKA

Barkirtzis A.G., Kazarlis S. A.,& Ptridis V. 1996. "*A genetic algorithm solution to the unit commitment problem*", IEEE Transaction on power systems, Vol. 11, No. 1, pp. 83-90.

Dasgupta Dispankar & Mcgregor R Douglas. 1993. "*Short Term Unit Commitment Using Genetic Algorithm*. Technical Report No IKBS-16-93, 10 Agustus.

Gonen Turan. 1986. *Electric Power Distribution System Engineering*. New York: McGraw-Hill, Inc.

Levitin Anany & Levitin Maria. 2011. *"Algorithmic Puzzels"*, New York: Oxford University Press.

Nagrath, I. J & Kothari, D. P. 2008. "*Power System Engineering, Second Edition*". New Delhi: McGraw-Hill, Inc.

Sivanagaraju, S & Sreenivasan, G. 2009. "*Power System Operation and Control*". India: Pearson Education.

Stanton K Neil, Giri C Jay & Bose Anjan. 2007. *Electric Power Enginering Handbook Second Edition*, *Power System Stability and Control* "*Energy Management*". New York: CRC Press Taylor & Francis Group.

Suyono Hadi. 2000. "*Optimasi Unit Pembangkit Tenaga Listrik dengan Algoritma Genetik*" Tesis tidak dipublikasikan. Jogjakarta: Universitas Gajah Mada.

Wood J Allen & Wollenberg F Bruce. 1984. *"Power Generation, Operation and Control"*. Singapore: John Wiley and Sons.

Zhu Jizhong, 2009, *"Optimization of Power Sistem Operation". New Jersey:* John Wiley and Sons.

LAMPIRAN A

Lampiran 1: **Data unit pembangkit dengan daya maksimum (MW) dan daya minimum (MW)**

No	Jenis BB	Jenis	Nama Unit	P_{MAX}	P_{MIN}
1	B.BARA	PLTU	SLAYA75	600,00	300,00
2	B.BARA	PLTU	SLAYA76	600,00	300,00
3	B.BARA	PLTU	SLAYA77	600,00	300,00
4	B.BARA	PLTU	SLAYA71	400,00	150,00
5	B.BARA	PLTU	SLAYA72	400,00	150,00
6	B.BARA	PLTU	SLAYA73	400,00	150,00
7	B.BARA	PLTU	SLAYA74	400,00	150,00
8	MFO	PLTU	MKRNGU&5	200,00	100,00
9	MFO	PLTU	MKRNGU&4	200,00	100,00
10	MFO	PLTU	PROKU	90,00	25,00
11	MFO	PLTU	MKRNGU&1	100,00	45,00
12	MFO	PLTU	MKRNGU&2	100,00	45,00
13	MFO	PLTU	MKRNGU&3	100,00	45,00
14	GAS	PLTP	SALAK	148,50	41,25
15	GAS	PLTGU	PRIOK11	130,00	70,00
16	GAS	PLTGU	PRIOK12	130,00	70,00
17	GAS	PLTGU	PRIOK13	130,00	70,00
18	GAS	PLTGU	PRIOK21	130,00	70,00
19	GAS	PLTGU	PRIOK22	130,00	70,00
20	GAS	PLTGU	PRIOK23	130,00	70,00
21	GAS	PLTGU	PRIOK10	200,00	100,00
22	GAS	PLTGU	PRIOK20	200,00	100,00
23	GAS	PLTG	PROKG55	48,80	24,00
24	GAS	PLTG	PROKG56	48,80	24,00
25	GAS	PLTG	PROKG57	48,80	24,00
26	HSD	PLTG	PROKG54	48,80	24,00
Jumlah Total Daya				5.713,70	2.617,25

Sumber: Hadi Suyono (2000: Lampiran)

Lampiran 2: **Biaya *start-up* untuk masing-masing unit pembangkit dan biaya pelumas**

No	Unit	Nama Pembangkit	Biaya *Start Up* (Rp)	Jenis BB	Pelumas (Rp/MWH)	B.Bakar (Rp/kg)
1	PLTU	SLAYA75	10.668.720,00	B.BARA	38	63,50
2	PLTU	SLAYA76	10.668.720,00	B.BARA	38	63,50
3	PLTU	SLAYA77	10.668.720,00	B.BARA	38	63,50
4	PLTU	SLAYA71	7.823.728,00	B.BARA	22	17,15
5	PLTU	SLAYA72	7.823.728,00	B.BARA	22	17,15
6	PLTU	SLAYA73	7.823.728,00	B.BARA	22	17,15
7	PLTU	SLAYA74	7.823.728,00	B.BARA	22	17,15
8	PLTU	MKRNGU&5	3.556.240,00	MFO	23	17,15
9	PLTU	MKRNGU&4	1.082.670,00	MFO	23	17,15
10	PLTU	PROKU	2.316.544,00	MFO	6	17,15
11	PLTU	MKRNGU&1	2.461.328,00	MFO	24	17,15
12	PLTU	MKRNGU&2	2.461.328,00	MFO	24	73,33
13	PLTU	MKRNGU&3	2.461.328,00	MFO	24	73,33
14	PLTP	SALAK	3.856.240,00	GAS	31	73,33
15	PLTGU	PRIOK11	703.736,00	GAS	13	73,33
16	PLTGU	PRIOK12	703.736,00	GAS	13	385,00
17	PLTGU	PRIOK13	703.736,00	GAS	13	80,74
18	PLTGU	PRIOK21	703.736,00	GAS	13	385,00
19	PLTGU	PRIOK22	703.736,00	GAS	13	17,15
20	PLTGU	PRIOK23	703.736,00	GAS	13	17,15
21	PLTGU	PRIOK10	1.082.670,00	GAS	13	17,15
22	PLTGU	PRIOK20	1.082.670,00	GAS	13	385,00
23	PLTG	PROKG55	447.518,00	GAS	76	385,00
24	PLTG	PROKG56	447.518,00	GAS	76	385,00
25	PLTG	PROKG57	447.518,00	GAS	76	385,00
26	PLTG	PROKG54	895.036,00	HSD	1138	585,00

Sumber: Hadi Suyono (2000: Lampiran)

Lampiran 3: **Konstanta persamaan biaya untuk masing-masing unit pembangkit** $\{F(P_i) = a.P_i^2 + b.P_1 + c\}$

No	Jenis	Nama Unit	c (Rp)	b (Rp/MWh)	a(Rp/MW^{2}h)
1	PLTU	SLAYA75	6.743.700,00	28.172,86	19,33
2	PLTU	SLAYA76	6.743.700,00	28.172,86	19,33
3	PLTU	SLAYA77	6.743.700,00	28.172,86	19,33
4	PLTU	SLAYA71	5.191.690,67	32.534,69	33,49
5	PLTU	SLAYA72	5.191.690,67	32.534,69	33,49
6	PLTU	SLAYA73	5.191.690,67	32.534,69	33,49
7	PLTU	SLAYA74	5.191.690,67	32.534,69	33,49
8	PLTU	MKRNGU&5	620.812,50	74.709,25	11,55
9	PLTU	MKRNGU&4	606.375,00	77.423,50	11,55
10	PLTU	PROKU	661.045,00	85.855,00	38,50
11	PLTU	MKRNGU&1	1.739.718,75	68.433,75	165,55
12	PLTU	MKRNGU&2	1.674.750,00	66.797,50	215,60
13	PLTU	MKRNGU&3	1.458.187,50	79.040,50	130,90
14	PLTP	SALAK	3.956.637,30	50.867,90	21,72
15	PLTGU	PRIOK11	1.226.087,80	40.319,65	56,60
16	PLTGU	PRIOK12	1.226.087,80	40.319,65	56,60
17	PLTGU	PRIOK13	1.226.087,80	40.319,65	56,60
18	PLTGU	PRIOK21	1.226.087,80	40.319,65	56,60
19	PLTGU	PRIOK22	1.226.087,80	40.319,65	56,60
20	PLTGU	PRIOK23	1.226.087,80	40.319,65	56,60
21	PLTGU	PRIOK10	1.886.551,45	40.216,75	37,73
22	PLTGU	PRIOK20	1.886.551,45	40.216,75	37,73
23	PLTG	PROKG55	1.556.654,05	51.998,80	51,45
24	PLTG	PROKG56	1.556.654,05	51.998,80	51,45
25	PLTG	PROKG57	1.556.654,05	51.998,80	51,45
26	PLTG	PROKG54	3.820.648,46	73.009,76	582,08

Sumber: Perhitungan

Lampiran 4: ***Average Full Load Cost*/AFLC (Rp/MWh), dan probabilitas *forced outage rate* (FOR)**

No	Jenis	Nama Pembangkit	FOR	AFLC (Rp/MWh)
1	PLTU	SLAYA75	0,0216	51.009,99
2	PLTU	SLAYA76	0,0245	51.009,99
3	PLTU	SLAYA77	0,0120	51.009,99
4	PLTU	SLAYA71	0,0005	58.908,37
5	PLTU	SLAYA72	0,0642	58.908,37
6	PLTU	SLAYA73	0,0014	58.908,37
7	PLTU	SLAYA74	0,0095	58.908,37
8	PLTU	MKRNGU&5	0,2340	80.123,31
9	PLTU	MKRNGU&4	0,1137	82.765,37
10	PLTU	PROKU	0,0289	96.664,94
11	PLTU	MKRNGU&1	0,0504	102.385,93
12	PLTU	MKRNGU&2	0,0289	105.105,00
13	PLTU	MKRNGU&3	0,1408	106.712,37
14	PLTP	SALAK	0,0536	80.737,19
15	PLTGU	PRIOK11	0,0005	57.108,44
16	PLTGU	PRIOK12	0,0166	57.108,44
17	PLTGU	PRIOK13	0,0582	57.108,44
18	PLTGU	PRIOK21	0,0032	57.108,44
19	PLTGU	PRIOK22	0,0157	57.108,44
20	PLTGU	PRIOK23	0,0485	57.108,44
21	PLTGU	PRIOK10	0,0012	57.195,50
22	PLTGU	PRIOK20	0,0197	57.195,50
23	PLTG	PROKG55	0,0285	86.408,21
24	PLTG	PROKG56	0,0180	86.408,21
25	PLTG	PROKG57	0,0145	86.408,21
26	PLTG	PROKG54	0,0155	179.706,99

Sumber: Perhitungan

Lampiran 5: **Contoh data beban harian di PLN-Jaya wilayah operasi area 1 (MW) (*16 Desember 1999*)**

Jam-ke	Beban Area-1	Total Pembangkitan	Sisa
1	3866,3	5.713,7	1.213,7
2	3941,3	5.713,7	1.172,4
3	4016,3	5.713,7	1.097,4
4	4041,3	5.713,7	1.072,4
5	3791,3	5.713,7	1.322,4
6	3591,3	5.713,7	1.522,4
7	3816,3	5.713,7	1.297,4
8	4154,3	5.713,7	959,4
9	4326,3	5.713,7	787,4
10	4366,3	5.713,7	747,4
11	4331,3	5.713,7	782,4
12	4234,3	5.713,7	879,4
13	4356,3	5.713,7	757,4
14	4426,3	5.713,7	687,4
15	4376,3	5.713,7	737,4
16	4436,3	5.713,7	677,4
17	4496,3	5.713,7	617,4
18	4586,3	5.713,7	527,4
19	4654,3	5.713,7	459,4
20	4606,3	5.713,7	507,4
21	4541,3	5.713,7	572,4
22	4316,3	5.713,7	797,4
23	4056,3	5.713,7	1.057,4
24	3856,3	5.713,7	1.257,4

Sumber: Hadi Suyono (2000: Lampiran)

Lampiran 6: **Penjadwalan 26 Unit Pembangkit Termal**

		Unit Pembangkit ke-																										
Waktu Ke-	**State ke-**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**
0	53	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
1	164	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0
2	374	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0
3	139	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0
4	59	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
5	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
6	59	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
7	1645	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0
8	2702	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0
9	978	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0
10	2662	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0
11	1427	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	0	0	0
12	1008	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0
13	1912	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
14	2447	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
15	1912	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
16	1491	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
17	956	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
18	605	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
19	809	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
20	1286	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
21	2746	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0
22	832	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0
23	53	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0

Lampiran 7: **Pembebanan 26 Unit Pembangkit Termal selama 24 jam (dalam pu)**

	Unit Pembangkit ke-																								
Waktu ke-	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**
1	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0	0	0.9515	0	0	0	0	0	0
2	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0	0.9843	0.9515	0	0	0	0	0	0
3	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0.9968	0.9843	0.9515	0	0	0	0	0	0
4	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0	0.9843	0.9515	0	0.9803	0	0	0	0
5	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0	0	0.9515	0	0	0	0	0	0
6	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0	0	0	0	0	0	0	0	0
7	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0	0	0.9515	0	0	0	0	0	0
8	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0.9834	0.9418	0.9968	0.9843	0.9515	0	0	0	0	0	0
9	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0	0.9803	0	0	0	0
10	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
11	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0	0.9803	0	0	0	0
12	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0.9834	0.9418	0.9968	0.9843	0.9515	0	0.9803	0	0	0	0
13	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
14	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
15	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
16	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
17	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
18	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
19	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
20	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
21	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0.9988	0.9803	0	0	0	0
22	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0.9995	0.9834	0.9418	0.9968	0.9843	0.9515	0	0.9803	0	0	0	0
23	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0.9418	0.9968	0.9843	0.9515	0	0	0	0	0	0
24	0.0216	0.0245	0.988	0.9995	0.9358	0.9986	0.9905	0.766	0.8863	0.9711	0.9496	0.9711	0.8592	0.9464	0	0	0	0	0	0.9515	0	0	0	0	0	0

LAMPIRAN B

Lampiran 8: ***Listing* Program Aplikasi dengan Metode *Dynamic Programming***

```
#include <stdlib.h>
#include <iostream.h>
#include <iomanip.h>
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#pragma hdrstop

FILE* INSDP1;
       const int NOGEN = 26;  //Jumlah Pembangkit yang di commit
   const int WAKTU = 24;              //
       const int STATE = 4096;       //Kombinasi keadaan pembangkit

double Pmin[NOGEN]={
Pmin[0]=300,   Pmin[1]=300,          Pmin[2]=300,   Pmin[3]=150,
Pmin[4]=150,   Pmin[5]=150,          Pmin[6]=150,   Pmin[7]=100,
Pmin[8]=100,   Pmin[9]=25,           Pmin[10]=45,   Pmin[11]=45,
Pmin[12]=45,   Pmin[13]=41.25,       Pmin[14]=70,   Pmin[15]=70,
Pmin[16]=70,   Pmin[17]=70,          Pmin[18]=70,   Pmin[19]=70,
Pmin[20]=100,  Pmin[21]=100,         Pmin[22]=24,   Pmin[23]=24,
Pmin[24]=24,   Pmin[25]=24};

double Pmak[NOGEN]={
Pmak[0]=600,   Pmak[1]=600,          Pmak[2]=600,   Pmak[3]=400,
Pmak[4]=400,   Pmak[5]=400,          Pmak[6]=400,   Pmak[7]=200,
Pmak[8]=200,   Pmak[9]=90,           Pmak[10]=100,  Pmak[11]=100,
Pmak[12]=100,  Pmak[13]=148.5,       Pmak[14]=130,  Pmak[15]=130,
Pmak[16]=130,  Pmak[17]=130,         Pmak[18]=130,  Pmak[19]=130,
Pmak[20]=200,  Pmak[21]=200,         Pmak[22]=48.8, Pmak[23]=48.8,
Pmak[24]=48.8, Pmak[25]=48.8};

double PB[WAKTU]={
PB[0]=3866.3,  PB[1]=3941.3,  PB[2]=4016.3,  PB[3]=4041.3,
PB[4]=3791.3,  PB[5]=3591.3,  PB[6]=3816.3,  PB[7]=4154.3,
PB[8]=4326.3,  PB[9]=4366.3,  PB[10]=4331.3, PB[11]=4234.3,
PB[12]=4356.3, PB[13]=4426.3,        PB[14]=4376.3, PB[15]=4436.3,
PB[16]=4496.3, PB[17]=4586.3,        PB[18]=4654.3, PB[19]=4606.3,
PB[20]=4541.3, PB[21]=4316.3,        PB[22]=4056.3, PB[23]=3856.3};

double a[NOGEN]={
a[0]=19.329,   a[1]=19.329,   a[2]=19.329,   a[3]=33.486,
a[4]=33.486,   a[5]=33.486,   a[6]=33.486,   a[7]=11.550,
a[8]=11.550,   a[9]=38.500,   a[10]=165.550, a[11]=215.600,
a[12]=130.900, a[13]=21.719,  a[14]=56.595,  a[15]=56.595,
a[16]=56.595,  a[17]=56.595,  a[18]=56.595,  a[19]=56.595,
a[20]=37.730,  a[21]=37.730,  a[22]=51.450,  a[23]=51.450,
a[24]=51.450,  a[25]=582.075};

double b[NOGEN]={
b[0]=28172.855,b[1]=28172.855,b[2]=28172.855,b[3]=32534.688,
b[4]=32534.688,b[5]=32534.688,b[6]=32534.688,b[7]=74709.250,
b[8]=77423.500,  b[9]=85855.000,b[10]=68433.750,b[11]=66797.500,
b[12]=79040.500,b[13]=50867.896, b[14]=40319.650,b[15]=40319.650,
b[16]=40319.650,b[17]=40319.650,b[18]=40319.650,b[19]=40319.650,
b[20]=40216.750,b[21]=40216.750,b[22]=51998.800,b[23]=51998.800,
b[24]=51998.800,b[25]=73009.755};


double c[NOGEN]={
c[0]=6743700,c[1]=6743700,c[2]=6743700,c[3]=5191690.670,
c[4]=5191690.670,c[5]=5191690.670,c[6]=5191690.670,c[7]=620812.500,
c[8]=606375.000, c[9]=661045.000,c[10]=1739718.750,c[11]=1674750.000,
c[12]=1458187.500,c[13]=3956637.298,c[14]=1226087.800,c[15]=1226087.8
00,
c[16]=1226087.800,c[17]=1226087.800,c[18]=1226087.800,c[19]=1226087.8
00,
c[20]=1886551.450,c[21]=1886551.450,c[22]=1556654.050,c[23]=1556654.0
50,
c[24]=1556654.050, c[25]=3820648.455};

//Biaya Pelumas uuntuk masing-masing unit dalam Rp/MWh
double o[NOGEN]={
o[0]=38,o[1]=38,o[2]=38,o[3]=22,o[4]=22,o[5]=22,
o[6]=22,o[7]=6,o[8]=6, o[9]=6, o[10]=6,o[11]=6,
o[12]=6,o[13]=31,o[14]=13,o[15]=13,o[16]=13,o[17]=13,
o[18]=13,o[19]=13,o[20]=13,  o[21]=13,o[22]=6,o[23]=6,
o[24]=6,o[25]=1138};
//Biaya Start-up tiap unit pembangkit
double start[NOGEN] ={
  start[0]=10668720,   start[1]=10668720,    start[2]=10668720,
```

```
   start[3]=7823728,          start[4]=7823728,
        start[5]=7823728,
   start[6]=7823728,   start[7]=3556240,      start[8]=1082670,
   start[9]=2316544,   start[10]=2461328,     start[11]=2461328,
   start[12]=2461328,         start[13]=3856240,
        start[14]=703735.5,
   start[15]=703735.5, start[16]=703735.5,    start[17]=703735.5,
   start[18]=703735.5,        start[19]=703735.5,
        start[20]=1082670,
   start[21]=1082670,  start[22]=447518,      start[23]=447518,
   start[24]=447518,   start[25]=895036};

//Laju kegagalan tiap-tiap unit pembangkit
double FOR[NOGEN]={
FOR[0]=0.0216, FOR[1]=0.0245, FOR[2]=0.0120, FOR[3]=0.0005,
FOR[4]=0.0642, FOR[5]=0.0014, FOR[6]=0.0095, FOR[7]=0.2340,
FOR[8]=0.1137, FOR[9]=0.0289, FOR[10]=0.0504,       FOR[11]=0.0289,
FOR[12]=0.1408,        FOR[13]=0.0536,        FOR[14]=0.0005,
       FOR[15]=0.0166,
FOR[16]=0.0582,        FOR[17]=0.0032,        FOR[18]=0.0157,
       FOR[19]=0.0485,
FOR[20]=0.0012,        FOR[21]=0.0197,        FOR[22]=0.0285,
       FOR[23]=0.0180,
FOR[24]=0.0145,        FOR[25]=0.0155};

   int GTDP[STATE][NOGEN];    //keadaan pembangkit dg DP
   int ** GTD[WAKTU];
   double DAYAMAX[STATE];
        double DAYAMIN[STATE];
        double** PoptDP[WAKTU];
   int GF[WAKTU];                          //Jumlah Unit OL
Feasible/WAKTU
   double** EvaL;                          //Nilai Evaluasi tiap
Jam/State
   int BS[WAKTU];                    //Best State;
        int BestState[WAKTU];   //Keadaan State Terbaik (minimum)
        double BestEval;
   int flopDP;
        double FuelCostDP;             //Biaya bahan bakar terbaik
   double BestStartCost;      //Biaya Start Cost terbaik
        double SRDP[WAKTU];            //Spinning Reserve terbaik
        double UnSecLevDP[WAKTU];//Unsecurity Level dengan DP
   double** DayamaX;                   //Dayamak feasible
   double** DayamiN;       //Dayamin feasible
```

```
void InisialDP(void)
{
       int j, k;
       double PT[12] = {2048,1024,512,256,128,64,32,16,8,4,2,1};
   double KG;

   //Pembangkit 1 dan 2 must Run
        for(j=0;       j<STATE; j++)
        for(k=0;       k<14;   k++)
        GTDP[j][k]=1;

   for(j=0;    j<STATE; j++)
        {
      KG = j;
      for(k=14; k<NOGEN; k++)
        {
        if((int (KG/PT[k-14])) == 1)
                                        {GTDP[j][k]=1;
          KG = KG - PT[k-14];
          }
        else
          GTDP[j][k]=0;
        }
      }

   /*fprintf(INSDP1,"\nGenerator On Line terbaik sampai generasi ke-
%3d\n");
        for(j=0; j<STATE; j++)
                {
        fprintf(INSDP1,"[State-%3d] = ",j);
                for(k = 0; k < NOGEN; k++)
        {
        fprintf(INSDP1,"%2d",GTDP[j][k]);
        }
      fprintf(INSDP1,"\n");
                }
        getch();*/

}


void LimPowMinDP(void)
{
   int j,k;
   double PMIN;
```

```
   for (j=0; j<STATE; j++)
        {
      PMIN = 0;
      for(k =0; k<NOGEN; k++)
        {
          if (GTDP[j][k])
                PMIN = PMIN + Pmin[k];
        }
      DAYAMIN[j]=PMIN;
      }
   //for (j=0; j<STATE; j++)
       //fprintf(INSDP1,"daya min, pada state :
%3d=%4.3f\n",j,DAYAMIN[j]);

}

void LimPowMaxDP(void)
{
   int j,k;
   double PMAX;

   for (j=0; j<STATE; j++)
        {
      PMAX = 0;
      for(k =0; k<NOGEN; k++)
        {
          if (GTDP[j][k])
                PMAX = PMAX + Pmak[k];
        }
      DAYAMAX[j]=PMAX;
      }
   //for (j=0; j<STATE; j++)
       //fprintf(INSDP1,"daya mak, pada state :
%3d=%4.3f\n",j,DAYAMAX[j]);
}


void GenOLFeasible(void)
{
        int i, ii, j, k;

   //Pengalokasian Array untuk Daya Optimum;
        for(int i = 0; i < WAKTU; i++)
                GTD[i] = new int* [STATE];
   for(int i=0; i < WAKTU; i++)
                for(int j=0; j < STATE; j++)
                GTD[i][j] = new int [NOGEN];

   for(i = 0; i < WAKTU; i++)
      {
      ii=0;
      for(j=0; j < STATE; j++)
        {
          //if(((PB[i]+ (0.1*PB[i])) >= DAYAMIN[j]) && ((PB[i]+
(0.1*PB[i])) <= DAYAMAX[j]))
          //if(((PB[i] >= DAYAMIN[j]) && (PB[i] <= DAYAMAX[j]))
          if(((PB[i] + 600) >= DAYAMIN[j]) && ((PB[i] + 600) <=
DAYAMAX[j]))
                {
            for(k=0; k < NOGEN; k++)
              GTD[i][ii][k]=GTDP[j][k];
            ii = ii + 1;
            }
        }
      GF[i]=ii; //Jumlah State feasible
                fprintf(INSDP1,"Jumlah State Feasible Time %2d =
%4d\n",i,GF[i]);
      }
      //getch();

       /*for(i = 0; i < WAKTU; i++)
                {
      for(ii=0; ii < GF[i]; ii++)
                        {
       fprintf(INSDP1,"\n[Time%2d,%3d]:",i,ii);
      for(k=0; k < NOGEN; k++)
                fprintf(INSDP1,"%2d ",GTD[i][ii][k]);
        }
      fprintf(INSDP1,"\n");
      //getch();
      }*/
}


void OptPowerDP(void)
{
        int i,j,k;
   double RES;
   double DayaComDP,POptDP;
   double PmaX,PmiN;
```

```
        DayamaX = new double* [WAKTU];
    for(int i = 0; i < WAKTU; i++)
        DayamaX[i] = new double [GF[i]];

    DayamiN = new double* [WAKTU];
    for(int i = 0; i < WAKTU; i++)
        DayamiN[i] = new double [GF[i]];

        //Pengalokasian Array untuk Daya Optimum;
        for(int i = 0; i < WAKTU; i++)
                PoptDP[i] = new double* [GF[i]];
    for(int i=0; i < WAKTU; i++)
                for(int j=0; j < GF[i]; j++)
                PoptDP[i][j] = new double [NOGEN];



    //Penentuan daya Maksimum pada keadaan State feasible tiap waktu
        for (i=0; i<WAKTU; i++)
        for (j=0; j<GF[i]; j++)
        {
       PmaX = 0;
       for(k =0; k<NOGEN; k++)
         {
          if (GTD[i][j][k])
                PmaX = PmaX + Pmak[k];
         }
       DayamaX[i][j]=PmaX;
       //fprintf(INSDP1,"Daya Maksimum Time %2d, State %3d =
%g\n",i,j,DayamaX[i][j]);
       }

    //Penentuan daya Minimum pada keadaan State feasible tiap waktu
        for (i=0; i<WAKTU; i++)
        for (j=0; j<GF[i]; j++)
        {
       PmiN = 0;
       for(k =0; k<NOGEN; k++)
         {
          if (GTD[i][j][k])
                PmiN = PmiN + Pmin[k];
         }
       DayamiN[i][j]=PmiN;
       //fprintf(INSDP1,"Daya Minimum Time %2d, State %3d =
%g\n",i,j,DayamiN[i][j]);
                }


for(i=0; i < WAKTU; i++)
                {
    for(j=0; j < GF[i]; j++)
     {
       if(PB[i] == DayamiN[i][j])
             for(k=0; k<NOGEN; k++)
                     {
             if(GTD[i][j][k])
                             PoptDP[i][j][k] = Pmin[k];
                     else
                             PoptDP[i][j][k] = 0;
             }

       if(PB[i] == DayamaX[i][j])
             for(k=0; k<NOGEN; k++)
                     {
             if(GTD[i][j][k])
                             PoptDP[i][j][k] = Pmak[k];
                     else
                             PoptDP[i][j][k] = 0;
             }

       if((PB[i] > DayamiN[i][j]) && (PB[i] < DayamaX[i][j]))
        {
         for(k=0; k<NOGEN; k++)
                     {
             if (GTD[i][j][k])
                             PoptDP[i][j][k] = Pmin[k];
                     else
                             PoptDP[i][j][k] = 0;
             }

         DayaComDP = DayamiN[i][j];
             RES = PB[i] - DayaComDP;
             k = 0;

             while(RES != 0 )
                     {
                     if (GTD[i][j][k])
                             {
                     POptDP = PoptDP[i][j][k] + RES;
                     if(POptDP > Pmak[k])
                             POptDP = Pmak[k];
```

```
                                        DayaComDP = DayaComDP +
POptDP - PoptDP[i][j][k];
                        PoptDP[i][j][k] = POptDP;
                        RES = PB[i] - DayaComDP;
                        }
                //fprintf(INSDP1,"%3.2f ",PoptDP[i][j][k]);
                        k=k+1;
                        }
             }
          }
       }


  /* fprintf(INSDP1,"\nOptimal Power Generation\n");
   for(i=0; i<WAKTU; i++)
                {
      //fprintf(INSDP1,"PB[%d]= %g ",i,PB[i]);
      for(j=0; j<GF[i]; j++)
                        {
          fprintf(INSDP1,"[Time-%2d, %3d] PB= %g :",i,j,PB[i]);
          for(k = 0; k < NOGEN; k++)
                fprintf(INSDP1," %g ",PoptDP[i][j][k]);
         fprintf(INSDP1,"\n");
         }
      }*/

}

//Objective Function Pada setiap Generator
double PowerCostDP(int i,int j,int k)
{
   double costDP;

   costDP = (a[k]*PoptDP[i][j][k]*PoptDP[i][j][k]) +
(b[k]+o[k])*PoptDP[i][j][k] + c[k];
   return costDP;
}


//Menentukan nilai Objective Function pada setiap populasi dan setiap
Jamnya
void EvalCostDP(void)
{
       int i,j,k;
       double EvaLMin[WAKTU];
       double VMIN;

       EvaL = new double* [WAKTU];
       for(int i = 0; i < WAKTU; i++)
            EvaL[i] = new double [GF[i]];

       for(i =0; i < WAKTU; i++)
            for(j =0; j < GF[i]; j++)
                        {
             EvaL[i][j] = 0;
            for(k =0; k < NOGEN; k++)
                  if (GTD[i][j][k])
                  {
                  EvaL[i][j] = EvaL[i][j] + PowerCostDP(i,j,k);
                  flopDP = flopDP + 3;
                                        }
            }
            //Banyaknya flop untuk evalcost
            fprintf(INSDP1,"Jumlah Flopdp1 = %d\n",flopDP);
            //getch();


       /*for(i =0; i < WAKTU; i++)
            for(j =0; j < GF[i]; j++)
                  fprintf(INSDP1,"Biaya Bahan Bakar pada Jam :%2d, %3d =
 %7.3f\n",i,j,EvaL[i][j]);
        */

            //Menentukan nilai biaya Bahan Bakar minimum
       for(i =0; i < WAKTU; i++)
            {
           VMIN = 1000000000000000;
            for(j =0; j < GF[i]; j++)
            {
              if(EvaL[i][j] < VMIN)
                  {
                  VMIN = EvaL[i][j];
                  BestState[i]=j;
                  }
              }
           EvaLMin[i] = VMIN;
           }
```

```
   for(i =0; i < WAKTU; i++)
        fprintf(INSDP1,"Biaya Minimum Jam :%2d, State %3d =
%7.3f\n",i,BestState[i],EvaLMin[i]);

   FuelCostDP=0;
   for(i =0; i < WAKTU; i++)
        {
      FuelCostDP = FuelCostDP + EvaLMin[i];
      }

       fprintf(INSDP1,"Biaya Total Bahan Bakar Minimum =
%7.3f\n",FuelCostDP);

   for(i=0; i<WAKTU; i++)
                {
      fprintf(INSDP1,"[Time-%2d, %3d] PB= %g
:",i,BestState[i],PB[i]);
      for(k = 0; k < NOGEN; k++)
       fprintf(INSDP1," %g ",PoptDP[i][BestState[i]][k]);
      fprintf(INSDP1,"\n");
      }

   for(i=0; i<WAKTU; i++)
                {
      fprintf(INSDP1,"[Time-%2d, %3d] :",i,BestState[i]);
      for(k = 0; k < NOGEN; k++)
       fprintf(INSDP1,"%2d ",GTD[i][BestState[i]][k]);
      fprintf(INSDP1,"\n");
      }
}


void BubleSort(void)
{
   int i,j,k,t;
        double temp;
   int temp1, sorted;
   double temp2;

   for(i=0; i<WAKTU; i++)
        {
      t = 0;
      sorted = 0;
      while(!sorted)
        {
          sorted = 1;
          t++;
          for(j=0; j<=(GF[i]-t); j++)
                if(EvaL[i][j-1] > EvaL[i][j])
                {
                temp = EvaL[i][j-1];
                EvaL[i][j-1] = EvaL[i][j];
                EvaL[i][j] = temp;
                                        for(k=0; k<NOGEN; k++)
                                        {
                   //Sortir keadaan pembangkit;
                   temp1 = GTD[i][j-1][k];
                                                GTD[i][j-1][k] =
GTD[i][j][k];
                   GTD[i][j][k] = temp1;

                   //sortir power dispatch;
                   temp2 = PoptDP[i][j-1][k];
                                                PoptDP[i][j-1][k] =
PoptDP[i][j][k];
                   PoptDP[i][j][k] = temp2;
                   }
                sorted = 0;
                }
          }
        }

    /*for(i =0; i < WAKTU; i++)
          for(j =0; j < GF[i]; j++)
                fprintf(INSDP1,"Biaya Sortir pada Jam :%2d, %3d =
%7.3f\n",i,j,EvaL[i][j]);

        for(i = 0; i < WAKTU; i++)
                {
          for(j=0; j < GF[i]; j++)
                        {
           fprintf(INSDP1,"\n[Sortir Time%2d,%3d]:",i,j);
          for(k=0; k < NOGEN; k++)
                fprintf(INSDP1,"%2d ",GTD[i][j][k]);
           }
        fprintf(INSDP1,"\n");
        }

    fprintf(INSDP1,"\nOptimal Power Generation\n");
    for(i=0; i<WAKTU; i++)
                {
```

```
      for(j=0; j<GF[i]; j++)
                        {
          fprintf(INSDP1,"[Sortir Time-%2d, %3d] PB= %g
:",i,j,PB[i]);
          for(k = 0; k < NOGEN; k++)
               fprintf(INSDP1," %g ",PoptDP[i][j][k]);
         fprintf(INSDP1,"\n");
         }
      }*/
}


//Cek kendala min up dan min down
void MinUpDown()
{
    int i,j,k;
    int MU[NOGEN]={8,8,5,5,6,3,3,1,1,1};
    int MD[NOGEN]={8,8,5,5,6,3,3,1,1,1};
    int IS[NOGEN]={8,8,-5,-5,-6,-3,-3,-1,-1,-1};
    int TU[WAKTU][NOGEN];
    int TD[WAKTU][NOGEN];
    //int BestGTD[WAKTU][NOGEN];
        //double BestPoptDP[WAKTU][NOGEN];
    int good[NOGEN];          //Untuk mengetes hasil layak atau
tidak;
        int Feasible;                       //Kondisi feasible atau
tidak;
    //keadaan awal unit on-line
    BS[0]=0;
    for(k=2; k<NOGEN; k++)
       {
      if(GTD[0][BS[0]][k]==0)
        {
          //Inisial negatif
          if((IS[k] + abs(IS[k])) == 0)
               {
             TD[0][k]=IS[k]-1;
             TU[0][k]=0;
             }
          //Inisial positif
          else
               {
             TU[0][k]=0;
             TD[0][k]=-1;
             }
       }

//Untuk GTD[0][0][k] = 1;
else
  {
    //Inisial negatif
    if((IS[k] + abs(IS[k])) == 0)
         {
       TD[0][k]=0;
       TU[0][k]=1;
       }
    //Inisial positif
    else
         {
       TU[0][k]=IS[k]+1;
       TD[0][k]=0;
       }
    }
}

for(i=1; i<WAKTU; i++)
    {
    j = 0;
    Feasible = 0;
    while(!Feasible)
         {
         for(k=2; k<NOGEN; k++)
                                 {
         if(GTD[i-1][BS[i-1]][k]==1)
                 {
                 if(GTD[i][j][k]==0)
                         {
                         if(TU[i-1][k] >= MU[k])       //Time up
minimum waktu lalu memenuhi Min Up
                                 {
                 TU[i][k]= 0;
                 TD[i][k]=1;
                 good[k]=1;     //feasible
                 }
                 else
  //Time Up Min waktu yg lalu tidak memenuhi
                                 good[k]=0;     //tidak feasible
                 }
                 else
                 {
                 TU[i][k]=TU[i-1][k]+1;
                 TD[i][k]=0;
                 good[k]=1;     //feasible
```

```
                    }
                    }
            //BestGTD waktu yang lalu kondisinya = 0;
            else
                    {
                    if(GTD[i][j][k]==1)
                            {
                            if(abs(TD[i-1][k]) >= MD[k])  //Time
Down minimum waktu lalu memenuhi Min Up
                            {
                    TU[i][k]=1;
                    TD[i][k]=0;
                    good[k]=1;      //feasible
                    }
                    else
      //Time Down Min waktu yg lalu tidak memenuhi
                            good[k]=0;      //tidak feasible
                    }
                    else
                    {
                    TD[i][k]=TD[i-1][k]-1;
                    TU[i][k]=0;
                    good[k]=1;      //feasible
                    }
                    }
            }//batas k
            //Pengecekan apakah untuk semua unit k kondisinya
feasible (good);
            Feasible = 1;
            for(k=2; k<NOGEN; k++)
                    {
            Feasible = Feasible * good[k];
            flopDP = flopDP + 1;
            }

            if(Feasible)
                    BS[i]=j;
            else
            {
                    j++;
            //for(k=0; k<NOGEN; k++)

            }
            }//Batas while
        }
    for(i=0; i<WAKTU; i++)
                    fprintf(INSDP1,"Best State time %2d = %3d\n",i,BS[i]);

            for(i=0; i<WAKTU; i++)
                    fprintf(INSDP1,"Best Eval State %2d =
%g\n",BS[i],EvaL[i][BS[i]]);

            BestEval = 0;
            for(i=0; i<WAKTU; i++)
              BestEval = BestEval + EvaL[i][BS[i]];

                    fprintf(INSDP1,"\nBest Eval = %g\n",BestEval);


            for(i = 0; i < WAKTU; i++)
                            {
                    fprintf(INSDP1,"[Best Unit on Line Time%2d]:",i);
                    for(k=0; k < NOGEN; k++)
                      fprintf(INSDP1,"%2d ",GTD[i][BS[i]][k]);
                    fprintf(INSDP1,"\n");
                    }
            }

           //Fungsi Start-Up Cost untuk masing-masing unit
           void StartUpCostDP(void)
           {
                    int i,k;
                  double startcost;
                  double* CostStart;
                  double startnol;
                  CostStart = new double [WAKTU];

                    startnol = 0;
                  for(k = 0; k < NOGEN; k++)
                    if(GTD[0][BestState[0]][k]==1)
                     startnol = startnol + start[k];

                  CostStart[0] = startnol;

                    for(i=1; i<WAKTU; i++)
                    {
                    startcost=0;
                    for(k = 0; k < NOGEN; k++)
                      {
                      startcost = startcost + (GTD[i][BestState[i]][k]*(1-GTD[i-
1][BestState[i-1]][k])*start[k]);
                      flopDP = flopDP + 2;
```

```
        }
      CostStart[i] = startcost;
      }

   BestStartCost = 0;
   for(i=0; i<WAKTU; i++)
      BestStartCost = BestStartCost + CostStart[i];

   for(i=0; i<WAKTU; i++)
      fprintf(INSDP1,"Best TransCost Time %d = %g\n",i,CostStart[i]);

   fprintf(INSDP1,"\nBest TransCost          = %g\n",BestStartCost);
       fprintf(INSDP1,"Biaya Total Bahan Bakar =
%7.3f\n",FuelCostDP);
       fprintf(INSDP1,"Biaya Total             =
%7.3f\n",FuelCostDP+BestStartCost);

   delete [] CostStart;
}

void SpinReserveDP(void)
{
       int j;

   for(j=0; j<WAKTU; j++)
      SRDP[j]=DayamaX[j][BestState[j]]-PB[j];

   for(j=0; j<WAKTU; j++)
       fprintf(INSDP1,"[Time-%2d],DayaMak = %g, PB = %g, Spinning
Reserve = %g\n",j+1,DAYAMAX[BestState[j]],PB[j],SRDP[j]);
}

void SecurityLevelDP(void)
{
   int i, k;
       double SecLevDP[WAKTU][NOGEN];       //Security level untuk
tiap jam

       for(i=0; i<WAKTU; i++)
       for(k=0; k<2; k++)
         SecLevDP[i][k]=FOR[k];

   for(i=0; i<WAKTU; i++)
       for(k=2; k<NOGEN; k++)
       {
        if(GTD[i][BestState[i]][k])
           SecLevDP[i][k]=1-FOR[k];
        else
                                SecLevDP[i][k]=0;
        }

   for(i=0; i<WAKTU; i++)
   {
   UnSecLevDP[i]=1;
     for(k=0; k<NOGEN; k++)
     if(SecLevDP[i][k])
                {
           UnSecLevDP[i]=UnSecLevDP[i]*SecLevDP[i][k];
           flopDP = flopDP + 1;
           }
   }

   for(i=0; i<WAKTU; i++)
   {
   fprintf(INSDP1,"[Time-%2d]: ",i+1);
     for(k=0; k<NOGEN; k++)
     fprintf(INSDP1,"%g ",SecLevDP[i][k]);
   fprintf(INSDP1,"\n");
   }

     for(i=0; i<WAKTU; i++)
     fprintf(INSDP1,"[Time-%2d] = %g\n",i+1,UnSecLevDP[i]);

}


void main(void)
{
       time_t first, second;

       INSDP1 = fopen("INSDP1.txt","w");

       first = time(NULL);  /* Gets system time */

       fprintf(INSDP1,"\n/*****************************************
*******************/");
       fprintf(INSDP1,"\n/* Program Aplikasi: Optimisasi Unit
Pembangkit Tenaga Listrik*/");
       fprintf(INSDP1,"\n/*
dengan menggunakan Algoritma Genetik    */");
       fprintf(INSDP1,"\n/* Nama file                : UCDP.CPP
*/");
```

```
        fprintf(INSDP1,"\n/* Diprogram untuk : Penelitan Skripsi
*/");
        fprintf(INSDP1,"\n/* Oleh                              : Gigih
Sanjaya S D                          */");
        fprintf(INSDP1,"\n/* NIM                               :
0510630044 - 63
*/");
        fprintf(INSDP1,"\n/*****************************************
*******************/");


flopDP = 0;
InisialDP();
LimPowMinDP();
LimPowMaxDP();
GenOLFeasible();
OptPowerDP();
EvalCostDP();
//BubleSort();
//MinUpDown();
StartUpCostDP();
        /*for(i=0; i<WAKTU; i++)
                {
      //fprintf(INSDP1,"PB[%d]= %g ",i,PB[i]);
      for(j=0; j<GF[i]; j++)
                    {
          fprintf(INSDP1,"[Time-%2d, %3d] PB= %g :",i,j,PB[i]);
          for(k = 0; k < NOGEN; k++)
               fprintf(INSDP1," %g ",PoptDP[i][j][k]);
          fprintf(INSDP1,"\n");
          }
      }*/

SpinReserveDP();
SecurityLevelDP();

fprintf(INSDP1,"=======================================\n");
fprintf(INSDP1,"Jumlah FlopDP = %d\n",flopDP);
fprintf(INSDP1,"=======================================\n");
second = time(NULL); /* Gets system time again */
fprintf(INSDP1,"=======================================\n");
fprintf(INSDP1,"Waktu yang dihabiskan adalah : %10.10f
\n",difftime(second,first));
fprintf(INSDP1,"=======================================\n");

fclose(INSDP1);
```

```
printf("Suksess!!\n");
printf("AlhamduLillah!!\n");
getch();

//Pendealokasian dari nilai Evaluasi
  for(int i = 0; i < WAKTU; i++)
                delete [] EvaL[i];
        delete [] EvaL;

//Pendealokasian dari daya optimum
  for(int i=0; i < WAKTU; i++)
                for(int j=0; j < GF[i]; j++)
                delete PoptDP[i][j];
    for(int i=0; i < WAKTU; i++)
        delete PoptDP[i];

//Pendealokasian dari GTD
  for(int i=0; i < WAKTU; i++)
                for(int ii=0; ii < STATE; ii++)
                delete  GTD[i][ii];
    for(int i=0; i < WAKTU; i++)
        delete GTD[i];

    //Pendealokasian dari Daya Maksimum
    for(int i = 0; i < WAKTU; i++)
                delete [] DayamaX[i];
        delete [] DayamaX;

    //Pendealokasian dari Daya Maksimum
    for(int i = 0; i < WAKTU; i++)
                delete [] DayamiN[i];
        delete [] DayamiN;

}
```