PENGEMBANGAN PERANGKAT LUNAK PENJADWALAN SISTEM MANUFAKTUR DENGAN TEKNIK RELAKSASI LAGRANGIAN

TUGAS AKHIR

Disusun Oleh :

MADE EMI KRISMARINI
NRP. 2696.100.084

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2001

PENGEMBANGAN PERANGKAT LUNAK PENJADWALAN SISTEM MANUFAKTUR DENGAN TEKNIK RELAKSASI LAGRANGIAN

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Ir. ARIF DJUNAIDY, M.Sc., Ph.D.	**RULLY SOELAIMAN, S.Kom.**
NIP. 131 633 403	**NIP. 132 085 802**

SURABAYA
Februari, 2001

ABSTRAK

ABSTRAK

Penjadwalan pada job shop merupakan salah satu permasalahan yang penting dan kompleks. Oleh karena itu, perlu adanya peningkatan dalam penanganannya. Beberapa strategi telah digunakan untuk menemukan metode penjadwalan yang menghasilkan jadwal yang optimal.

Dalam tugas akhir ini dirancang dan diimplementasikan perangkat lunak penjadwalan sistem manufaktur job shop dengan menggunakan teknik relaksasi Lagrangian. Teknik relaksasi Lagrangian merupakan suatu teknik matematika untuk menyelesaikan permasalahan sistem manufaktur job shop yang mempunyai fungsi obyektif meminimumkan keterlambatan penyelesaian suatu pekerjaan dengan batasan jumlah mesin yang tersedia dan urutan operasi–operasinya pada tiap mesin. Proses penyelesaian pertama kali dilakukan pada level yang lebih kecil, yaitu level operasi, kemudian dilanjutkan pada level yang lebih tinggi, yaitu level pekerjaan. Proses terakhir adalah penyusunan jadwal berdasarkan hasil dari proses sebelumnya disesuaikan dengan batasan–batasan yang mempengaruhinya.

Hasil uji coba perbandingan antara teknik relaksasi Lagrangian dengan algoritma pada perangkat lunak sejenis (yaitu perangkat lunak Lekin) menghasilkan jadwal yang lebih optimal. Pengembangan lebih lanjut dapat dilakukan pada perangkat lunak ini berdasarkan jenis data yang akan diolah dan teknik penyelesaian persamaannya.

KATA PENGANTAR

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas anugrah yang diberikan sehingga dapat terselesaikannya Tugas Akhir ini dengan judul :

PENGEMBANGAN PERANGKAT LUNAK

PENJADWALAN SISTEM MANUFAKTUR DENGAN

TEKNIK RELAKSASI LAGRANGIAN

Tugas Akhir ini disusun untuk memenuhi persyaratan meraih gelar Sarjana S-1 pada Jurusan Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember , Surabaya.

Melalui kesempatan ini, penulis menyampaikan terima kasih sebesar – besarnya kepada :

1. Bapak Ir. Arif Djunaidy, M.Sc. Ph.D, selaku Ketua Jurusan Teknik Informatika FTI – ITS dan dosen pembimbing yang telah memberikan arahan dan bimbingan selama pengerjaan tugas akhir ini.
2. Bapak Ir. M. Husni, M.Kom, selaku Sekretaris Jurusan Teknik Informatika FTI – ITS.
3. Bapak Rully Soelaiman, S.Kom, selaku dosen pembimbing yang telah memberikan ide, bimbingan dan bantuan selama pengerjaan tugas akhir ini.
4. Bapak Agus Zainal, S.Kom dan Bapak Dwi Sunaryono, S.Kom selaku dosen wali selama perkuliahan di Jurusan Teknik Informatika ITS.

5. Seluruh staf pengajar Teknik Informatika ITS atas ilmu yang diberikan selama masa perkuliahan.
6. Mas Hamsi, yang telah memberikan bimbingan, pengetahuan, kritik dan saran selama penulis mengerjakan tugas akhir ini.
7. Segenap staf pegawai dan tata usaha Jurusan Teknik Informatika ITS.
8. Papa dan Mama tercinta, kakak dan adik-adikku tersayang Mbak Putu, Indri, Wida, Irma, dan Arya yang selalu memberikan doa, dukungan dan kasih sayang kepada penulis.
9. Mas Anto yang telah memberikan dorongan, semangat, bantuan, kritik dan saran selama ini dan sebelum seminar tugas akhir.
10. Sahabat dan teman – temanku : Astri, Monica, Diana, Dhani, Teguh, Indra, Dayat, Mas Setya, Vira, Goesan, Silvi, Azzah, dan banyak lagi yang telah memberikan semangat walaupun jarak jauh.
11. Saudara – saudaraku, mantan penghuni SS – 24 dan RR – 21 : Aciek, Pay, Ayu, Mbak Yuli, Mbak Jesi, Mbak Warma, Mbak Kasmi dan Novi yang telah memberikan hiburan, dukungan dan semangat selama kuliah di Surabaya ini, semoga kita tetap rukun sampai nanti.
12. Rekan – rekanku C-0C yang tidak dapat disebutkan satu – per satu atas kerja samanya selama ini.
13. Dan pihak – pihak lain yang juga turut mendukung dan memberikan dorongan selama ini.

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR TABEL

DAFTAR SIMBOL

DAFTAR SIMBOL

K : waktu horizon, yaitu waktu yang disediakan untuk melakukan penjadwalan

N : jumlah pekerjaan yang akan dijadwalkan

M_{kh} : jumlah mesin h yang bekerja pada waktu k

w_i : nilai prioritas untuk tiap pekerjaan

c_i : waktu penyelesaian pekerjaan i

d_i : waktu seharusnya dari pekerjaan i

t_{ijh} : waktu yang dibutuhkan oleh pekerjaan i operasi j pada jenis mesin h

S_{ijl} : waktu sela (time out) antara operasi (i, j) dan (i, l).

λ_{ijl} : pengali Lagrange untuk batasan operasi sebelumnya

π_{kh} : pengali Lagrange untuk batasan kapasitas mesin

p_{ijl} : nilai penalti

b_{ij} : waktu awal operasi j pekerjaan i dikerjakan pada suatu mesin.

c_{ij} : waktu di mana tiap pekerjaan i operasi j diselesaikan

T_i : kelambatan (tardiness) dengan nilai ($\max[0, c_i - d_i]$)

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Penjadwalan merupakan salah satu perencanaan yang paling penting pada sistem manufaktur dan hasilnya dapat menghasilkan penghematan biaya produksi. Penjadwalan yang optimal diperlukan agar tidak terjadi hilangnya waktu efektif, kemampuan mesin yang rendah, dan waktu penyelesaian yang tidak dapat diprediksi ataupun dikontrol.

Pada sistem manufaktur, perubahan dinamik merupakan elemen yang tidak dapat dihindari. Adanya pekerjaan baru, mesin-mesin yang rusak bahkan terjadinya perubahan pada rencana proses dari beberapa pekerjaan merupakan perubahan yang dapat mempengaruhi jadwal. Sehingga diperlukan suatu jadwal yang dapat menampung semua perubahan tersebut.

Penjadwalan pada sistem manufaktur dapat dipandang sebagai permasalahan pengambilan keputusan dengan pengoptimalan sebuah fungsi obyektif yang berhubungan dengan batasannya. Karena metode penjadwalan yang optimal tidak tercapai, telah dibuat suatu penelitian yang bertujuan untuk mengembangkan metodologi solusi yang efisien yang dapat menghasilkan solusi yang mendekati optimal, dan mengembangkan

metode untuk membentuk suatu jadwal yang dapat menampung perubahan dinamik tersebut.

Relaksasi lagrangian merupakan salah satu teknik matematika untuk menyelesaikan masalah optimasi dan merupakan metode yang tepat untuk menyelesaikan masalah penjadwalan yang kompleks. Dengan pengurangan batasan kapasitas mesin, penggunaan pengali lagrange, metode pemecahan sebuah masalah menjadi sejumlah sub masalah yang lebih kecil, permasalahan akan lebih mudah dalam penyelesaiannya. Setelah semua sub masalah (level rendah) telah diselesaikan, pengali dapat disesuaikan pada level tinggi di mana batasan kapasitas terpenuhi melalui suatu iterasi. Dengan pengali lagrange dari jadwal sebelumnya, sebuah jadwal dapat dengan efektif dibuat untuk menampung perubahan dinamik. Hal ini dapat dilakukan karena perubahan pekerjaan tidak terlalu drastis dari hari ke hari sehingga nilainya tidak terlalu berbeda.

1.2 PERMASALAHAN

Permasalahan pada Tugas Akhir ini adalah :

1. Bagaimana suatu teknik Relaksasi Lagrangian bisa digunakan untuk memberikan penyelesaian yang mendekati optimal untuk sistem manufaktur khususnya job shop dengan meminimumkan kuadrat nilai bobot kelambatan tiap pekerjaan.
2. Bagaimana suatu jadwal dapat ditentukan nilai kualias optimalnya.

1.3 TUJUAN DAN MANFAAT

Tujuan dari Tugas Akhir ini adalah mengembangkan suatu perangkat lunak untuk memperoleh jadwal yang optimal dan valid dari suatu sistem manufaktur khususnya job shop dengan teknik relaksasi lagrangian.

Adapun manfaat yang dapat diperoleh dari pembuatan tugas akhir ini adalah digunakan sebagai bahan pertimbangan untuk membuat suatu penjadwalan dalam industri manufaktur.

1.4 BATASAN PERMASALAHAN

Tugas akhir ini menitikberatkan pada persoalan penjadwalan sistem manufaktur job shop yang memperhatikan batas waktu untuk pengerjaan suatu pekerjaan sehingga fungsi obyektifnya adalah meminimumkan sejumlah kuadrat prioritas nilai kelambatan tiap pekerjaan tersebut. Persoalan dirumuskan dalam bentuk suatu persamaan lagrange dengan teknik relaksasi lagrangian.

Pengembangan perangkat lunak yang akan dibuat adalah berupa suatu alat untuk menghasilkan jadwal mendekati optimum, bukan pembuatan jadwal dengan studi kasus tertentu.

Batasan permasalahan diuraikan dalam bentuk asumsi-asumsi terhadap pekerjaan ataupun operasi, mesin dan alokasi waktu yang dibutuhkan untuk memproses seluruh pekerjaan. Setiap pekerjaan memiliki waktu yang dibutuhkan untuk melakukan suatu proses, bobot yang

mempengaruhi tingkat kepentingannya, dan batas waktu pengerjaan berupa nilai konstan yang dapat ditentukan. Setiap mesin diasumsikan mempunyai jenis berbeda dan kapasitas tidak sama yang nilainya dapat ditentukan dan hanya dapat mengerjakan satu kali proses pada satu alokasi waktu. Proses yang dikerjakan harus sampai selesai, tidak ada penundaan suatu proses (*non pre-emptive*).

1.5 METODOLOGI PENGERJAAN TUGAS AKHIR

1. Studi kepustakaan

   Pencarian sumber-sumber yang berhubungan dengan pengembangan perangkat lunak ini, berupa buku, jurnal dan literatur yang lain.

2. Perancangan dan pembuatan perangkat lunak

   Solusi dari permasalahan dirancang dan dibuat suatu algoritma yang merupakan kerangka dari perangkat lunak dan akan dikembangkan serta dilanjutkan dengan pembuatan perangkat lunak.

3. Pengujian perangkat lunak

   Perangkat lunak yang telah dikembangkan ini diuji dengan menggunakan data-data pendukung permasalahan.

4. Evaluasi dan modifikasi perangkat lunak

   Evaluasi dan modifikasi untuk mengoptimalkan kerja dari perangkat lunak yang telah dibuat sehingga diperoleh hasil yang baik.

5. Penulisan tugas akhir

   Membuat laporan dalam bentu buku Tugas Akhir.

1.6 SISTEMATIKA PEMBAHASAN

Sistematika yang digunakan dalam Tugas Akhir dijelaskan berikut ini.

- Bab I Pendahuluan, menjelaskan mengenai latar belakang, permasalahan dan batasannya, tujuan dan manfaat, metodologi penelitian dan sistematika penulisan.
- Bab II Penjadwalan Sistem Manufaktur, berupa pembahasan dasar teori penjadwalan secara umum.
- Bab III Penerapan Relaksasi Lagrangian pada Penjadwalan Sistem Manufaktur Job Shop, menjelaskan teknik relaksasi Lagrangian dan pembahasan penyelesaian persoalan sistem manufaktur job shop dengan menggunakan teknik relaksasi Lagrangian.
- Bab IV Perancangan dan Pembuatan Perangkat Lunak, membahas sistem perangkat lunak penjadwalan suatu sistem manufaktur jenis job shop, perancangan dan implementasi struktur data yang digunakan oleh perangkat lunak .
- Bab V Uji Coba dan Evaluasi Perangkat Lunak, membahas hasil uji coba perangkat lunak dan mengevaluasi kemampuannya.
- Bab VI Penutup, menguraikan kesimpulan dari bab-bab yang lain serta kemungkinan pengembangan yang berkaitan dengan perangkat lunak ini.

BAB II

PENJADWALAN SISTEM MANUFAKTUR

BAB II

PENJADWALAN SISTEM MANUFAKTUR

Persoalan penjadwalan timbul pada aktifitas manusia : aktifitas pada suatu proyek, adanya kelas pada suatu universitas, penerbangan, penghasilan suatu produk tertentu dan lain-lain. Penjadwalan ditekankan pada waktu mulai dan selesainya suatu pekerjaan yang dikumpulkan pada suatu himpunan tertentu yang dibatasi oleh beberapa batasan yang mempengaruhi pekerjaan tersebut.

Untuk merealisasikan perencanaan produksi, penjadwalan memegang peranan penting dalam mewujudkan efektifitas dan efisiensi produksi. Semakin kompleks suatu sistem produksi maka dibutuhkan suatu sistem penjadwalan yang baik.

Penjadwalan pada umumnya merupakan suatu permasalahan yang tidak mungkin mendapatkan solusi optimal terjamin dengan menggunakan algortima yang cepat, walaupun beberapa permasalahan dapat dipecahkan dengan optimal dengan menggunakan algoritma yang cepat. Saat membangun suatu skema perkiraan yang spesifik, beberapa metode dan teknik dapat digunakan bersamaan saperti pendekatan secara hirarki dan iteratif, metode polinomial, metode relaksasi, metode konstraktif, teknik peningkatan pencarian lokal, algoritma genetik dan lain-lain.

Permasalahan penjadwalan ini berkembang semakin kompleks dan penyelesaiannya masih belum dikatakan optimal. Beberapa metode telah dikembangkan untuk penyelesaian yang mendekati optimal. Metode penjadwalan yang baik akan dapat memberikan petunjuk dalam pelaksanaan produksi. Sistem penjadwalan yang dibuat akan memutuskan bagaimana urutan pekerjaan-pekerjaan yang akan dikerjakan pada setiap mesin dan memberikan informasi kapan seharusnya masing-masing pekerjaan dimulai dan berakhir serta kapan seharusnya semua pekerjaan selesai dikerjakan.

2.1 ATURAN PENJADWALAN

Sebelum melakukan penjadwalan, informasi yang diperlukan untuk mendefinisikan suatu permasalahan penjadwalan adalah jumlah pekerjaan yang harus dijadwalkan, jumlah mesin yang tersedia, urutan proses masing-masing pekerjaan, jenis fasilitas manufakturing, dan kriteria tujuan penjadwalan yang ingin dicapai (fungsi obyektifnya).

Obyek yang dijadwalkan adalah suatu produk tersendiri. Sebuah produk dapat merupakan suatu unit tunggal atau kumpulan unit yang akan diproses. Untuk membuat produk yang diberikan, maka diperlukan suatu mesin. Untuk mendapatkan produk terakhir, suatu himpunan operasi harus dibentuk pada satu atau beberapa unit. Persoalan-persoalan yang timbul pada penjadwalan adalah sebagai berikut :

- Adanya batas waktu dari suatu produk, jika suatu produk selesai sebelum target ataupun disebut terlambat jika selesai melebihi waktu yang telah ditentukan.
- Waktu mulai untuk pertama kali suatu proses produksi.
- Jalur yang dapat digunakan untuk memperoleh hasil dari produk dengan :
    i. Berhubungan dengan waktu proses
    ii. Berhubungan dengan mesin yang digunakan untuk tiap operasi pada tiap jalur
    iii. Himpunan dari batasan operasi sebelumnya pada tiap jalur.

Penjadwalan dititikberatkan pada penempatan sejumlah sumber dengan persoalan tertentu pada suatu waktu. Hal ini merupakan proses pengambilan keputusan yang mempunyai satu atau lebih tujuan. Sumber itu bisa berupa mesin-mesin pada suatu workshop, landasan pesawat terbang pada bandara, pekerja pada perusahaan konstruksi, dan lain-lain. Persoalannya antara lain operasi pada proses produksi, penerbangan dan pendaratan pada bandara, tahapan pada penyusunan proyek, dan lain-lain. Tiap persoalan mempunyai tingkatan prioritas yang berbeda, waktu mulai pekerjaan yang paling awal dan batas waktu. Bentuk tujuannya bisa bermacam-macam, meminimumkan waktu penyelesaian, meminimumkan sejumlah persoalan yang selesai setelah batas waktu, dan lain-lain.

Penjadwalan merupakan proses pembuatan keputusan yang paling banyak terjadi pada manufakturing dan sistem produksi serta terjadi pada sistem transportasi dan distribusi pada jenis industri pelayanan.

Pada penjelasan berikut ini, sebuah diskripsi diberikan pada lingkungan manufakturing umum dan aturan dari sebuah proses penjadwalan dalam lingkungan tertentu.

Pada sistem manufaktur, pesanan dimulai dan diwujudkan menjadi pekerjaan dengan dihubungkan batas waktu. Pekerjaan diproses oleh mesin dalam urutan yang diberikan. Pekerjaan-pekerjaan menunggu untuk diproses pada mesin yang bekerja, dan penundaan mungkin terjadi ketika pekerjaan dengan tingkat yang lebih tinggi datang pada mesin tersebut dan harus diproses di sana. Penjadwalan secara rinci dari suatu persoalan yang akan dibentuk pada suatu sistem produksi diperlukan untuk memelihara efisiensi dan mengontrol operasi.

Fungsi penjadwalan harus dihadapkan dengan beberapa fungsi penting yang lain pada suatu organisasi. Hal ini dipengaruhi oleh proses perencanaan produksi, yang memegang rencana jangka pendek sampai jangka panjang untuk seluruh organisasi. Proses ini mempertimbangkan tingkat inventaris, peramalan dan kebutuhan sumber untuk mengoptimasi dari hasil campuran dan jangka panjang dari penempatan sumber. Keputusan yang diambil oleh fungsi perencanaan ini mempunyai pengaruh pada penjadwalan. Penjadwalan juga menerima input dari kontrol shopfloor. Kejadian yang tidak diperkirakan pada shopfloor seperti

kerusakan mesin atau waktu proses lebih lama dari yang diperkirakan, harus diperhitungkan karena mempunyai pengaruh utama pada penjadwalan.

Pabrik modern saat ini mempunyai sistem informasi manufaktur yang rumit. Sebuah pabrik minimum mempunyai sebuah pusat komputer dan pusat basis data. Hubungan dengan pusat komputer ini adalah LAN pada PC, workstation, terminal input data yang mungkin digunakan untuk mendapatkan kembali dari basis data atau memasukkan data baru. Fungsi dari penjadwalan biasanya dilakukan pada sebuah komputer PC atau workstation yang dihubungkan dengan komputer utama dari pabrik. Terminal pada lokasi kunci dihubungkan dengan komputer penjadwalan yang mengijinkan departemen mengakses informasi penjadwalan yang diperlukan dan memungkinkan departemen untuk menyediakan sistem penjadwalan dengan informasi yang berhubungan seperti status mesin, perubahan pada data pekerjaan dan lain-lain.

2.2 MODEL DETERMINISTIK

Pada empat dekade yang lalu, riset secara teori telah dilakukan pada bidang penjadwalan yang deterministik. Jumlah dan variasi dari model yang berbeda sangat mengejutkan. Bagian pertama pada sub bab ini menjelaskan sebuah versi perubahan pada notasi. Bagian kedua menjelaskan beberapa kelas dari penjadwalan. Suatu kelas dari penjadwalan dikategorikan secara khusus pada proses pembuatan keputusan.

2.2.1 KERANGKA DAN NOTASI

Pada semua permasalahan penjadwalan, sejumlah pekerjaan dan mesin diasumsikan terbatas. Sejumlah pekerjaan dinyatakan dengan *i* dan sejumlah mesin dinyatakan dengan *h*. Jika pekerjaan membutuhkan sejumlah langkah proses atau operasi, maka pasangan (h, i) menyatakan operasi dari pekerjaan *i* pada mesin *h*. Berikut ini penjelasan data yang dihubungkan dengan pekerjaan i [Pin-95].

Waktu proses (processing time (p_{hi})). P_{hi} menyatakan waktu proses dari pekerjaan *i* pada mesin *h*. Indeks *h* dihilangkan jika waktu proses dari pekerjaan *i* tidak tergantung pada mesin atau pekerjaan *i* hanya diproses pada satu mesin yang diberikan.

Waktu datang (release date (r_i)). Waktu datang dari pekerjaan *i* mungkin dihubungkan dengan waktu kesiapan (ready date). Hal ini merupakan waktu kedatangan pekerjaan pada sistem, di mana waktu paling awal pada tiap pekerjaan *i* mulai dapat diproses.

Waktu seharusnya (due date (d_i)). Waktu seharusnya d_i dari pekerjaan *i* menyatakan waktu penyelesaian (waktu di mana pekerjaan telah dijanjikan kepada konsumen). Penyelesaian pekerjaan setelah waktu seharusnya diperbolehkan, tetapi dikenakan suatu penalti. Ketika waktu seharusnya harus dipenuhi, hal ini mengacu pada suatu batas waktu.

Prioritas (weight (w_i)). Prioritas w_i dari pekerjaan *i* pada dasarnya merupakan suatu faktor prioritas, menunjukkan tingkat kepentingan pekerjaan *i* terhadap pekerjaan lainnya pada suatu sistem. Nilai ini bisa

berupa nilai inventaris; mungkin juga sejumlah nilai yang telah tersedia ditambahkan pada pekerjaan.

Persoalan penjadwalan dinyatakan dengan sebuah $\alpha \mid \beta \mid \gamma$, α menyatakan lingkungan dari mesin dan berisi sebuah masukan. β menyediakan detail dari karakteristik proses dan batasan-batasan yang mungkin tidak memiliki masukan, sebuah masukan atau beberapa masukan. γ berisi tujuan untuk meminimumkan dan biasanya berisi sebuah masukan.

Berikut ini merupakan contoh lingkungan mesin yang mungkin pada α.

Mesin tunggal (1). Kasus pada mesin tunggal adalah yang paling sederhana dari semua lingkungan mesin dan merupakan kasus yang khusus dari lingkungan mesin yang rumit lainnya.

Mesin sejenis yang paralel (P_m). Ada *m* mesin sejenis yang paralel. Pekerjaan *i* membutuhkan sebuah operator dan mungkin diproses pada salah satu dari *m* mesin atau salah satu dari subset yang diberikan.

Flow shop (F_m). Ada sejumlah mesin *m* yang seri. Tiap pekerjaan harus diproses pada salah satu dari mesin *m*. Semua pekerjaan mempunyai jalur yang sama, yaitu mereka harus diproses pertama kali pada mesin 1, kemudian mesin 2 dan seterusnya. Setelah menyelesaikan pada satu mesin, sebuah pekerjaan tersebut bergabung dengan antrian pada mesin selanjutnya. Biasanya, semua antrian diasumsikan dikerjakan secara FIFO (first in first out), yaitu sebuah pekerjaan tidak dapat

melewati yang lain selama proses menunggu pada antrian. Dengan adanya FIFO, flow shop merupakan suatu permutasi dan β termasuk masukan prmu.

Flexible flow shop (FFs). Sebuah flexible flow shop adalah bentuk umum dari flow shop dan lingkungan mesin paralel. Sebagai pengganti *m* mesin seri ada stage seri dengan sejumlah mesin paralel pada tiap stage. Tiap pekerjaan harus diproses pada pertama kali pada stage 1, kemudian stage 2 dan seterusnya. Fungsi dari stage sebagai tempat dari mesin paralel; tiap stage pekerjaan *i* hanya membutuhkan satu mesin dan biasanya beberapa mesin dapat memproses beberapa pekerjaan. Antrian antara beberapa stage biasanya dikerjakan secara FIFO.

Open shop (O_m). Ada sejumlah *m* mesin, tiap pekerjaan harus diproses lagi pada salah satu dari mesin *m*. Pada jenis ini tidak ada pembatasan jalur tiap pekerjaan pada jenis mesin. Jadwal mengijinkan untuk menentukan jalur tiap pekerjaan dan pekerjaan yang berbeda mungkin mempunyai jalur yang berbeda.

Job shop (J_m). Pada job shop dengan *m* mesin, tiap pekerjaan mempunyai jalur tersendiri yang akan diikutinya. Perbedaannya dengan open shop, di mana tiap pekerjaan mengunjungi beberapa mesin hanya sekali dan job shop mungkin mengunjungi mesin lebih dari sekali.

Batasan proses dinyatakan dalam β terdiri dari beberapa masukan. Masukan yang mungkin pada β adalah :

Waktu datang (release date (r_i)). Jika simbol ini dinyatakan dalam β, maka pekerjaan *i* tidak dapat dimulai prosesnya sebelum waktu awal r_i. Jika r_i tidak ada pada β maka proses pada pekerjaan *i* dapat dimulai kapan saja. Berbeda dengan waktu awal, waktu seharusnya tidak dinyatakan dalam bagian ini. Jenis fungsi obyektifnya memberikan indikasi yang cukup walaupun ada waktu seharusnya ataupun tidak.

Urutan yang tergantung dengan waktu persiapan (s_{il}). S_{il} menyatakan urutan yang tergantung dengan waktu persiapan antara pekerjaan *i* dan *l*; s_{0l} menyatakan waktu persiapan untuk pekerjaan *l* jika pekerjaan *l* adalah pertama pada urutan dan s_{i0} waktu akhir dari pekerjaan *i* jika pekerjaan *i* adalah terakhir pada urutan (tentu saja, nilai s_{0l} dan s_{i0} adalah 0). Jika waktu persiapan antara pekerjaan *i* dan *l* tergantung pada mesin, maka indeks *h* diikutkan, menjadi s_{hil}. Jika tidak ada s_{hil} pada β maka seluruh waktu persiapan diasumsikan nol.

Penundaan (preemption (prmp)). Penundaan menyatakan bahwa tidak penting untuk mempertahankan sebuah pekerjaan pada suatu mesin sampai selesai. Jadwal diijinkan untuk memotong proses dari sebuah pekerjaan pada waktu tertentu dan menempatkan pekerjaan yang berbeda pada mesin tersebut. Ketika sebuah pekerjaan yang tertunda tersebut dikembalikan pada mesin tersebut (atau mesin yang lain, pada kasus mesin paralel), hanya membutuhkan mesin untuk mengingat waktu proses. Jika penundaan diijinkan maka prmp dimasukkan dalam β, jika tidak dimasukkan maka penundaan tidak diijinkan.

Kemacetan (breakdown (brkdwn)). Mesin macet menyatakan bahwa mesin tidak tersedia terus-menerus. Periode di mana mesin tidak tersedia, diasumsikan tetap (misalnya pemeliharaan dijadwalkan). Jika ada sejumlah mesin sejenis yang paralel, jumlah mesin yang tersedia pada suatu waktu pada sebuah fungsi waktu yaitu m(t).

Batasan syarat pada mesin (M_i). M_i ada pada β jika jenis mesin adalah mesin paralel. Jika terdapat M_i, maka tidak semua mesin *h* dapat memproses pekerjaan *i*. himpunan dari M_i menyatakan himpunan mesin yang dapat memproses pekerjaan *i*. jika M_i tidak ada pada β maka pekerjaan *i* dapat diproses pada salah satu dari *m* mesin tersebut.

Permutasi (prmu). Sebuah batasan yang muncul pada lingkungan flow shop di mana antrian di depan tiap mesin berdasarkan FIFO.

Penghalangan (block). Penghalangan adalah fenomena yang mungkin terjadi pada flow shop. Jika suatu flow shop mempunyai penyangga yang terbatas di antara 2 mesin yang berurutan, hal ini mungkin terjadi jika penyangga penuh, mesin tidak diperbolehkan untuk memulai sebuah pekerjaan. Fenomena ini dikenal sebagai penghalangan; pekerjaan yang telah menyelesaikan prosesnya pada mesin yang telah diberikan tidak dapat meninggalkan mesin tersebut karena pekerjaan sebelumnya yang berada pada mesin selanjutnya tersebut belum selesai dikerjakan.

Tidak ada waktu menunggu (no wait (nwt)). Nwt merupakan fenomena lain yang mungkin terjadi pada flow shop. Pekerjaan tidak

diijinkan untuk menunggu di antara dua mesin yang berurutan. Hal ini menyatakan bahwa waktu mulai dari suatu pekerjaan pada mesin pertama harus ditunda untuk meyakinkan bahwa pekerjaan dapat melalui flow shop tanpa harus menunggu mesin yang lain. Sangat jelas bahwa nwt suatu mesin juga mengoperasikan dalam bentuk FIFO.

Perputaran (recirculation (recrc)). Suatu perputaran mungkin terjadi pada job shop ketika suatu pekerjaan mengunjungi mesin lebih dari satu kali.

Masukan yang lain pada β dapat berupa penjelasan tersendiri. Misalnya p_i=p menyatakan bahwa semua waktu proses adalah sama, d_i = d menyatakan bahwa waktu sebenarnya semua pekerjaan adalah sama.

Suatu obyektif / tujuan untuk meminimumkan selalu berupa sebuah fungsi dari waktu penyelesaian dari pekerjaan yang tergantung dengan jadwal. Waktu penyelesaian pekerjaan *i* pada mesin *h* dinyatakan dengan C_{hi}. Waktu di mana pekerjaan *i* masih berada pada sistem (misalnya waktu penyelesaiannya pada mesin terakhir yang dibutuhkannya) dinyatakan dengan C_i. Suatu fungsi obyektif juga bisa berupa fungsi dari waktu sebenarnya d_i.

Keterlambatan (lateness) dari suatu pekerjaan *i* dapat dinyatakan dengan :

$$L_i = C_i - d_i$$

Dengan nilai positif jika pekerjaan i terlambat penyelesaiannya dan negatif jika sebaliknya.

Kelambatan (tardiness) suatu pekerjaan j dinyatakan dengan :

$T_i = \max (C_i - d_i, 0) = \max (L_i, 0)$

Perbedaan antar L_i dan T_i terletak pada nilai T_i tidak pernah negatif.

Unit penalti dari pekerjaan i dinyatakan dengan :

$U_i = 1$ jika $C_i > d_i$

$= 0$ jika sebaliknya

Keterlambatan, kelambatan dan unit penalti merupakan tiga fungsi dasar dari waktu sebenarnya yang berhubungan dengan penalti.

Makespan (C_{max}). Makespan menyatakan sebagai max $(C_1,...,C_n)$, yang sama dengan waktu penyelesaian dari pekerjaan terakhir yang meninggalkan sistem. Meminimumkan makespan biasanya menyatakan sebuah kemampuan dari suatu mesin.

Maksimum keterlambatan (L_{max}). Maksimum keterlambatan, L_{max}, dinyatakan sebagai max $(L_1,...,L_n)$. Hal ini mengukur pelanggaran waktu sebenarnya yang paling buruk.

Jumlah prioritas waktu penyelesaian ($\sum w_i C_i$). Jumlah prioritas waktu penyelesaian dari i pekerjaan memberikan sebuah indikasi dari jumlah inventaris, nilai jadwal. Jumlah waktu penyelesaian pada suatu literatur selalu mengacu sebagai waktu aliran. Jumlah prioritas waktu penyelesaian kemudian lebih dikenal dengan prioritas waktu aliran

Jumlah prioritas kelambatan ($\sum w_i T_i$). Bentuk ini merupakan bentuk yang lebih umum daripada jumlah prioritas waktu penyelesaian.

Prioritas jumlah pekerjaan yang terlambat ($\sum w_i U_i$). Prioritas dari pekerjaan yang terlambat tidak hanya suatu ukuran dari minat akademik, tapi sering merupakan suatu tujuan pada prakteknya sebagai ukuran yang dapat dicatat dengan sangat mudah.

2.2.2 PENGGOLONGAN JADWAL [Pin-95]

Pada istilah penjadwalan, perbedaan selalu dibuat antara suatu urutan, dan penjadwalan. Urutan selalu berhubungan dengan permutasi dari himpunan pekerjaan atau urutan di mana pekerjaan diproses pada mesin yang telah diberikan. Penjadwalan biasanya lebih mengarah pada penempatan pekerjaan dengan susunan mesin yang lebih rumit, yang mengijinkan penundaan pekerjaan oleh pekerjaan lain yang dimulai setelah waktu itu. Asumsi telah dibuat sebagai pedoman suatu jadwal, apa yang harus dilakukan dan apa yang tidak boleh dilakukan.

Definisi 1, suatu jadwal yang fisibel disebut non delay jika tidak ada mesin dalam keadaan diam (idle) ketika ada suatu operasi tersedia untuk melakukan proses. Kebutuhan di mana suatu jadwal menjadi non delay adalah sama dengan pencegahan adanya waktu diam. Untuk sebagian besar model, termasuk semua model yang mengijinkan adanya penundaan, ada jadwal yang optimal yang merupakan non delay.

Contoh berikut ini merupakan ilustrasi kenyataan bahwa mungkin terjadi pada anomali yang tidak disangka sama sekali.

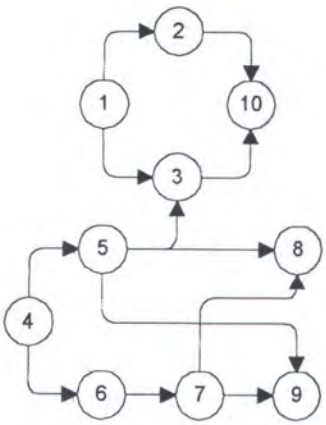
Terdapat 10 pekerjaan dan waktu proses berikut ini :

pekerjaan	1	2	3	4	5	6	7	8	9	10
p_i	8	7	7	2	3	2	2	8	8	15

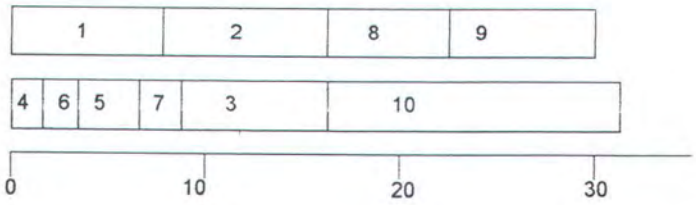
Pekerjaan disusun dengan memperhatikan batasan pekerjaan sebelumnya seperti pada gambar 2-1. Makespan untuk jadwal yang non delay gambar 2-2(a) adalah 31 dan jadwal tersusun dengan optimal.

Jika satu dari 10 waktu proses tersebut dikurangi satu unit waktu, makespan seharusnya kurang dari 31. Akan tetapi pada jadwal non delay hasil yang diperoleh seperti gambar 2-2 (b) dengan sebuah makespan 32.

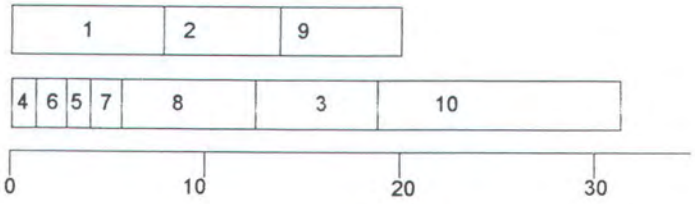
Jika ada penambahan jumlah mesin, hal yang diharapkan adalah menghasilkan suatu makespan kurang dari 31, asumsi waktu proses sebenarnya. Ternyata, hasil dari jadwal non delay tidak sesuai harapan, makespan menjadi 36 (gambar 2-3 (c)).



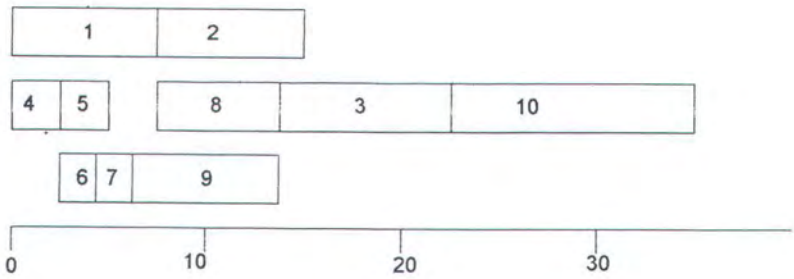
Gambar 2-1 Graph untuk batasan operasi sebelumnya



(a)



(b)



(c)

Gambar 2-2 Gant chart untuk jadwal non delay

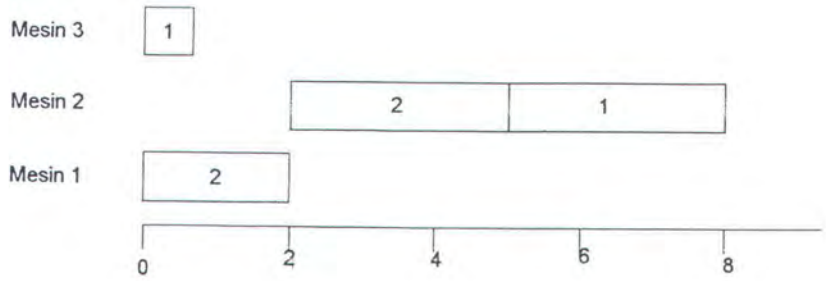
(a) Jadwal yang asli **(b)** Waktu proses dengan pengurangan 1 unit waktu

(c) Waktu proses dengan penambahan 1 mesin

Definisi 2, suatu jadwal yang fisibel disebut aktif jika tidak ada operasi dapat diselesaikan lebih dahulu dengan mengubah urutan proses pada mesin dan tidak ada penundaan pada operasi tersebut. Jadwal aktif terbentuk karena proses left shift yang mengijinkan sebuah operasi melompati operasi lainnya menuju interval waktu idle jika interval waktu tersebut cukup untuk menampung operasi yang bersangkutan

Suatu jadwal non delay merupakan jadwal aktif tetapi tidak bisa sebaliknya. Contoh berikut ini menjelaskan jadwal yang aktf tapi bukan jadwal yang non delay.

Sebuah job shop dengan 3 mesin dan 2 pekerjaan. Pekerjaan 1 membutuhkan 1 unit waktu dari proses pada mesin 1 dan 3 pada mesin 2. Pekerjaan 2 membutuhkan 2 unit waktu pada mesin 3 dan pada 3 pada mesin 2. Kedua pekerjaan tersebut diproses terakhir kali pada mesin 2. Dinyatakan, bahwa jadwal yang memproses pekerjaan 2 pada mesin 2 sebelum pekerjaan 1 (gambar 2.3). Adalah jelas bahwa jadwal ini aktif; dengan membalik urutan dari dua pekerjaan pada mesin 2 menunda proses pekerjaan 2. Bagaimanapun, jadwal bukan jadwal non delay. Mesin 2 tetap diam sampai waktu 2, karena ada pekerjaan yang diproses sejak waktu 1.



Gambar 2-3 Jadwal aktif yang bukan non delay.

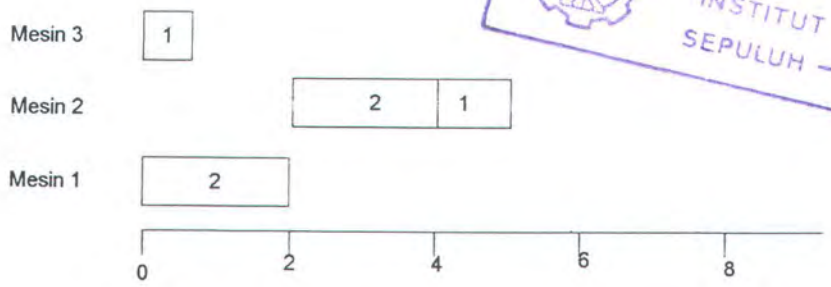
Golongan jadwal yang lebih besar dijelaskan berikut ini.

Definisi 3, suatu jadwal yang fisibel disebut semiaktif jika tidak ada operasi dapat diselesaikan lebih awal tanpa mengubah urutan proses pada beberapa mesin.

Adalah jelas, bahwa jadwal aktif merupakan semi aktif, tapi tidak berlaku sebaliknya.

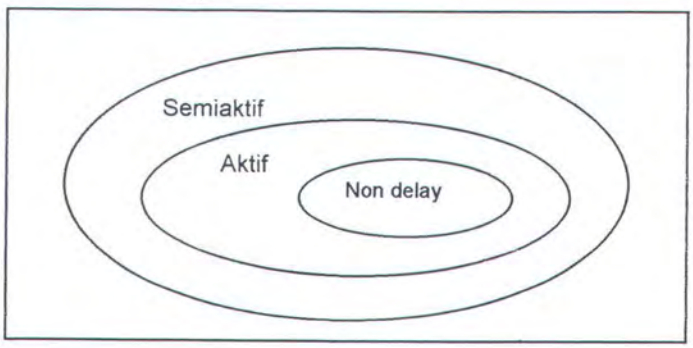
Contoh, suatu job shop dengan 3 mesin dan 2 pekerjaan. Jalur dari 2 pekerjaan tersebut sama dengan contoh sebelumnya. Waktu proses dari pekerjaan 1 pada mesin 1 dan mesin 2 adalah 1. Waktu proses dari pekerjaan 2 pada mesin 2 dan mesin 3 adalah 2. Pada jadwal, pekerjaan 2 diproses pada mesin 2 sebelum mesin 1 (gambar 2.4). Pekerjaan 2 memulai prosesnya pada mesin 2 pada waktu 2 dan pekerjaan 1 memulai prosesnya pada mesin 2 pada waktu 4. Jadwal ini merupakan semiaktif. Akan tetapi, jadwal ini bukan jadwal aktif, di mana pekerjaan 1 dapat diproses pada mesin 2 tanpa menunda proses pekerjaan 2 di mesin 2.

Contoh dari jadwal yang bukan semiaktif dapat disusun dengan mudah. Penundaan proses awal dari pekerjaan 1 pada mesin 2 untuk satu unit waktu, di mana mesin 2 tetap diam untuk satu unit dari waktu antara proses pekerjaan 2 dan 1.



Gambar 2-4 Jadwal semiaktif yang bukan aktif

Secara umum, hubungan antara jadwal semiaktif, aktif dan non delay dapat digambarkan sebagai berikut [Gen-97] :



Gambar 2-5 Hubungan antar jadwal

BAB III

PENERAPAN RELAKSASI LAGRANGIAN PADA PENJADWALAN SISTEM MANUFAKTOR JOB SHOP

BAB III

PENERAPAN RELAKSASI LAGRANGIAN PADA PENJADWALAN SISTEM MANUFAKTUR JOB SHOP

Relaksasi Lagrangian merupakan metode untuk mengoptimasi suatu permasalahan pemecahan/pemisahan pada non linear programming ataupun integer programming. Ide pokok dari pendekatan ini adalah suatu *"nutshell"*, yaitu dekomposisi dan koordinasi, di mana dekomposisi berdasarkan model pemisahan/pemecahan dan koordinasi berdasarkan konsep pembaruan suatu nilai pengali lagrange. Pada bab ini akan dijelaskan lebih rinci tentang metode relaksasi lagrangian dan penerapannya pada sistem manufaktur job shop. ...

3.1 RELAKSASI LAGRANGIAN [Zha-99]

Pada metode ini, suatu batasan direlaksasi melalui pengali lagrange, dan permasalahan relaksasi dapat didekomposisi menjadi sub permasalahan yang lebih kecil. Dengan adanya pengali lagrange, sub permasalahan ini dapat lebih mudah diselesaikan dan diminimumkan. Pengali lagrange kemudian secara iteratif disesuaikan berdasarkan level dari batasan yang mempengaruhinya. Solusi dari permasalahan dilakukan melalui 2 tingkat pendekatan secara iteratif, di mana level rendah

merupakan pemecahan tiap-tiap sub permasalahan dan koordinasi dari solusi sub permasalahan dibentuk melalui pembaruan dari pengali lagrange pada tingkat tinggi.

Pada optimasi dengan fungsi primal yang diminimumkan, maka suatu fungsi dual dimaksimumkan. Pada proses ini, solusi sub permasalahan akan cenderung pada suatu solusi yang fisibel (optimal dan valid) selama fungsi dual menghasilkan nilai yang merupakan batas bawah dari fungsi primal.

Penerapan relaksasi lagrangian pada integer programming, metode sub gradien digunakan untuk memaksimumkan fungsi dual jika fungsi dual tersebut merupakan nondiferensial. Metode sub gradien ini umum digunakan.

3.1.1 PERMASALAHAN INTEGER PROGRAMMING

Suatu permasalahan integer programming dapat dinyatakan dalam :

$$\min_{x^l \le x \le x^u} \sum_{i=1}^{I} J_i(x_i) \qquad (3\text{-}1)$$

dengan batasan $Ax \le b \text{ dan } x_i \in Z^{n_i}, i = 1,\ldots,I$

Dalam hal ini, $x = [x_1, x_2,..,x_I]^T$ adalah suatu variabel nx1 dengan $n = \sum_{i=1}^{I} n_i, x^l \text{ dan } x^u$ adalah batas bawah dan batas atas dari x, dan Z merupakan himpunan integer. Matrik A (mxn) adalah dibentuk dari $[a_1, a_2,..a_I]$ di mana a_i adalah matrik mxn_i, b merupakan vektor mx1 dan $\{ J_i(x_i) \}$ adalah fungsi non linear.

3.1.2 TEKNIK RELAKSASI LAGRANGIAN

Adanya m batasan dari $Ax \leq b$, menyebabkan (3-1) sulit untuk diselesaikan. Relaksasi lagrangian dari (3-1) tersebut diberikan oleh :

$$L(\lambda) \min_{x^l \leq x \leq x^u \in Z^n} [\sum_{i=1}^{I} J_i(x_i) + \lambda^T (Ax - b)] \qquad (3\text{-}2)$$

λ adalah sebuah matrik mx1 dari pengali lagrange dan fungsi $L(\lambda)$ adalah dual lagrangian. Jika varibel x_i dipisah melalui pengali lagrange λ, persamaan tersebut dapat ditulis dalam bentuk tiap-tiap sub permasalahan

$$L_i(\lambda) = \min_{x^I \leq x \leq x^u \in Z^n} \left[\sum_{i=1}^{I} J_i(x_i) + \lambda^T (a_i x_i) \right] \qquad (3\text{-}3)$$

Dan

$$L(\lambda) = \sum_{i=1}^{I} L_i(\lambda) - b^T \lambda \qquad (3\text{-}4)$$

Minimisasi $L_i(\lambda)$ dapat lebih mudah diselesaikan daripada (3-1) karena tiap sub permasalahan dapat diselesaikan secara independen. Permasalahan dual lagrangian adalah :

$$(\text{LD}) \ \max_{\lambda > 0} L(\lambda), \qquad (3\text{-}5)$$

dengan solusi optimal dinyatakan dengan $L^* = L(\lambda^*)$.

3.1.3 MEMAKSIMUMKAN FUNGSI DUAL

Jika (3-5) merupakan non diferensial, maka ada beberapa metode untuk memaksimumkan fungsi dual [Wan-97] yaitu :

- Metode Subgradient

  Metode ini merupakan metode yang umum digunakan untuk memperbarui pengali lagrange (dalam hal ini memaksimumkan fungsi dual) karena teknik ini sederhana dan waktu komputasi yang sedikit.

- Metode Interleaved Subgradient

  Penyelesaian iteratif dari permasalahan dual membutuhkan fungsi dual untuk mengevaluasi menyelesaikan semua sub permasalahan (disebut satu iterasi) beberapa kali. Cara seperti ini dapat menghabiskan waktu yang sangat banyak untuk permasalahan yang sangat besar. Metode ini kebalikan dari metode di atas, yaitu memperbarui pengali setelah menyelesaikan tiap sub permasalahan.

- Metode Surrogate Subgradient

  Pada metode ini, perkiraan optimasi dari sebuah sub permasalahan diperlukan untuk mendapatkan arah subgradien. Metode interleaved subgradient merupakan kasus yang khusus dari surrogate.

- Metode Facet Ascending

  Fungsi dual lagrangian terbuat dari banyak facet. Metode ini merupakan suatu algoritma untuk menemukan suatu interseksi dari facet-facet yang berdekatan. Facet tersebut kemudian diproyeksikan pada interseksi untuk memperoleh arah yang naik, dan teknik pencarian garis digunakan untuk menentukan sejauh mana perpindahan sepanjang arah tersebut. Kelebihan metode ini adalah tidak adanya suatu hasil yang zigzag seperti pada metode subgradient.

Kekurangannya adalah jika permasalahan cukup besar maka suatu interseksi akan dibentuk oleh banyak facet, di mana pembuatan suatu interseksi akan sulit dan membutuhkan waktu komputasi yang sangat besar.

- Metode Bundle

Metode ini mengakumulasi dan memanfaatkan titik subgradien dan sekelilingnya pada iterasi saat itu untuk menemukan g^n arah naik (sepanjang nilai fungsi dapat meningkat paling sedikit g^n). Untuk mendapatkan suatu arah atau mendeteksi g^n optimal, membutuhkan penyelesaian sejumlah permasalahan kuadratik programming. Sepanjang arah naik dari g^n, teknik pencarian garis digunakan untuk menentukan ukuran per langkah untuk memperbarui pengali. Sama seperti metode facet ascending, jumlah yang besar pada fungsi dual dalam mengakumulasi subgradien dan untuk membentuk pencarian garis adalah membutuhkan waktu yang banyak.

Penjelasan yang lebih terinci pada metode subgradient adalah sebagai berikut.

Pengali diperbarui dengan :

$$\lambda^{(k+1)} = \lambda^k = s^k g^k \qquad (3\text{-}6)$$

$g^k = g(\lambda^k)$ merupakan subgradien dari $L(\lambda)$ pada saat λ^k, dan dapat dinyatakan dengan :

$$g(\lambda^k) = Ax(\lambda^k) - b = \sum_{i=1}^{I} a_i x_i(\lambda^k) - b \qquad (3\text{-}7)$$

dengan

$$x_i(\lambda^k) = \min_{x^l \le x \le x^u \in Z^n} [J_i(x_i) + (\lambda^k)^T (a_i x_i)] \qquad (3\text{-}8)$$

dengan ukuran langkah s^k adalah

$$0 < s^k < \frac{2(L^* - L^k)}{|g^k|^2} \qquad (3\text{-}9)$$

Nilai dari $L^* - L^k$ terletak antara $0 \le (L^* - L^k) \le (\lambda^* - \lambda^k)^T g(\lambda^k)$

3.2 PENERAPAN PADA SISTEM MANUFAKTUR JOB SHOP

Pada penjadwalan sistem manufaktur job shop ini, ada beberapa keterangan yang menjelaskan batasan-batasan dalam prosesnya, antara lain :

Keterangan untuk pekerjaan :

- Sebuah pekerjaan datang pada suatu sistem untuk dikerjakan pertama kali di suatu mesin pada operasi pertamanya..
- Karakteristik untuk tiap pekerjaan tidak dipengaruhi oleh pekerjaan yang lain.
- Tiap pekerjaan mempunyai urutan mesin yang spesifik yang seharusnya dikunjungi.
- Tiap pekerjaan membutuhkan waktu proses untuk tiap operasinya. (Pada beberapa hal dapat juga diasumsikan bahwa waktu proses dapat ditentukan sebelum pekerjaan diproses).

Keterangan untuk mesin :

- Tiap pusat mesin terdiri dari satu mesin dan tiap mesin tidak dipengaruhi oleh mesin yang lain.
- Tiap mesin selalu tersedia untuk proses suatu pekerjaan, dan tidak ada interupsi seperti kerusakan, perawatan ataupun kejadian yang lain.

Aturan operasi :

1. tiap pekerjaan dipertimbangkan sebagai wujud yang tidak dapat dibagi, walaupun disusun dari beberapa unit (operasi).
2. Tiap operasi dikerjakan untuk diselesaikan sampai akhir, tidak ada penundaan atau interupsi.
3. Tiap operasi untuk suatu pekerjaan, pertama kali dimulai pada satu mesin, diselesaikan sebelum operasi yang lain mulai dikerjakan pada mesin tersebut
4. Tiap operasi diselesaikan tidak lebih dari satu mesin pada suatu waktu tertentu.
5. Tiap pusat mesin disediakan dengan waktu tunggu yang cukup untuk membiarkan suatu operasi menunggu sebelum memulai prosesnya
6. Tiap pusat mesin disediakan dengan ruang output yang cukup untuk mengijinkan operasi yang telah selesai menunggu sampai mereka keluar dari pusat mesin

3.2.1 RUMUSAN PERMASALAHAN

Suatu rumusan integer programming merupakan cara yang umum untuk merepresentasikan suatu permasalahan penjadwalan. Nilai yang harus diminimumkan adalah J yang merupakan jumlah dari prioritas

kuadrat nilai kelambatan (tardiness) tiap pekerjaan, dengan indeks i merupakan pekerjaan ke - i.

$$J \equiv \sum_i w_i T_i^2 \qquad (3\text{-}10)$$

Tardiness merupakan waktu yang melebihi batas waktu, dan dinyatakan sebagai nilai maksimum dari 0 dengan selisih dari waktu penyelesaian dengan batas waktu dapat dinyatakan $T_i = \max[0, c_i - d_i]$. Fungsi obyektif berdasarkan nilai tardiness ini menghitung nilai dari tiap pekerjaan dengan melihat bahwa adanya batas waktu merupakan hal yang penting dan dapat dilihat bahwa sebuah pekerjaan akan semakin kritis dengan tiap waktu setelah melewati batas waktu yang telah ditentukan.

Batasan kapasitas mesin menyatakan jumlah pekerjaan yang aktif (yang akan diproses) pada waktu k pada jenis mesin h harus lebih sedikit atau sama dengan jumlah mesin yang tersedia pada waktu k. Maka variabel integer δ_{ijkh} mempunyai nilai [0,1], yaitu akan sama dengan 1 jika pekerjaan (i, j) aktif saat waktu k pada mesin h dan 0 jika sebaliknya. Untuk merumuskan masalah penjadwalan dengan mesin yang tidak sejenis, H_{ij} menyatakan himpunan mesin yang dapat melakukan operasi (i, j). Maka, batasan kapasitas tiap jenis mesin h adalah :

$$\sum_{ij} \delta_{ijkh} \leq M_{kh} \qquad (3\text{-}11)$$

dengan δ_{ijkh} adalah 1 jika operasi (i, j) dijadwalkan pada jenis mesin $h \in H_{ij}$ pada waktu k dan 0 jika sebaliknya.

I_{ij} menyatakan himpunan operasi sesudah operasi (i, j), maka batasan untuk operasi selanjutnya dapat dinyatakan dengan :

$$c_{ij} + S_{ijl} + 1 \leq b_{il} \qquad (3\text{-}12)$$

dengan i = 1,..,I ; j = 1,..,N_i ; l $\in$ I_{ij} dan S_{ijl} adalah time out antara operasi (i, j) dan (i, l).

t_{ijh} menyatakan waktu proses dari operasi (i, j) pada jenis mesin h $\in$ H_{ij}, kebutuhan waktu proses suatu operasi untuk tiap jenis mesin $h^* \in H_{ij}$:

$$c_{ij} - b_{ij} + 1 = t_{ijh^*} \qquad (3\text{-}13)$$

Batasan kapasitas dan batasan untuk operasi selanjutnya direlaksasi dengan menggunakan pengali lagrange yang berbeda. Untuk melakukannya, permasalahan penjadwalan job shop didekomposisi menjadi sub permasalahan level operasi yang lebih kecil.

3.2.2 DEKOMPOSISI DAN PENYELESAIAN PERSAMAAN SUB PERMASALAHAN

Langkah pertama adalah mengubah pertidaksamaan pada batasan untuk operasi selanjutnya dengan cara :

$$c_{ij} + s_{ijl} + 1 = b_{il}, \mathrm{i} = 1,....,\mathrm{I}\,;\, \mathrm{j} = 1,....,\mathrm{N_i}\,;\, \mathrm{l} \in \mathrm{I_{ij}} \text{ dengan } s_{ijl} \geq S_{ijl} \qquad (3\text{-}14)$$

Minimisasi dari persamaan lagrangian merupakan suatu sub permasalahan level pekerjaan :

$$\min_{(b_{ijl}, s_{ijl}, h \subset H_{ij})} L_i \text{ dengan } L_i = \{w_i T_i^2 + \sum_j [\sum_{k=b_{ij}}^{c_{ij}} \pi_{kh} +$$

$$\sum_{l \subset I_{ij}} [\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - b_{il}) + \frac{p_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - b_{il})^2]]\} \qquad (3\text{-}15)$$

Penambahan koefisien kuadrat penalti diperlukan untuk mencegah terjadinya osilasi. Jika operasi (i, j) bukan merupakan operasi terakhir dari pekerjaan i ($j \neq N_i$) maka waktu awal b_{ij} dipilih dengan suatu pertukaran antara nilai kemampuan suatu mesin $\sum_{k=b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh}$ dengan nilai batasan untuk operasi sesudahnya $\left[\sum_{l \subset I_{ij}} \lambda_{ijl} - \sum_{l:j \subset I_{jl}} \lambda_{iljl}\right] b_{ij}$, yang merupakan suatu fungsi linier dari b_{ij}.

Nilai b_{ij}^* akan sangat kecil jika koefisien dari b_{ij} adalah positif dan besar jika negatif. Hasil dari b_{ij}^* sangat sensitif untuk tiap perubahan kecil dari pengali dan mungkin dapat menyebabkan osilasi selama iterasi. Sehingga bentuk kuadrat pengali ditambahkan untuk mendapatkan keseimbangan antara jadwal awal dengan jadwal sesudahnya untuk penyesuaian semua operasi dari suatu pekerjaan bersamaan.

Sub permasalahan pada level pekerjaan (3-15) tidak dapat didekomposisi menjadi sub permasalahan level operasi. Karena tiap pekerjaan terdiri dari beberapa operasi maka solusi dengan cara enumerasi tidak praktis lagi. Untuk menyelesaikan persoalan ini, digunakan pendekatan iteratif, yaitu waktu awal operasi dari suatu

pekerjaan diselesaikan sesuai dengan urutan, dimulai dari operasi pertama. Pada pemilihan waktu awal b_{ij}, semua yang berhubungan dengan waktu awal operasi b_{ij} (b_{il}, $l \in I_{ij}$ untuk operasi sesudahnya dan b_{il}, $j \in I_{il}$ untuk operasi sebelumnya) diasumsikan tetap, diambil dari nilai akhir komputasi ataupun dari inisialisasi. Dengan semua nilai yang berhubungan dengan waktu awal operasi b_{ij} tetap, maka nilai b_{ij} dengan dapat ditentukan.

Setelah semua waktu awal operasi dari suatu pekerjaan telah dicapai, jika solusi dari iterasi saat itu sama dengan solusi dari iterasi sebelumnya (atau dari inisialisasi), proses telah selesai dan $\{b_{ij}^{*}\}_{j=1}^{N_i}$ telah dicapai.

Waktu awal operasi yang baru dapat digunakan sebagai permulaan untuk iterasi selanjutnya sampai proses selesai atau jumlah iterasi yang ditentukan telah dicapai. Teknik ini disebut teknik iterasi Gauss Seidel, di mana sub permasalahan level operasi dapat dituliskan sebagai berikut :

$$\min_{b_{ij},s_{ijl},h\subset H_{ij}} \{w_i T_i^{\,2} \Delta_{jN_i} + \sum_{k=b_{ij}}^{c_{ij}} \pi_{kh} + \sum_{l\subset I_{ij}} [\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - \bar{b}_{il}) + \frac{p_{ijl}}{2}(b_{ij} + t_{ilh} + s_{ijl} - \bar{b}_{il})^2]$$

$$\sum_{l:j\subset I_{ij}} [\lambda_{ijl}(\bar{b}_{il} + t_{ilh}\cdot + s_{ilj} - b_{ij}) + \frac{p_{ilj}}{2}(\bar{b}_{il} + t_{ilh}\cdot + s_{ilj} - b_{ij})^2]\} \qquad (3\text{-}16)$$

di mana Δ_{jNi} = 1 jika (i, j) adalah operasi terakhir dari pekerjaan i dan 0 jika sebaliknya; $\bar{b}_{il}$ dan $\bar{s}_{ilj}$ adalah nilai awal yang baru dihitung dan waktu

antar operasi yang berhubungan dengan operasi (i, l); dan h^* adalah jenis mesin yang dipilih untuk membentuk operasi sebelumnya (i, l) untuk $j \in I_{il}$.

Pada persamaan (3-16), ada 3 variabel yang akan dipilih : waktu awal operasi b_{ij}, waktu antar operasi s_{ijl}, dan jenis mesin h. Untuk menyelesaikannya dibuat suatu fungsi s_{ijl} dengan dua variabel b_{ij} dan h dan kemudian dienumerasi melalui b_{ij} dan h (karena batasan dalam pembuatan sistem ini adalah jumlah mesin adalah satu untuk tiap jenis mesin, maka h yang dipilih tidak perlu diiterasi lagi). Hal ini dapat dengan mudah ditunjukkan bahwa dengan b_{ij}, h, $\bar{b}_{il}$, $\bar{s}_{ilj}$ yang diberikan, nilai optimal dari s_{ijl} *adalah* $\max[S_{ijl}, s^r_{ijl}]$, di mana s^r_{ijl} adalah pembulatan nilai integer dari $[\bar{b}_{il} - b_{ij} - t_{ijh} - \frac{\lambda_{ijl}}{p_{ijl}}]$. (3-17)

b_{ij}^* dan h^* kemudian dapat dicapai dengan enumerasi waktu awal dari 0 sepanjang waktu horison K untuk tiap jenis $h \in H_{ij}$ yang mungkin.

Apabila dalam iterasi Gauss – Seidel tersebut tidak dicapai nilai yang konvergen (nilai yang sama) tetapi jumlah iterasi yang telah ditentukan telah tercapai, maka penyelesaian yang menghasilkan nilai pekerjaan yang minimal (dari 3-15) digunakan. Untuk mencapai nilai dari (3-15), tiap nilai operasi pada (3-16) dikurangi dengan jumlah dari berikut ini :

$$\sum_{l \subset I_{ij}} [\frac{\lambda_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - \bar{b}_{il}) + \frac{p_{ijl}}{4}(b_{ij} + t_{ilh} + s_{ijl} - \bar{b}_{il})^2 +$$

$$\sum_{l:j \subset I_{ij}} [\frac{\lambda_{ijl}}{2}(\bar{b}_{il} + t_{ilh^*} + s_{ilj} - b_{ij}) + \frac{p_{ilj}}{4}(\bar{b}_{il} + t_{ilh^*} + s_{ilj} - b_{ij})^2 \qquad (3\text{-}18)$$

3.2.3 PENYELESAIAN PERSAMAAN DUAL LAGRANGIAN

Untuk dekomposisi permasalahan, penambahan pengali lagrange dibentuk sesuai dengan batasannya, yaitu batasan kapasitas dikurangi dengan pengali lagrange (π_{kh}) non negatif, batasan untuk operasi selanjutnya dikurangi dengan pengali lagrange (λ_{ijl}) dan bentuk kuadrat penalti dengan koefisien { p_{ijl} } ditambahkan pada fungsi obyektif.

Persamaan dual lagrangian adalah :

$$\max_{\lambda,\pi \geq 0} L, \text{dengan } L = \{-\sum_{kh} \pi_{kh} M_{kh} + \sum_i \min_{b_{ij}, s_{ijl}, h \subset H_{ij}} \{w_i {T_i}^2 +$$

$$\sum_j [\sum_{k=b_{ij}}^{c_{ij}} \pi_{kh} + \sum_{l \subset I_{ij}} [\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - b_{il}) + \frac{p_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - b_{il})2]]\}\}$$

(3-19)

Untuk menyelesaikan dual lagrangian (pada level yang tinggi) tersebut, digunakan metode sub gradient (sub bab 3.1.3) untuk memperbarui pengali lagrange π dan λ.

Pengali lagrange λ, diperbarui dengan rumus :

$$\lambda^{n+1} = \lambda^n + p^n(\lambda^n) \qquad (3\text{-}20)$$

dimana n adalah indeks iterasi pada level tinggi, p^n adalah koefisien penalti pada iterasi ke n dan $g(\lambda^n)$ adalah subgradien dari L berdasarkan λ.

Komponen subgradien berhubungan dengan operasi (i,j) dan (i,l) untuk $l \in l_{ij}$ adalah $\left(b_{ij} + t_{ijh} + s_{ijl} - b_{il}\right)$ (3-21)

Koefisien penalti p_{ijl} dikali dengan suatu faktor ζ (> 1) secara periodik (tiap 5 atau 10 iterasi) jika komponen subgradien $g(\lambda^n)$ bukan nol pada iterasi tersebut. Parameter ζ dipilih sehingga p_{ijl} tidak mengalami penurunan (Bertsekas menyatakan bahwa $\zeta \in [4,10]$).

Pengali Lagrange π diperbarui dengan persamaan :

$$\pi^{n+1} = \pi^n + \alpha^n g(\pi^n) \qquad (3\text{-}22)$$

$g(\pi^n)$ adalah subgradien dari $L(\pi)$ pada π^n dengan

$$g(\pi^n) = \sum_{ij} \delta_{ijkh} - M_{kh} \qquad (3\text{-}23)$$

α^n adalah ukuran dari tiap langkah dengan

$$\alpha^n = \gamma \frac{\overline{L} - L^n}{g\left(\pi^n\right)^T g\left(\pi^n\right)} \qquad (3\text{-}24)$$

Pengali Lagrange π_{kh}, menyatakan nilai dari penggunaan mesin jenis h pada waktu k. Pengali lagrange λ_{ijl} merelaksasi batasan untuk operasi selanjutnya, dapat ditafsirkan sebagai nilai yang mempengaruhi batasan ini dari satu unit waktu. Dari pandangan yang berbeda, dapat ditafsirkan pula sebagai nilai untuk operasi yang saling tumpang tindih atau nilai untuk mengurangi waktu proses t_{ijh}^* atau s_{ijl} oleh satu unit waktu.

3.2.4 MENYUSUN JADWAL YANG FISIBEL

Membuat daftar jadwal berdasarkan waktu awal yang optimal (b_i^*) dari solusi dual lagrangian. Pengurutan jadwal berdasarkan nilai terkecil dari b_i^* (dengan urutan naik). Pekerjaan disusun sesuai dengan mesin yang tersedia saat itu.

Jika beberapa pekerjaan mempunyai b_i^* yang sama tetapi jumlah mesin tidak mencukupi maka dibuat suatu heuristik yang menentukan pekerjaan mana akan dikerjakan terlebih dahulu dan pekerjaan mana akan ditunda. Penundaan pekerjaan dapat menyebabkan peningkatan pada nilai tardiness.

Pekerjaan ditandai pada suatu mesin dengan urutan yang memiliki nilai bobot tardiness terbesar yang akan dikerjakan terlebih dahulu. Ketika semua mesin yang tersedia pada waktu itu telah ditandai maka sisa dari pekerjaan ditunda pada waktu tersebut dan dilanjutkan pada waktu yang lain.

Duality gap, yaitu perbedaan antara nilai fungsi obyektif dengan dual Lagrangian yang meyatakan ukuran dari suatu fungsi obyektif. Nilai dari fungsi obyektif J untuk jadwal yang fisibel adalah batas atas dari nilai optimal J^*. Nilai optimal dari dual Lagrangian L^* adalah batas bawah dari J^*.

Perbedaan antara $\left(J^* - L^*\right)$ merupakan suatu duality gap (3-25)

$\frac{J - L^*}{L^*}$ adalah ukuran dari optimal jadwal yang fisibel berdasarkan bentuk optimalnya. (3-26)

BAB IV

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB IV

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Pengembangan perangkat lunak pada Tugas Akhir ini merupakan implementasi teori – teori tentang penjadwalan dengan menggunakan teknik relaksasi lagrangian yang telah diuraikan pada bab sebelumnya. Pada bab ini akan dibahas tentang perancangan sistem perangkat lunak, dan pembuatan sistem perangkat lunak dengan menjelaskan tahapan – tahapan proses dan struktur data.

Sistem perangkat lunak ini merupakan suatu alat untuk membuat jadwal yang optimal disertai dengan ukuran dari nilai optimal tersebut.

4.1 PERANCANGAN PERANGKAT LUNAK

Perancangan sistem perangkat lunak dilakukan sebelum pembuatan sistem perangkat lunak. Dalam perancangan sistem ini akan dibahas lebih lanjut tentang perancangan data, perancangan proses yang digambarkan dengan diagram aliran data (*data flow diagram*) untuk keperluan analisa kebutuhan dan perancangan antar muka.

4.1.1 PERANCANGAN DATA

Perancangan data dalam pengembangan perangkat lunak ini secara garis besar terdapat tiga macam data yaitu data masukan, data

proses dan data hasil. Data – data ini diperlukan mulai dari awal proses sampai dengan akhir proses. Data – data tersebut disimpan pada suatu variabel dengan kumpulannya dalam bentuk kelas – kelas yang terbagi sesuai dengan jenisnya.

4.1.1.1 DATA MASUKAN

Data masukan merupakan data yang dimasukkan oleh pengguna melalui suatu form tertentu. Data masukan yang diperlukan pada penjadwalan ini adalah :

1. Jumlah pekerjaan yang akan dijadwalkan, bentuk data : array integer.
2. Waktu maksimum (waktu harison) yang merupakan waktu maksimal dari proses yang akan dijadwalkan, bentuk data : integer.
3. Jumlah mesin yang akan digunakan pada proses penjadwalan ini, bentuk data : array integer.
4. Prioritas tiap pekerjaan yang merupakan bobot masing – masing pekerjaan, bentuk data : double.
5. Batas waktu untuk tiap pekerjaan, yang merupakan waktu yang diinginkan untuk menyelesaikan proses pekerjaan tersebut, bentuk data : integer.
6. Jumlah operasi tiap pekerjaan yang merupakan per bagian dari pekerjaan tersebut yang bekerja pada satu mesin, bentuk data : array integer.
7. Mesin yang digunakan tiap operasi untuk mengerjakan prosesnya, bentuk data : integer.

8. Waktu proses tiap operasi, bentuk data : integer.

4.1.1.2 DATA PROSES

Data proses merupakan data yang berasal dari proses transformasi data masukan maupun dari program dan digunakan selama proses dijalankan. Data proses yang dihasilkan antara lain :

1. Waktu antar operasi, yang merupakan waktu kosong antara suatu operasi dengan operasi yang lain, bentuk data : integer.
2. Pengali Lagrange, yang pada proses awal merupakan inisialisasi nilai, bentuk data : double.
3. Waktu penyelesaian tiap pekerjaan, bentuk data : integer.
4. Nilai tardiness yang merupakan selisih antara waktu penyelesaian dengan batas waktu untuk tiap pekerjaan, bentuk data : integer.

4.1.1.3 DATA HASIL

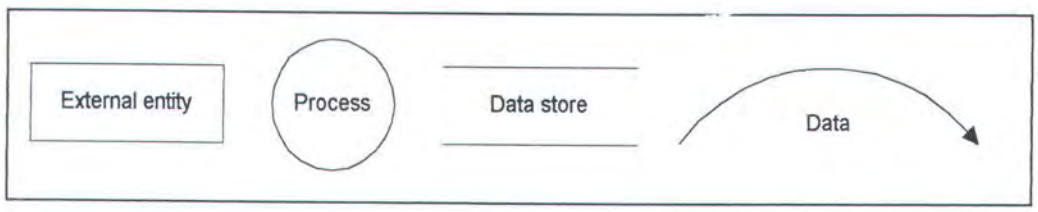
Data hasil merupakan hasil akhir dari proses, antara lain :

1. Waktu awal tiap – tiap pekerjaan, yang akan digunakan sebagai dasar dalam penyusunan jadwal yang fisibel, bentuk data : integer.
2. Nilai dual lagrangian, yang merupakan nilai maksimal dari beberapa iterasi yang berhenti setelah nilai yang tetap telah dicapai, bentuk data : double.
3. Nilai fungsi primal yang optimal setelah nilai dual lagrangian diperoleh, bentuk data : double.

4. Nilai fungsi obyektif setelah jadwal yang fisibel (optimal dan efisien) telah disusun, bentuk data : double.
5. Nilai pengali lagrange setelah diperbarui selama iterasi berlangsung, bentuk data : double.
6. Nilai duality gap, yang merupakan selisih antara nilai fungsi dual dengan nilai fungsi primal, bentuk data : double.

4.1.2 PERANCANGAN PROSES

Perancangan proses digambarkan dalam bentuk Data Flow Diagram (DFD) yang merupakan teknik untuk menggambarkan aliran informasi dan transformasi data dari input sampai dengan output. Simbol dari DFD adalah sebagai berikut :



Gambar 4-1 Simbol DFD

External entity merupakan entitas yang berperan sebagai penghasil atau pemakai informasi yang berada di luar sistem yang dimodelkan.

Process merupakan proses transformasi informasi yang berada dalam sistem.

Data store merupakan penyimpanan data dalam sistem yang dapat diakses oleh satu atau beberapa proses. Biasanya berupa file data dalam media penyimpanan.

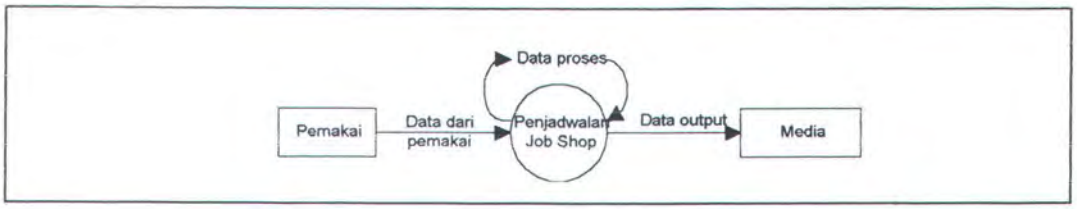
Data merupakan satu atau sekumpulan data atau informasi yang mengalir dari satu proses ke proses lain, dan tanda panah merupakan arah aliran data.

Pada sistem perangkat lunak ini secara garis besar terdiri dari tiga proses utama yaitu :

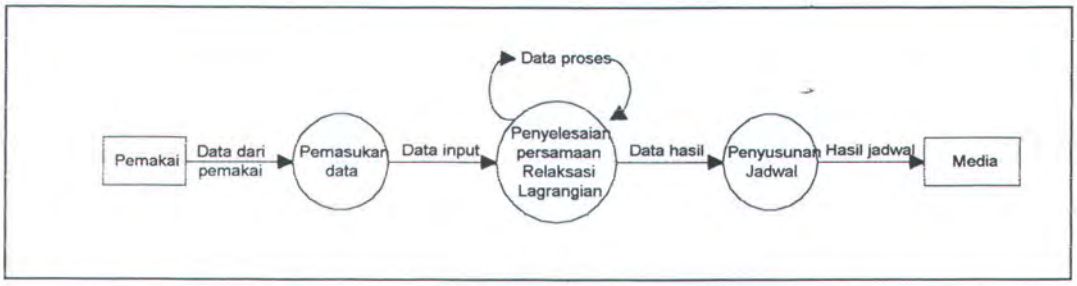
- Proses penyelesaian persamaan relaksasi lagrangian yang merupakan suatu fungsi primal yang mempunyai fungsi dual, data masukan yang diperlukan adalah waktu proses tiap operasi, mesin yang akan digunakan, prioritas tiap pekerjaan. Proses ini akan menghasilkan data proses nilai tardiness, waktu awal, dan waktu penyelesaian tiap pekerjaan. Data proses ini akan dipakai kembali selama proses ini berlangsung. Data hasil yang akan dihasilkan adalah nilai dual dan nilai primal lagrangian.
- Proses perbaruan pengali lagrange. Proses ini memerlukan data proses waktu awal dan waktu penyelesaian tiap operasi, dan pengali lagrange yang digunakan selama proses ini berlangsung. Hasil akhir adalah pengali lagrange yang merupakan akhir dari perbaruan ini.
- Proses penyusunan jadual yang fisibel. Proses ini merupakan proses akhir dan membutuhkan data proses waktu awal tiap operasi untuk menentukan waktu awal yang sebenarnya dengan memperhatikan

batasan – batasan yang telah digunakan. Apabila tiap operasi memiliki nilai yang sama, maka penentuannya berdasarkan nilai prioritas kuadrat tardinessnya. Data hasilnya merupakan nilai fungsi primal setelah jadual yang fisibel disusun.

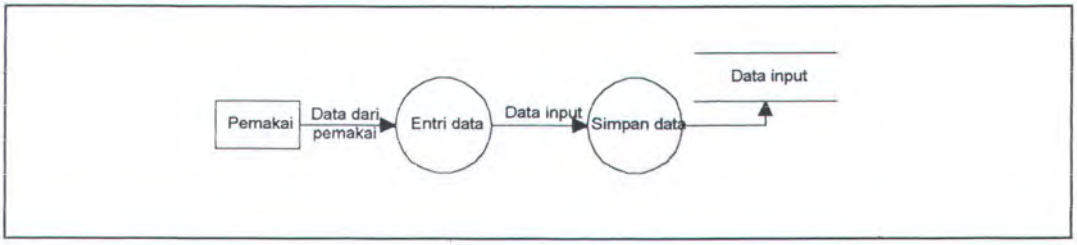
DFD untuk perancangan sistem ini dimulai dengan DFD level 0.



Gambar 4-2 DFD level 0



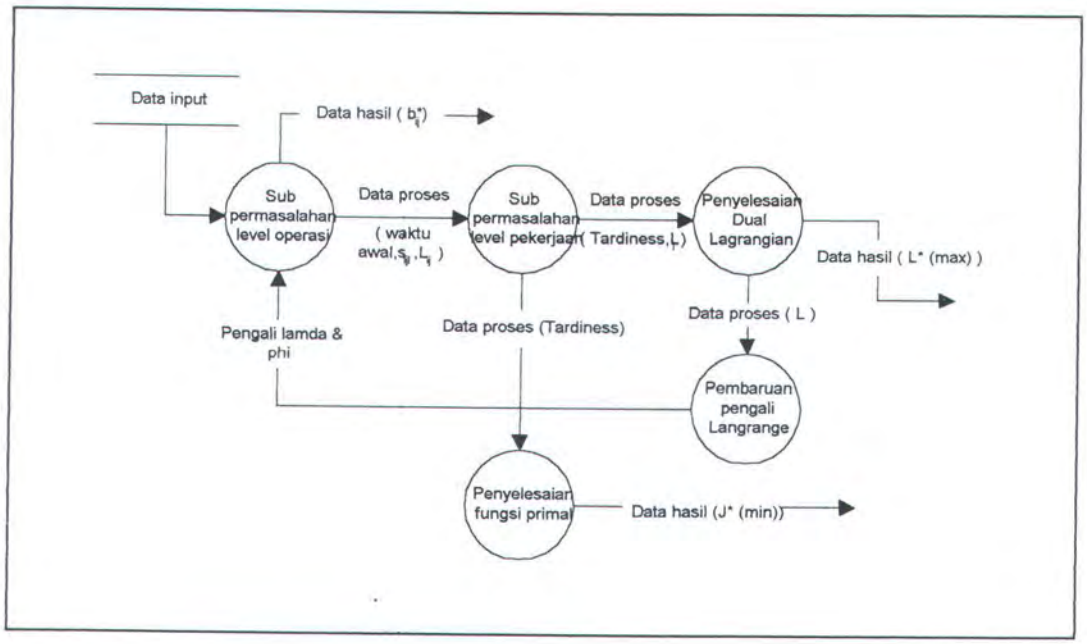
Gambar 4-3 DFD level 1, Detail dari penjadwalan sistem job shop



Gambar 4-4 DFD level 2, Detail dari pemasukan data

Pseudocode untuk level 2 di atas adalah :

```
input(EditJobID->Text);
input(EditWeight->Text);
input(EditRelease->Text);
input(EditDue->Text);
input(EditNumOperation->Text);

input(EditOpID->Text);
input(EditJobID->Text);
input(EditReqOp->Text);
input(EditTypeMachine->Text);
```



Gambar 4-5 DFD level 2, Detail dari penyelesaian persamaan relaksasi Lagrangian

Spesifikasi dari primitif proses di atas dalam bentuk pseudocode adalah :

1. Proses sub level operasi

```
for (1 ≤ j ≤ NumOperation) loop
  for (ReleaseTime ≤ bi < (MaxTime-proccesTime)) loop
    Operation[i][j].BeginTimeOp=bi;
```

```
        interOperation();
        for (j+1 ≤ l ≤ NumOperation) loop
          temp1=Operation[i][j].BeginTimeOp+
              Operation[i][j].TimeReqOp+interOp[i][j][l]-Operation[i][l].
                 BeginTimeOp);
          temp2=temp2+(lamda[i][j][l]*temp1)+ (penalty[i][j][l]/2)*pow(temp1,2);
        end loop for
        for (1 < l < j) loop
          temp3=Operation[i][l].BeginTimeOp+Operation[i][l]. TimeReqOp+
                 interOp[i][l][j]-Operation[i][j].BeginTimeOp;
         temp4=temp4+(lamda[i][l][j]*temp3)+ ( penalty[i][l][j]/2)*pow(temp3,2);
        end loop for
        Operation[i][j].CompleteTimeOp=Operation[i][j].BeginTimeOp+
           Operation[i][j].TimeReqOp-1;
        h=Process.Operation[i][j].MachineUse;
       for (Operation[i][j].BeginTimeOp≤k≤Process. Operation[i][j].CompleteTimeOp)
        loop
            SumPhi=SumPhi+ phi[k][h];
        end loop for
     end loop for
     if (j ≠ Process.Job[i].NumOperation)  L1[i][j]=SumPhi+temp2+temp4;
     else Job[i].CompleteTime=Operation[i][j].CompleteTimeOp;
      tar=Job[i].CompleteTime-Job[i].DueTime-Job[i].ReleaseTime
      if (tar > 0 )  Job[i].Tardiness=tar;
      else  Job[i].Tardiness=0;
       L1[i][j]=SumPhi+temp2+temp4+(Job[i].Weight*((pow(Job[i].Tardiness,2)));
   end  loop  for
  interOperation();
  tar1=Operation[i][j].CompleteTimeOp- Job[i].DueTime-Job[i].ReleaseTime;
  if (tar1 > 0 ) Operation[i][j].Tardiness=tar1;
  else   Operation[i][j].Tardiness=0;
  end if
end loop for
```

2. Proses sub level pekerjaan

```
    for (1 ≤ i ≤ NumJob) loop
     Job[i].BeginTime= Operation[i][1].BeginTimeOp;

   Job[i].CompleteTime=Operation[i][Job[i].NumOperation].CompleteTimeOp;
     tar1=(Job[i].CompleteTime-Job[i].DueTime)-Job[i].ReleaseTime;
     if (tar1 > 0 )  Job[i].Tardiness=tar1;
      else  Job[i].Tardiness=0;
      for (1 ≤ j ≤ Job[i].NumOperation) loop
       h= Operation[i][j].MachineUse;
      end loop for
      for(Operation[i][j].BeginTimeOp≤k≤Operation[i][j].CompleteTimeOp)
         SumPhi=SumPhi+ phi[k][h];
      for (j+1 ≤ l ≤ Job[i].NumOperation) loop
         temp1=Operation[i][j].BeginTimeOp+Operation[i][j].TimeReqOp+
                interOp[i][j][l]-Operation[i][l].BeginTimeOp;
```

```
    temp2=temp2+(lamda[i][j][l]*temp1)+( penalty[i][j][l]/2)*pow(temp1,2);
    temp_Li=temp_Li+SumPhi+temp2;
  end loop for
  Li[i]=temp_Li+( Job[i].Weight*((pow(Process.Job[i].Tardiness,2))));
```

3. Proses pembaruan pengali lagrange

a. Pengali lagrange phi

```
for (0≤k≤MaxTime) loop
 for (1≤h≤NumTypeMachine) loop
  for (1≤i≤NumJob) loop
   for (1≤j≤Job[i].NumOperation) loop
    if (Operation[i][j].MachineUse==h)
     if (Operation[i][j].BeginTimeOp≤k≤Operation[i][j].CompleteTimeOp)
      d[k][h]=d[k][h]+1;
     end if
   end loop for
  end loop for
  g[k][h]=d[k][h]-1;
  if (g[k][h]==0) StepSize[k][h]=0;
  else temp_step=0.05*((Lx-L)/(pow(g[k][h],2)));
  if ( temp_step < 0 ) StepSize[k][h]=0;
  else StepSize[k][h]=temp_step;
  temp_P=phi[k][h]+(StepSize[k][h]*g[k][h]);
  if (temp_P>0) phi[k][h]=temp_P;
  else  phi[k][h]=0;
 end loop for
end loop for
```

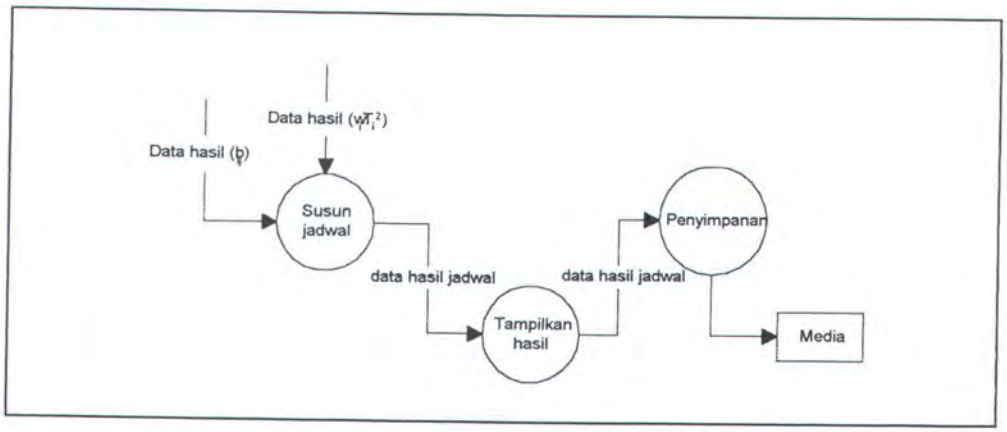
b. pengali lamda

```
for (1≤i≤NumJob) loop
 for (1≤j≤Job[i].NumOperation) loop
  for (j<l≤Job[i].NumOperation) loop
   g1[i][j][l]=Operation[i][j].BeginTimeOp+Operation[i][j].TimeReqOp
          +interOp[i][j][l]-Operation[i][l].BeginTimeOp;
   if (enumerate%10==0)
    if (g1[i][j][l]!=0) penalty[i][j][l]=penalty[i][j][l]*4;
   lamda[i][j][l]=lamda[i][j][l]+(penalty[i][j][l]*g1[i][j][l]);
  end loop for
 end loop for
end loop for
```

4. Proses penyelesaian fungsi primal dan dual

```
for (1≤i≤NumJob) loop
 temp_L=temp_L+Process.Li[i];
```

```
ObjectiveFunction=temp_L;
for (0≤k≤MaxTime) loop
 for (1≤h≤NumTypeMachine) loop
   TempPhi=TempPhi+ phi[k][h];
 end loop for
end loop for
L=(-TempPhi)+temp_L;
end loop for
```



Gambar 4-6 DFD level 2, Detail dari penyusunan jadwal yang fisibel

Spesifikasinya adalah :

```
while (temp_sum != sum_op) loop
 for (1≤h≤NumTypeMachine) loop
  for (1≤i≤NumJob) loop
   if ((queue[i] ≤Job[i].NumOperation)&(queue[i]!=0))
    if ((Operation[i][queue[i]].MachineUse=h))
     status_temp=true;
       bij=Operation[i][queue[i]].BeginTimeOp;
       if (bij<temp_awal)
         temp_awal=bij; temp_i=i; temp_j=queue[i];
       end if
       else if (bij==temp_awal)
        T =Job[i].Weight*(pow(Operation[i][queue[i]].Tardiness,2));
        temp_T =Job[temp_i].Weight*(pow(Operation [temp_i][temp_j].
                 Tardiness,2));
        if (T>temp_T)
          temp_awal=bij; temp_i=i; temp_j=queue[i];
        end if
        else if (T==temp_T)
         if (Job[i].Weight>Job[temp_i].Weight)
           temp_awal=bij; temp_i=i; temp_j=queue[i];
          else if (Job[i].Weight=Job[temp_i].Weight)
           if (Operation[i][queue[i]].Tardiness>Operation[temp_i][temp_j].
```

```
                Tardiness)
                temp_awal=bij; temp_i=i; temp_j=queue[i];
              end if
              else if (Operation[i][queue[i]].Tardiness= Operation[temp_i]
                    [temp_j].Tardiness)
               if (Operation[i][queue[i]]. CompleteTimeOp <Operation[temp_i]
                  [temp_j].CompleteTimeOp)
                 temp_awal=bij; temp_i=i; temp_j=queue[i];
               end if
               else if (Operation[i][queue[i]]. CompleteTimeOp= Operation
                      [temp_i][temp_j].CompleteTimeOp)
                 if (Job[i].DueTime<Job[temp_i].DueTime)
                   temp_awal=bij; temp_i=i; temp_j=queue[i];
                 end if
               end else if
             end else if
            end else if
          end else if
        end loop for
    if ( status_temp=true)
     while (Machine[h].index[index].IDJob!=0)
       index++;
     Machine[h].index[index].IDJob=temp_i;
     Machine[h].index[index].IDOp=temp_j;
     if (temp_j == Job[temp_i].NumOperation) queue[temp_i]=0;
     else  ++queue[temp_i];
     temp_sum++;
     status_temp=false;
   end loop for
  end loop while
```

4.1.3 PERANCANGAN ANTAR MUKA

Untuk menghasilkan hasil implementasi perangkat lunak yang dibangun, digunakan dua form utama yaitu form frmInput dan form frmSchedule.

FrmInput merupakan suatu fasilitas untuk memasukkan data – data input yang diperlukan dalam pejadwalan ini. Data yang dimasukkan merupakan data yang baru.

FrmScedule merupakan form untuk menghasilkan jadwal. Dalam form ini dilakukan proses penyelesaian persamaan relaksasi lagrangian

yang dapat menghasilkan nilai untuk fungsi primal dan fungsi dual dan proses penyusunan jadwal berdasarkan hasil dari proses sebelumnya. Selain itu, juga ditampilkan nilia pengali lagrange yang merupakan nilai dari batasan – batasan dari persamaan fungsi obyektifnya.

Menu pada frmSchedule ini terdapat Run|Solving Dual Lagrange, ini fungsinya untuk menyelesaikan persamaan – persamaan pada relaksasi lagrangian. Menu Run|Feasible Schedule untuk menyusun hasil yang dari proses sebelumnya menjadi suatu jadwal. Menu File|Save Multiplier untuk menyimpan pengali – pengali Lagrange yang telah dihasilkan. Menu Multiplier untuk menampilkan hasil pengali Lagrange dan menu Feasible Schedule untuk menampilkan jadwal yang dihasilkan untuk tiap – tiap mesin yang digunakan.

4.2 PEMBUATAN PERANGKAT LUNAK

Sub bab ini membahas lebih lanjut tentang struktur data yang digunakan dan fungsi dasar yang diaplikasikan dalam implementasi rancangan perangkat lunak ke dalam bahasa pemrograman. Sistem ini dibangun dengan menggunakan Borland C++ Builder 3.0.

4.2.1 IMPLEMENTASI DATA

Untuk pembacaan dan penyimpanan suatu urutan dari nilai digunakan suatu array. Sebuah array yang dasar, membutuhkan suatu deklarasi ukuran array sehingga suatu kompiler dapat mengalokasikan sejumlah memori yang tepat dan array yang akan digunakan harus dideklarasikan sebelum pemakaian array pertama kali. Suatu array dinamik digunakan karena tidak ada pembatasan ukuran dan penentuan ukuran array dapat diperbesar ataupun diperkecil selama program berlangsung. Suatu array dinamik dideklarasikan pertama kali dengan menggunakan *new* dan setelah selesai pemakaian, dihilangkan dari memori dengan menggunakan *delete*. Contoh pada program untuk sistem ini adalah :

Dibuat array baru :

```
Li = new float [NumJob+1];
```

Dan dihilangkan dari memori :

```
Delete[] Li;
```

Suatu bentuk kelas hampir sama dengan struktur (*struct*) kecuali bahwa pada kelas, semua anggota tidak dapat dipakai oleh pengguna secara umum dari kelas lain (dideklarasikan dengan *private*), dan hanya dapat digunakan oleh fungsi yang merupakan anggota dari kelas tersebut, fungsi tersebut disebut fungsi anggota (*member function*). Sedangkan untuk anggota yang dapat dipakai secara umum oleh kelas lain dideklarasikan dengan *public*. Jika kita memanggil sebuah member function, maka kita mengirim sebuah pesan pada suatu obyek. Penggunaan kelas – kelas dilakukan agar lebih terstruktur, mempermudah perhitungan dan penyimpanan data proses dan data hasil. Kelas – kelas tersebut antara lain :

1. Kelas LROperation

Kelas ini terdiri dari variabel – variabel dari operasi pada tiap pekerjaan. Tiap operasi mempunyai waktu proses, waktu awal proses, waktu senggang, mesin yang digunakan untuk memproses operasi tersebut, dan waktu penyelesaian proses.

```
class LROperation
{
public:
   int IDOperation;
   int TimeReqOp;
   int BeginTimeOp;
   int TimeOut;
   int Tardiness;
   int CompleteTimeOp;
   int MachineUse;
   LROperation();
   void GetDataOperation(int,int,int,int);
};
```

2. Kelas LRJob

Kelas ini terdiri dari variabel - variabel pada suatu pekerjaan. Tiap pekerjaan mempunyai batas waktu, nilai tardiness, prioritas, waktu awal proses, waktu penyelesaian tiap pekerjaan dan jumlah operasi yang dimilikinya.

```
class LRJob
{
public:
   int IDJob;
   int DueTime;
   int Tardiness;
   float Weight;
   int BeginTime;
   int CompleteTime;
   int NumOperation;
   LRJob();
   void GetDataJob(int,double,int,int,int);
};
```

3. Kelas LRMachine

Kelas ini terdiri dari variabel ID yang merupakan nomor mesin yang digunakan pada suatu proses, dan variabel index yang berupa suatu array untuk menyimpan ID tiap operasi yang akan dikerjakan pada mesin tersebut.

```
class LRMachine
{
public:
int IDMachine;
   struct Index {
     int IDJob;
     int IDOp;
   } *index;
};
```

4. Kelas LRProcess

Kelas ini adalah kelas yang paling utama, karena pada kelas ini terdapat semua fungsi dan prosedur yang digunakan untuk proses pada sistem dan perhitungan penyelesaian masalah.

```
class LRProcess
{
public:
   int NumJob;
   int NumTypeMachine;
   int MaxTime;
   LRJob *Job;
   LRMachine *Machine;
   LROperation **Operation;
   double ***lamda;
   int ***interOp;
   double **phi;
   double ***penalty;
   double **L1;
   double *Li;
   double L;
   double **d;
   double **g;
   double **StepSize;
   double ***g1;
   double Lx;
   double ObjectiveFunction;
   int *queue;

   LRProcess();
   void SetFirst(const int,const int,const int);
   void SetJob(int,LRJob);
   void SetMachine(int,int);
   void SetOperation(int,int,LROperation);
   void Initial();
   void interOperation();
   void SetNol();
   void Update_multiplier_phi();
   void Update_multiplier_lamda(int const);
   void feasible(int,bool *);
   void Construct_schedule(int);
   void LeftOver();
};
```

4.2.1.1 IMPLEMENTASI DARI DEKLARASI KELAS

Implementasi ini menjelaskan tentang variabel dan fungsi yang ada pada kelas – kelas yang telah terbentuk.

1. Kelas LROperation

a. Constructor, merupakan inisialisasi awal variabel yang akan digunakan.

```
LROperation::LROperation()
  {
    IDOperation=1; TimeReqOp=1; TimeOut=0;
    BeginTimeOp=0; CompleteTimeOp=1; MachineUse=1;
  }
```

b. Fungsi GetDataOperation merupakan fungsi untuk menyimpan sementara nilai pada kelas LROperation sebelum disimpan pada variabel yang tetap.

```
void LROperation::GetDataOperation(int id,int idj,int t,int m)
{
      IDOperation=id;
      IDJob=idj;
      TimeReqOp=t;
      MachineUse=m;
      TimeOut=1;
}
```

2. Kelas LRJob

a. Constructor, merupakan inisialisasi awal pada kelas LRJob.

```
LRJob::LRJob()
{
  IDJob=1; DueTime=1;
  Tardiness=0; Weight=1;
  BeginTime=1; CompleteTime=1;
  NumOperation=1;
}
```

c. Fungsi GetDataJob, merupakan fungsi yang menyimpan sementara nilai dari kelas LRJob sebelum disimpan pada variabel yang tetap.

```
void LRJob::GetDataJob(int id,double w,int r,int d,int n)
{
  IDJob = id; Weight = w; ReleaseTime=r; DueTime=d;
  NumOperation = n;
}
```

3. Kelas LRProcess

a. Constructor, untuk inisialisasi.

```
LRProcess::LRProcess()
{
  Job=0;
  Machine=0;
  Operation=0;
}
```

b. Fungsi SetFirst, merupakan fungsi untuk menyimpan nilai variabel yang pertama kali ditentukan nilainya.

```
void LRProcess::SetFirst(const int nj,const int nm,const int th)
{
   NumJob=nj;
   NumTypeMachine=nm;
   MaxTime=th;
}
```

c. Fungsi SetJob, adalah fungsi untuk memasukkan nilai ke dalam array yang telah dibuat berdasarkan nilai pada fungsi GetDataJob pada kelas LRJob.

```
void LRProcess::SetJob(int id,LRJob job)
{
   Job[id].IDJob=job.IDJob;
   Job[id].BeginTime = 0;
   Job[id].Weight=job.Weight;
   Job[id].ReleaseTime=job.ReleaseTime;
   Job[id].DueTime=job.DueTime;
   Job[id].NumOperation=job.NumOperation;
}
```

d. Fungsi SetOperation adalah fungsi untuk memasukkan nilai ke dalam array yang telah dibuat berdasarkan nilai pada fungsi GetDataOperation pada kelas LROperation. Operation diimplementasikan dalam bentuk array 2 dimensi, yaitu untuk penyimpanan IDJob dan IDOperation untuk memudahkan pencarian dan pembacaan data.

```
void LRProcess::SetOperation(int id,int idJob,LROperation op)
    {
    Operation[idJob][id].IDOperation = op.IDOperation;
    Operation[idJob][id].IDJob = idJob;
    Operation[idJob][id].TimeReqOp = op.TimeReqOp;
    Operation[idJob][id].MachineUse = op.MachineUse;
    Operation[idJob][id].TimeOut = op.TimeOut;
    }
```

e. Fungsi SetMachine, merupakan fungsi untuk menyiapkan banyaknya indeks yang digunakan dalam penyusunan jadual berdasarkan jumlah operasi yang menggunakan mesin tersebut.

```
void LRProcess::SetMachine(int m,int r)
{ Machine[m].IDMachine=m;
  Machine[m].index = new Index[r+1];
  for(int y=0;y<=r;y++)
  {
    Machine[m].index[y].IDJob=0;
    Machine[m].index[y].IDOp=0;
  }
}
```

Machine[m].index[y] digunakan untuk menyimpan ID dari operasi yang bekerja pada mesin jenis m.

f. Fungsi Initial, yang merupakan fungsi untuk membuat array yang dinamik untuk semua variabel pada kelas LRProcess yang berupa array satu, dua dimensi maupun tiga dimensi.

```
void LRProcess::Initial()
{
   Li = new double [NumJob+1];
   interOp = new int **[NumJob+1];
   L1 = new double *[NumJob+1];
   lamda = new double **[NumJob+1];
   penalty = new double **[NumJob+1];
   g1 = new double **[NumJob+1];
   queue = new int [NumJob+1];

   for (int i=1;i<=NumJob;i++)
   {  interOp[i] = new int *[Job[i].NumOperation+1];
      L1[i] = new double [Job[i].NumOperation+1];

      lamda[i] = new double *[Job[i].NumOperation+1];
      penalty[i] = new double *[Job[i].NumOperation+1];
      g1[i] = new double *[Job[i].NumOperation+1];
      for ( int j=1;j<=Job[i].NumOperation;j++)
      { lamda[i][j] = new double [Job[i].NumOperation+1];
        penalty[i][j] =new double [Job[i].NumOperation+1];
        g1[i][j] =  new double [Job[i].NumOperation+1];
        interOp[i][j] = new int [Job[i].NumOperation+1];
      }}

   phi = new double *[MaxTime+1];
```

```
d = new double *[MaxTime+1];
g = new double *[MaxTime+1];
StepSize = new double *[MaxTime+1];

for (int k=0;k<=MaxTime;k++)
{ phi[k] = new double [NumTypeMachine+1];
  d[k] = new double [NumTypeMachine+1];
  g[k] = new double [NumTypeMachine+1];
  StepSize[k]= new double [NumTypeMachine+1]; }

for (int i=1;i<=NumJob;i++){
   for (int j=1;j<=Job[i].NumOperation;j++)
   { for ( int l=1;l<=Job[i].NumOperation;l++) {
        lamda[i][j][l]=0.1;
        penalty[i][j][l]=0.5;
        g1[i][j][l]=0;

       interOp[i][j][l]=Operation[i][j].TimeOut; }
   } }

for (int k=0;k<=MaxTime;k++)
{ for (int h=1;h<=NumTypeMachine;h++)
  { phi[k][h]=0;
    d[k][h]=0;
    g[k][h]=0;
    StepSize[k][h]=0; } }
}
```

g. Karena penggunaan array dinamik memakai alamat memori maka akhir dari program ini semua array dihapus. Prosedur untuk menghapusnya adalah :

```
for (int i=1;i<=Process.NumJob;i++)  {
     delete[] Process.Operation[i];
     delete[] Process.L1[i];
     for (int j=1;j<=Process.Job[i].NumOperation;j++)
     {  delete[] Process.penalty[i][j];
        delete[] Process.lamda[i][j];
        delete[] Process.g1[i][j];
        delete[] Process.interOp[i][j]; }
     delete[] Process.lamda[i];
     delete[] Process.penalty[i];
     delete[] Process.g1[i];
     delete[] Process.interOp[i];  }

  for (int h=1;h<=Process.NumTypeMachine;h++)
    delete[] Process.Machine[h].index;
```

```
for (int k=0;k<=Process.MaxTime;k++){
    delete[] Process.phi[k];
    delete[] Process.d[k];
    delete[] Process.g[k];
    delete[] Process.StepSize[k]; }

 delete[] Process.Operation;
 delete[] Process.Job;
 delete[] Process.Machine;
 delete[] Process.interOp;
 delete[] Process.lamda;
 delete[] Process.penalty;
 delete[] Process.phi;
 delete[] Process.Li;
 delete[] Process.L1;
 delete[] Process.d;
 delete[] Process.g;
 delete[] Process.StepSize;
 delete[] Process.g1;
 delete[] Process.queue;
```

4.2.2 IMPLEMENTASI PROSES

Perangkat lunak ini diimplementasikan dalam 3 proses utama yaitu proses pemasukan data, proses penyelesaian persamaan relaksasi lagrangian dan proses penyusunan jadwal. Dalam proses penyelesaian persamaan relaksasi lagrangian, dapat dipecah menjadi empat sub proses yang merupakan suatu proses yang berurutan, proses tersebut adalah proses sub permasalahan level operasi, sub permasalahan level pekerjaan, penyelesaian persamaan dual dan penyelesaian persamaan primal.

4.2.2.1 PROSES PEMASUKAN DATA

Proses ini terdiri dari proses entri data oleh pemakai melalui form frmIput dan penyimpanan data ke dalam suatu file. Ada 3 jenis file yang digunakan untuk menyimpan data masukan, yaitu *.in untuk menyimpan secara umum jumlah pekerjaan, jumlah mesin dan waktu maksimal, *.j untuk menyimpan data yang berkaitan dengan pekerjaan (terutama yang berkaitan dengan kelas LRJob) dan yang terakhir adalah file *.op untuk meyimpan data yang berkaitan dengan operasi – operasi tiap pekerjaan (yang berkaitan dengan kelas LROperation).

4.2.2.2 PROSES PENYELESAIAN PERSAMAAN RELAKSASI LAGRANGIAN

Proses ini terdiri dari beberapa sub proses, yaitu proses sub permasalahan level operasi, proses sub permasalahan level pekerjaan, proses penyelesaian dual Lagrangian, proses pembaruan pengali Lagrange dan proses penyelesaian persamaan primal (fungsi obyektif). Beberapa hasil dari proses ini akan digunakan untuk proses selanjutnya yaitu proses penyusunan jadwal.

4.2.2.2.1 PROSES SUB PERMASALAHAN LEVEL OPERASI

Proses ini merupakan proses untuk menyelesaikan persamaan yang merupakan sub permasalahan level rendah yaitu level operasi. Proses ini merupakan proses awal dalam penyelesaian persamaan relaksasi Lagrangian.

Pada proses ini, dihasilkan data proses waktu antar operasi, waktu awal operasi (yang pada akhir iterasi merupakan data hasil) dan L_{ij} yang merupakan nilai persamaan Lagrange pada sub level operasi.

Prosedur waktu antar operasi adalah sebagai berikut :

(Dari rumus (3-17))

```
void LRProcess::interOperation()
{ int Temp=0;
  for (int i=1;i<=NumJob;i++)
  { for (int j=1;j<=Job[i].NumOperation;j++)
    { for (int l=Job[i].NumOperation;l>j;l--)
      {
       Temp=Operation[i][l].BeginTimeOp-Operation[i][j].BeginTimeOp-
              Operation[i][j].TimeReqOp-ceil(lamda[i][j][l]/penalty[i][j][l]);
       if (Temp > Operation[i][j].TimeOut)  interOp[i][j][l]=Temp;
       else   interOp[i][j][l]=Operation[i][j].TimeOut; }
      }}}
```

Penentuan waktu awal operasi dilakukan dengan cara melakukan enumerasi dari nilai k=0 sampai dengan k=waktu maksimum-waktu yang dibutuhkan oleh pekerjaan tersebut untuk melakukan proses.

Untuk menyelesaikan persamaan (3-16), digunakan teknik iterasi Gauss – Seidel. Untuk menentukan nilai dari waktu awal operasi, waktu awal yang berhubungan dengannya telah ditentukan terlebih dahulu, diambil dari nilai iterasi sebelumnya atau dari nilai inisialisasi. Setelah nilai waktu awal semua operasi telah ditentukan, jika solusi untuk iterasi saat itu sama dengan solusi untuk iterasi sebelumnya maka proses telah mengalami konvergen. Akan tetapi, jika tidak sama maka waktu awal operasi yang baru saja dihitung digunakan untuk iterasi selanjutnya sampai proses mengalami konvergen atau iterasi telah mencapai nilai jumlah iterasi yang telah ditentukan .

Cuplikan program untuk level operasi ini adalah sebagai berikut :

(Dari rumus (3-16))

```
for (int j=1;j<=Process.Job[i].NumOperation;j++){
  for (int bi=Process.Job[i].ReleaseTime;
      bi<(Process.MaxTime-temp_t) ;bi++)
  {
    Process.Operation[i][j].BeginTimeOp=bi;
    Process.interOperation();
    temp1=0, temp2=0;
    for (int l=j+1;l<=Process.Job[i].NumOperation;l++)
    {
      temp1=(double)(Process.Operation[i][j].BeginTimeOp+Process.
        Operation[i][j].TimeReqOp+Process.interOp[i][j][l]-Process.
        Operation[i][l].BeginTimeOp);
      temp2=temp2+((Process.lamda[i][j][l]*temp1)+((Process.penalty
          [i][j][l]/2)* (pow(temp1,2))));
    }
    temp3=0;temp4=0;
    for (int l=1;l<j;l++) {
      temp3=(double)(Process.Operation[i][l].BeginTimeOp+Process.
          Operation[i][l].TimeReqOp+Process.interOp[i][l][j]-Process.
          Operation[i][j].BeginTimeOp);
      temp4=temp4+((Process.lamda[i][l][j]*temp3)+((Process.penalty
          [i][l][j]/2)* (pow(temp3,2))));
    }
    Process.Operation[i][j].CompleteTimeOp=Process.Operation[i][j].Begin
      TimeOp+Process. Operation[i][j].TimeReqOp-1;
    h=Process.Operation[i][j].MachineUse;
    SumPhi=0;
    for (int k=Process.Operation[i][j].BeginTimeOp;k<=Process. Operation
      [i][j].CompleteTimeOp;k++)
      SumPhi=SumPhi+Process.phi[k][h];
    int tar=0;
    if (j!=Process.Job[i].NumOperation)
      Process.L1[i][j]=SumPhi+temp2+temp4;
    else {
     Process.Job[i].CompleteTime=Process.Operation[i][j].CompleteTimeOp;
     tar=(Process.Job[i].CompleteTime-Process.Job[i].DueTime)-Process.
       Job[i].ReleaseTime;
     if (tar > 0 )Process.Job[i].Tardiness=tar;
     else Process.Job[i].Tardiness=0;
     Process.L1[i][j]=SumPhi+temp2+temp4+(Process.Job[i].Weight*
       ((pow(Process.Job[i].Tardiness,2))));
    }
  }
  Process.interOperation();
  tar1=(Process.Operation[i][j].CompleteTimeOp-Process.Job[i].DueTime)-
      Process.Job[i].ReleaseTime;
  if (tar1 > 0 )Process.Operation[i][j].Tardiness=tar1;
  else Process.Operation[i][j].Tardiness=0;
} }
```

4.2.2.2.2 PROSES SUB PERMASALAHAN LEVEL PEKERJAAN

Proses ini merupakan kelanjutan dari proses sub permasalahan level operasi, karena tiap pekerjaan terdiri dari operasi yang merupakan proses – proses dari suatu pekerjaan pada mesin tertentu. Proses ini menghasilkan nilai tardiness yang merupakan selisih antara waktu penyelesaian tiap pekerjaan dengan batas waktu yang telah ditentukan. Selain itu, proses ini menghasilkan nilai Lagrange yang akan digunakan pada proses selanjutnya yaitu proses penyelesaian dual lagrangian. Nilai lagrange (L_i) merupakan nilai minimum yang dalam perhitungannya memakai data proses yang berasal dari proses sebelumnya yaitu proses permasalahan level operasi.

Cuplikan program untuk sub level pekerjaan adalah sebagai berikut:

(Dari rumus (3-15))

```
For (int i=1;I<=Process.NumJob;i++) {
Process.Job[i].BeginTime=Process.Operation[i][1].BeginTimeOp;
Process.Job[i].CompleteTime=Process.Operation[i][Process.Job[i].
      NumOperation].CompleteTimeOp;
int tar1=0;
tar1=(Process.Job[i].CompleteTime-Process.Job[i].DueTime)-Process.
      Job[i].ReleaseTime;
if (tar1 > 0 )Process.Job[i].Tardiness=tar1;
else Process.Job[i].Tardiness=0;
temp_Li=0;
for (int j=1;j<=Process.Job[i].NumOperation;j++){
  SumPhi=0;
  h=Process.Operation[i][j].MachineUse;
  for (int k=Process.Operation[i][j].BeginTimeOp; k<=Process.Operation
       [i][j].CompleteTimeOp;k++)
    SumPhi=SumPhi+Process.phi[k][h];
  temp1=0, temp2=0;
  for (int l=j+1;l<=Process.Job[i].NumOperation;l++) {
   temp1=(double)(Process.Operation[i][j].BeginTimeOp+Process.Operation
        [i][j].TimeReqOp+Process.interOp[i][j][l]-
Process.Operation[i][l].BeginTimeOp);
    temp2=temp2+((Process.lamda[i][j][l]*temp1)+((Process.penalty
          [i][j][l]/2)*(pow(temp1,2))));
  }
```

```
    temp_Li=temp_Li+SumPhi+temp2;
}
Process.Li[i]=temp_Li+(Process.Job[i].Weight*((pow(Process.Job[i].
                Tardiness,2)))); }
```

4.2.2.2.3 PROSES PENYELESAIAN PERSAMAAN DUAL LAGRANGIAN DAN PERSAMAAN PRIMAL

Proses ini merupakan penyelesaian untuk persamaan dual dari fungsi obyektif yang telah ditentukan. Nilai yang dicapai adalah merupakan nilai maksimum dari beberapa iterasi yang telah dilakukannya. Untuk menyelesaikan persamaan dual, maka digunakan metode subgradien. Pada metode ini, pembaruan nilai pengali lagrange dilakukan tiap iterasi disertai dengan penyelesaian tiap sub permasalahannya.

Persamaan primal merupakan fungsi obyektif dari penjadwalan ini. Nilai untuk persamaan primal merupakan batas atas dari nilai dual Lagrangian karena fungsi ini merupakan fungsi konfeks. Apabila ada perbedaan antara nilai primal dengan nilai dual maka perbedaan itu disebut duality gap.

Proses ini merupakan kelanjutan dari proses – proses sebelumnya. Dari rumus (3-19) dapat dilihat bahwa nilai L merupakan nilai penjumlahan dari persamaan (3–15) yang merupakan nilai penyelesaian dari level pekerjaan. Cuplikan dari program yang khusus memakai persamaan ini adalah sebagai berikut :

(Dari rumus (3-19))

```
temp_L=0;
```

```
for (int i=1;i<=Process.NumJob;i++)
   temp_L=temp_L+Process.Li[i];

Process.ObjectiveFunction=temp_L;
TempPhi=0;
for (int k=0;k<=Process.MaxTime;k++)
 for (int h=1;h<=Process.NumTypeMachine;h++)
    TempPhi=TempPhi+Process.phi[k][h];
Process.L=(-TempPhi)+temp_L;
Process.Lx=ceil(Process.L);
Process.SetNol();
Process.Update_multiplier_lamda(enumerate);
Process.Update_multiplier_phi();
```

Cuplikan dari rumus (3-25) adalah :

```
EditDuality->Text=FloatToStr(((Process.ObjectiveFunction-Process.L)/
Process.ObjectiveFunction)*100)+" %";
```

4.2.2.2.4 PROSES PEMBARUAN PENGALI LAGRANGE

Pembaruan pengali lagrange dilakukan untuk mendapatkan nilai dual lagrange yang maksimum. Pengali lagrange π mempunyai nilai positif. Pengali ini merupakan nilai kapasitas dari mesin. Pengali lagrange λ merupakan suatu nilai untuk operasi selanjutnya. Pembaruan dilakukan secara iterasi sampai nilai untuk dual Lagrangian tidak berubah dan merupakan nilai maksimum.

Fungsi untuk perbaruan dua jenis pengali Lagrange ini adalah sebagai berikut :

(Dari rumus (3-21))

```
void LRProcess::Update_multiplier_phi()
{
   float temp_P=0,temp_step=0;
   for (int k=0;k<=MaxTime;k++) {
    for (int h=1;h<=NumTypeMachine;h++) {
     for (int i=1;i<=NumJob;i++) {
      for (int j=1;j<=Job[i].NumOperation;j++)
      {
```

```
        if (Operation[i][j].MachineUse==h)
          if (k>=Operation[i][j].BeginTimeOp&&k<= Operation
            [i][j].CompleteTimeOp)
           d[k][h]=d[k][h]+1;
        }
        g[k][h]=d[k][h]-1;
        if (g[k][h]==0)StepSize[k][h]=0;
        else { temp_step=0.05*((Lx-L)/(pow(g[k][h],2)));
          if ( temp_step < 0 )StepSize[k][h]=0;
          else StepSize[k][h]=temp_step;
        }
        temp_P=phi[k][h]+(StepSize[k][h]*g[k][h]);
        if (temp_P>0) phi[k][h]=temp_P;
        else phi[k][h]=0;
      }}}
```


(Dari rumus (3-20))

```
void LRProcess::Update_multiplier_lamda(int const enumerate)
{
 for (int i=1;i<=NumJob;i++)
  for (int j=1;j<=Job[i].NumOperation;j++)
    for (int l=Job[i].NumOperation;l>j;l--)
    {
      g1[i][j][l]=Operation[i][j].BeginTimeOp+Operation[i][j].
       TimeReqOp+interOp[i][j][l]-Operation[i][l].BeginTimeOp;
      if (enumerate%10==0)
       if (g1[i][j][l]!=0)
        penalty[i][j][l]=penalty[i][j][l]*4;
        lamda[i][j][l]=lamda[i][j][l]+(penalty[i][j][l]
          *g1[i][j][l]);
      } }
```

4.2.2.3 PROSES PENYUSUNAN JADWAL

Proses ini dilakukan, karena hasil dari dual Lagrangian tidak menghasilkan jadwal yang fisibel karena adanya batasan untuk kapasitas mesin dan batasan operasi selanjutnya. Penyusunan jadwal berdasarkan nilai waktu awal tiap operasi yang telah dihasilkan pada sub permasalahan level operasi yang diurutkan dari waktu terkecil sampai terbesar. Jika ada operasi yang mempunyai waktu awal yang sama, maka pengurutan

berdasarkan nilai prioritas dari tiap pekerjaan dikalikan dengan kuadrat dari nilai tardinessnya.

Fungsi untuk menyusun jadwal ini ada 3 macam, yaitu fungsi Construct_Schedule() yang digunakan untuk menyusun operasi yang akan diletakkan pada indeks tiap mesin berdasarkan mesin yang digunakan oleh operasi tersebut, fungsi Feasible() yang digunakan untuk menyusun jadwal berdasarkan batasan kapasitas mesin dan batasan operasi selanjutnya dan yang terakhir adalah fungsi LeftOver() yang digunakan untuk memindahkan posisi operasi bila memungkinkan untuk digeser ke kiri. Fungsi – fungsi tersebut adalah sebagai berikut :

```
void LRProcess::Construct_schedule(int sum_op)
{
  while (temp_sum != sum_op) {
   for (int h=1;h<=NumTypeMachine;h++) {
    index=0; temp_awal=999999;
    for (int i=1;i<=NumJob;i++)  {
     if ( (queue[i]<=Job[i].NumOperation) && (queue[i]!=0)) {
      if ((Operation[i][queue[i]].MachineUse==h) ) {
       status_temp=true;
       bij=Operation[i][queue[i]].BeginTimeOp;
       if (bij<temp_awal) {
         temp_awal=bij; temp_i=i;
         temp_j=queue[i]; }
       else  { if (bij==temp_awal) {
          T=Job[i].Weight*(pow((double)Operation[i][queue[i]].Tardiness,2));
          temp_T=Job[temp_i].Weight*(pow((double)Operation[temp_i]
             [temp_j].Tardiness,2));
          if (T>temp_T) {
            temp_awal=bij; temp_i=i;
            temp_j=queue[i];  }
          else if (T==temp_T) {
            if (Job[i].Weight>Job[temp_i].Weight) {
              temp_awal=bij; temp_i=i;
              temp_j=queue[i]; }
            else   if (Job[i].Weight==Job[temp_i].Weight) {
              if (Operation[i][queue[i]].Tardiness > Operation[temp_i][temp_j].
                Tardiness)
              { temp_awal=bij; temp_i=i;
              temp_j=queue[i]; }
              else if (Operation[i][queue[i]].Tardiness ==Operation[temp_i]
                [temp_j].Tardiness)
```

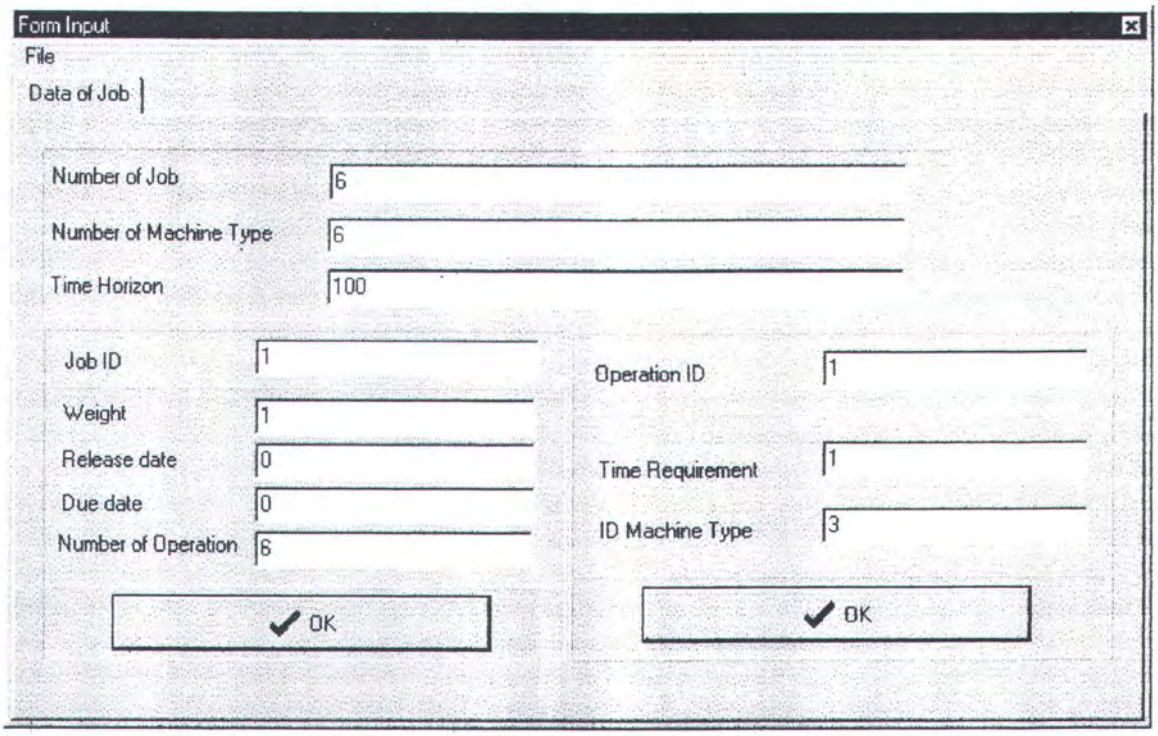
```
            {
             if (Operation[i][queue[i]].CompleteTimeOp<Operation[temp_i]
               [temp_j].CompleteTimeOp)
             { temp_awal=bij; temp_i=i;
               temp_j=queue[i]; }
             else if (Operation[i][queue[i]].CompleteTimeOp==Operation
                [temp_i][temp_j].CompleteTimeOp)
             { if (Job[i].DueTime < Job[temp_i].DueTime)
               { temp_awal=bij; temp_i=i;
                 temp_j=queue[i]; }
            } } } } } } } } }

      if ( status_temp==true)
      {
       while (Machine[h].index[index].IDJob!=0) index++;
       Machine[h].index[index].IDJob=temp_i;
       Machine[h].index[index].IDOp=temp_j;
       if (temp_j == Job[temp_i].NumOperation) queue[temp_i]=0;
       else  ++queue[temp_i];
       temp_sum++;
       status_temp=false;
      } } } }


void LRProcess::feasible(int sum_op, bool *change)
{
 while ( temp_sum != sum_op ) {
   for (int h=1;h<=NumTypeMachine;h++){
     index=0; temp_b=0; temp_bl=0; *change = false;
     while ( Machine[h].index[index].IDJob != 0 ) {
       x=Machine[h].index[index].IDJob;
       y=Machine[h].index[index].IDOp;
       if ( index != 0 ) {
          a = Machine[h].index[index-1].IDJob;
          b = Machine[h].index[index-1].IDOp;
          temp_b = Operation[a][b].CompleteTimeOp+1;
          if (y-1!=0) temp_bl = Operation[x][y-1].CompleteTimeOp+1;
          else  temp_bl=0;  }
       else
       {  temp_b = 0;
          if (y-1!=0)  temp_bl = Operation[x][y-1].CompleteTimeOp+1;
          else temp_bl= 0; }
       if (temp_b > Operation[x][y].BeginTimeOp) {
          Operation[x][y].BeginTimeOp = temp_b;
          Operation[x][y].CompleteTimeOp=Operation[x][y].
             BeginTimeOp+Operation[x][y].TimeReqOp-1;
          *change=true;  }
       if (temp_bl > Operation[x][y].BeginTimeOp) {
          Operation[x][y].BeginTimeOp = temp_bl;
          Operation[x][y].CompleteTimeOp=Operation[x][y].BeginTimeOp
            +Operation[x][y].TimeReqOp-1;
          *change=true; }
       if (y==1)  Job[x].BeginTime=Operation[x][y].BeginTimeOp;
```

```
if (y== Job[x].NumOperation) {
  Job[x].CompleteTime=Operation[x][y].CompleteTimeOp;
  int temp_T=0;
  temp_T=(Job[x].CompleteTime-Job[x].DueTime)-Job[x].ReleaseTime
  if (temp_T<0) Job[x].Tardiness=0;
  else Job[x].Tardiness=temp_T; }
 temp_sum++; index++;
} } } }
```

4.2.3 IMPLEMENTASI ANTAR MUKA

Seperti yang telah dijjelaskan pada rancangan antar muka, form utama ada dua macam, yaitu frmInput dan frmSchedule.

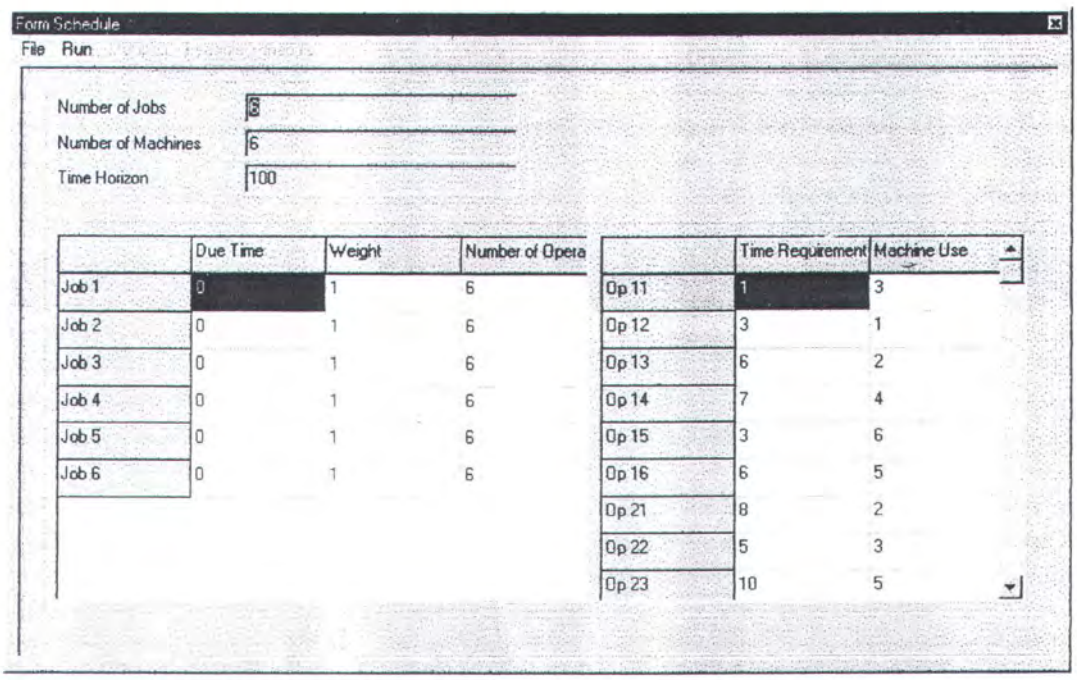
Tampilan antar muka untuk pemasukan data, dalam sistem ini, dapat dilihat seperti gambar berikut ini.



Gambar 4-7 Form pemasukan data

Pada form di atas, data – data yang perlu dimasukkan adalah jumlah pekerjaan, jumlah mesin, waktu maksimum, bobot tiap pekerjaan, waktu kedatangan suatu pekerjaan, batas waktu pengerjaan, jumlah operasi yang berlangsung tiap pekerjaan, waktu yang dibutuhkan oleh sebuah operasi yang bekerja pada suatu mesin tertentu. Setelah semua data dimasukkan, data disimpan pada suatu file (telah dijelaskan pada sub bab 4.2.2.1).

Tampilan antar muka untuk penyelesaian persamaan relaksasi lagrangian dan penyusunan jadwal adalah seperti berikut ini.

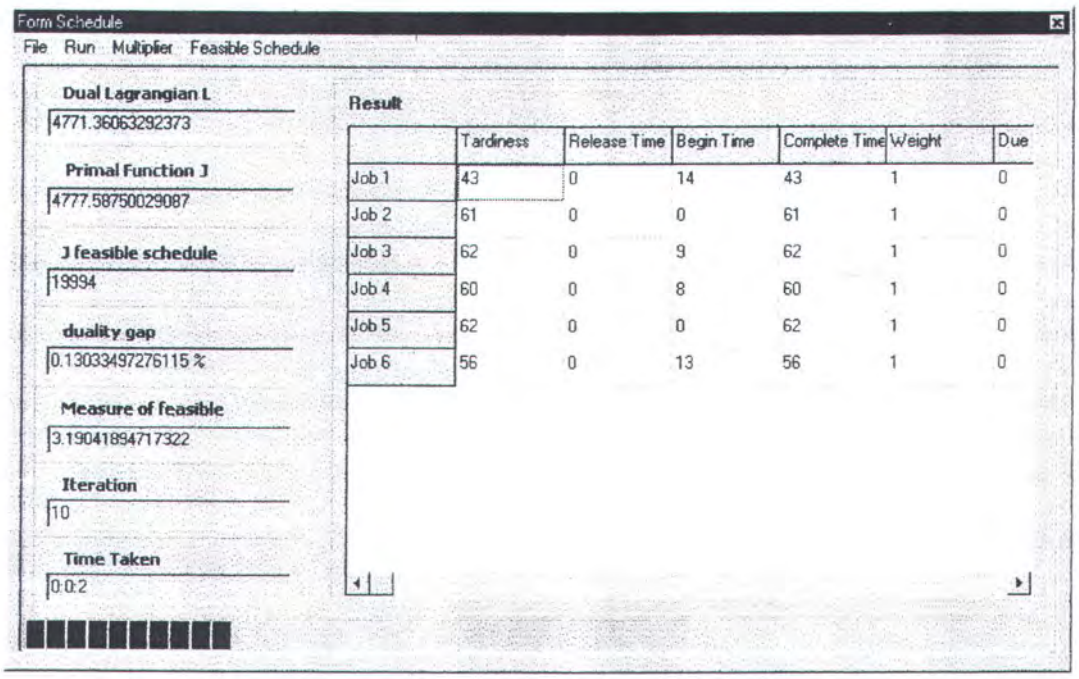


Gambar 4-8 Tampilan awal form frmSchedule

Form di atas, menampilkan data – data yang akan dijadwalkan (diambil dari file). Tabel sebelah kiri menjelaskan data –data yang

berhubungan dengan pekerjaan dan tabel sebelah kanan menjelaskan data – data yang berhubungan dengan operasi dari tiap – tiap pekerjaan.

Tampilan akhir setelah proses penyelesaian persamaan relaksasi lagrange dilakukan dan penyusunan jadwal yang fisibel adalah seperti berikut ini.



Gambar 4-9 Tampilan akhir

Form di atas merupakan hasil akhir dari proses perhitungan persamaan relaksasi lagrangian dan penyusunan jadwal. Nilai yang ditampilkan adalah nilai dual lagrange, nilai fungsi primal, nilai fungsi obyektif, nilai duality gap, jumlah iterasi yang dilakukan dan waktu yang dibutuhkan untuk proses penyelesaian persamaan relaksasi lagrangian.

BAB V

HASIL UJI COBA DAN EVALUASI

BAB V

HASIL UJI COBA DAN EVALUASI

Perangkat lunak yang akan dibangun pada tugas akhir ini dirancang untuk membuat suatu jadwal yang optimal, disertai dengan ukuran dari suatu jadwal. Relaksasi lagrangian diterapkan pada persamaan yang merupakan suatu fungsi obyektif yang meminimumkan. Karena fungsi obyektifnya merupakan fungsi yang konfeks, maka dapat ditentukan fungsi dual yang merupakan fungsi konkaf.

Dalam bab ini akan dijelaskan lebih lanjut tentang uji coba yang dilakukan terhadap perangkat lunak ini, serta pembahasan dan evaluasi yang diberikan berdasarkan hasil uji coba tersebut.

5.1 LINGKUNGAN UJI COBA

Spesifikasi perangkat lunak yang digunakan dalam uji coba perangkat lunak ini adalah sebagai berikut :

- Processor Intel 233 MMX
- RAM 80 MB
- Hardisk 3,2 GB

Spesifikasi lain yang juga digunakan dan turut menunjang uji coba ini dan pembuatan perangkat lunak ini adalah :

- Sistem operasi Windows 98
- Borland C++ Builder 3.0
- Perangkat lunak Lekin (Sebagai pembanding)

5.2 PELAKSANAAN UJI COBA

Pelaksanaan uji coba dilakukan pada sistem perangkat lunak ini dengan memperhatikan jadwal yang dihasilkan, nilai – nilai fungsi primal dan fungsi dual, waktu komputasi dan nilai – nilai pengali lagrange.

5.2.1 PEMASUKAN DATA

Data – data dimasukkan melalui form frmInput. Data – data yang diperlukan antara lain :

- Jumlah pekerjaan
- Jumlah mesin
- Waktu maksimum
- Waktu awal pekerjaan datang
- Batas waktu tiap pekerjaan
- Jumlah operasi tiap pekerjaan
- Waktu proses tiap operasi
- Mesin yang digunakan tiap operasi

Data yang digunakan telah disediakan pada perangkat lunak Lekin. Untuk uji coba ini ada 3 data yang digunakan yaitu 3x3 (jumlah pekerjaan 3 dengan jumlah mesin 3), 6x6, dan 10x10.

5.2.2 DATA UJI COBA

Pada bab ini, yang akan dijelaskan lebih terinci adalah data 6x6. Sesuai dengan sub bab 5.2.1, maka data – data yang diperlukan antara lain :

Tabel 5-1 Data input 6x6

ID pekerjaan	Jumlah operasi	Waktu awal	Bobot	Batas waktu	Waktu yang dibutuhkan
1	6	0	1	0	26
2	6	0	1	0	47
3	6	0	1	0	34
4	6	0	1	0	35
5	6	0	1	0	25
6	6	0	1	0	30

Data untuk tiap operasinya adalah :

ID Pekerjaan	ID Operasi	Waktu proses
1	1	1
	2	3
	3	6
	4	7
	5	3
	6	6
2	1	8
	2	5
	3	10
	4	10
	5	10
	6	4
3	1	5
	2	4
	3	8
	4	9
	5	1
	6	7

ID Pekerjaan	ID Operasi	Waktu proses
4	1	5
	2	5
	3	5
	4	3
	5	8
	6	9
5	1	9
	2	3
	3	5
	4	4
	5	3
	6	1
6	1	3
	2	3
	3	9
	4	10
	5	4
	6	1

Persoalan terdiri dari 6 pekerjaan. Berturut – turut persoalan diselesaikan dengan 6 buah mesin dengan waktu maksimal proses adalah 100. Tiap pekerjaan mempunyai 6 operasi dengan tiap operasi bekerja pada 1 jenis mesin. Bobot (w) tiap pekerjaan sama yaitu 1 dengan waktu seharusnya selesai adalah 0 yaitu tiap pekerjaan tidak mempunyai batas waktu pengerjaan. Waktu proses tiap operasi pada suatu mesin telah ditentukan.

5.2.3 PENYELESAIAN PERSAMAAN RELAKSASI LAGRANGIAN

Uji coba penyelesaian persamaan relaksasi lagrangian dilakukan dengan menekan tombol Run|Solving Dual Lagrangian pada frmSchedule. Pada uji coba ini dihasilkan nilai dual lagrangian, nilai fungsi primal, nilai duality gap, jumlah iterasi yang dilakukan dan waktu komputasi penyelesaian persamaan relaksasi lagrangian ini.

Pada proses ini, jumlah pekerjaan yang akan dijadwalkan, jumlah operasi tiap pekerjaan, jumlah mesin yang diperlukan dan yang tersedia sangat mempengaruhi proses perhitungan dan lamanya proses yang dijalankan. Semakin banyak pekerjaan yang akan dijadwalkan maka waktu komputasi untuk perhitungan akan semakin lama. Lamanya proses komputasi karena adanya proses perulangan pada data yang banyak dan tergantung dengan perolehan solusi yang tetap selama iterasi tersebut. Selain itu, lamanya waktu komputasi juga tergantung dengan spesifikasi hardware yang digunakan.

Seperti telah dijelaskan sebelumnya, suatu fungsi primal yang konfeks akan mempunyai fungsi dual yang konkaf. Jika fungsi primal dan fungsi dual mempunyai selisih (yang disebut duality gap) maka disebut *weak duality*. Nilai fungsi primal selalu lebih besar dari nilai fungsi dual. Dan sebaliknya, jika tidak ada selisih antara fungsi primal dan dual (nilai fungsi primal = nilai fungsi dual) maka disebut *strong duality*.

Nilai pengali Lagrange π diinisialisasikan sama dengan 0. Pengali lagrange π merupakan suatu nilai untuk kapasitas mesin. Dari rumusan

pembaruan suatu pengali lagrange π (rumus (3-21)) dapat dilihat bahwa suatu nilai π dipengaruhi oleh jumlah operasi (i,j) yang bekerja pada jenis mesin pada waktu k. Jika jumlah tersebut lebih besar dari jumlah mesin h yang tersedia pada waktu k (dalam tugas akhir ini, telah dibatasi bahwa job shop yang digunakan adalah klasik job shop yaitu job shop yang hanya memiliki 1 mesin untuk tiap jenis mesin) maka nilai π akan semakin besar. Sehingga semakin banyak operasi yang bekerja pada suatu mesin pada waktu tertentu maka akan semakin besar nilai π - nya.

Pengali lagrange λ merupakan nilai dari operasi yang berurutan (operasi (i,j) dengan operasi (i,l)) yang mempunyai waktu *overlapping* atau suatu nilai yang dapat mengurangi waktu proses dan diinisialiasikan dengan nilai 0,1 karena nilai 0 menunjukkan bukan operasi yang sesudahnya..

5.2.4 PENYUSUNAN JADWAL

Hasil dari perhitungan fungsi dual bukanlah suatu hasil yang fisibel (optimal dan efisien) karena pada perhitungan tersebut terdapat batasan kapasitas mesin pada waktu tertentu dan terdapat operasi yang *overlapping*. Sehingga perlu disusun suatu jadwal yang fisibel.

Seperti yang telah dijelaskan pada bab sebelumnya, penyusunan jadwal berdasarkan nilai waktu awal tiap operasi dengan memperhatikan operasi sebelumnya dan sesudahnya. Jika ada beberapa operasi mempunyai waktu awal yang sama dan bekerja pada mesin yang sama

pula maka prioritas berdasarkan bobot tardiness tiap pekerjaan, suatu pekerjaan yang mempunyai nilai bobot terdiness terbesar yang akan didahulukan dan pekerjaan yang lain ditunda terlebih dahulu. Adanya penundaan ini akan menyebabkan nilai tardiness pada pekerjaan tersebut meningkat sehingga dapat meningkatkan nilai fungsi obyektifnya.

Hasil jadwal dengan teknik ini akan dibandingkan dengan penjadwalan dengan menggunakan program Lekin. Pada perangkat lunak Lekin, algoritma yang digunakan adalah cara heuristik dengan prosedur yang umum total weighted tardiness dan shifting bottleneck total weighted tardiness.

5.3 HASIL UJI COBA DAN EVALUASI

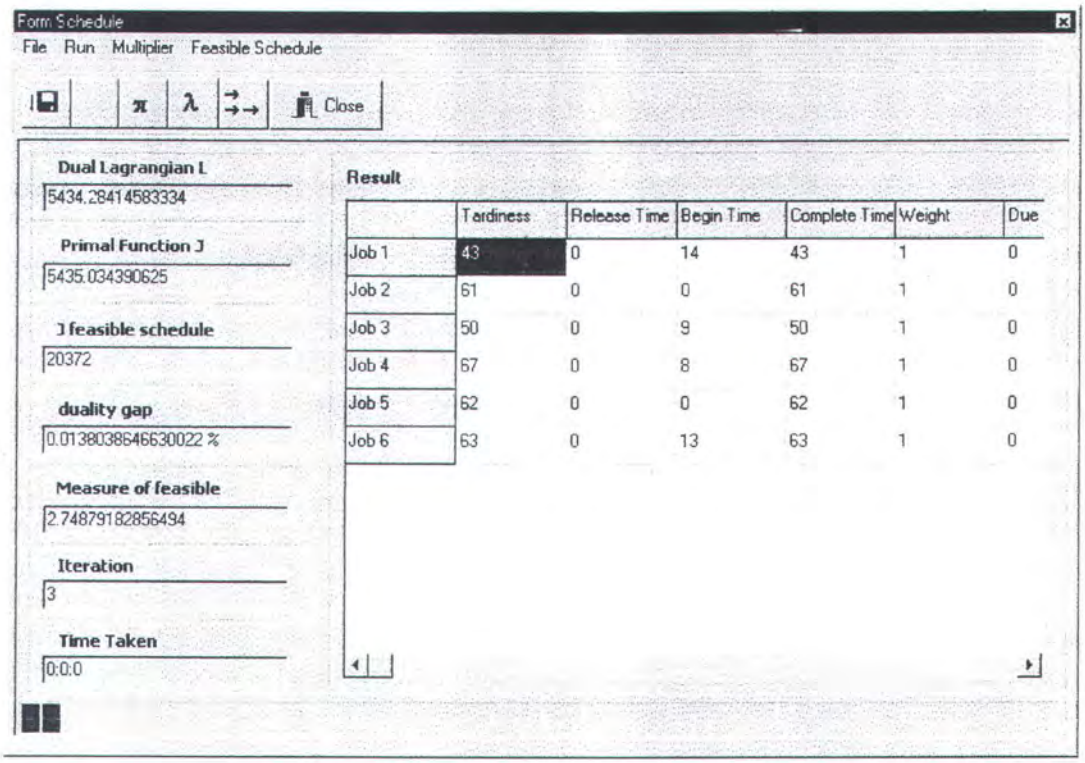
Hasil uji coba penyelesaian persamaan relaksasi lagrangian untuk data dari perangkat lunak Lekin adalah 3x3, 6x6, dan 10x10 serta 18x10 dan 25x10 yang telah diinputkan dapat dilihat pada tabel berikut ini.

Tabel 5-2 Tabel hasil berdasarkan jumlah pekerjaan dan mesin

No	Jumlah pekerjaan	Jumlah mesin	Nilai fungsi dual (L)	Nilai fungsi primal (J)	Duality Gap	Jumlah iterasi	Waktu (detik)
1.	3	3	0.02375	0.45	94.7%	2	0:0:0
2.	6	6	5434.28	5435.034	0.013%	3	0:0:1
3.	10	10	93.211	96.613	3.52%	4	0:0:26
4.	18	10	2.316	4.346	46.69%	2	0:0:2
5.	25	10	1.785	3.2147	44.47%	2	0:0:4

Untuk data 3x3, mempunyai nilai fungsi dual adalah 0,02375 dan fungsi primal 0,45. Nilai duality gap adalah selisih antara fungsi primal dan fungsi dual. Jumlah iterasi adalah 2 karena proses telah mengalami konvergen (telah dijelaskan pada sub bab 3.2.2). Waktu yang diperlukan untuk proses ini adalah 0 (kurang dari 1 detik). Untuk data – data yang lain, keterangan sama dengan data 3x3.

Sesuai dengan data uji coba yang digunakan (sub bab 5.2.2), hasil dari perangkat lunak pada tugas akhir ini dapat dilihat pada gambar 5-1 yang menampilkan hasil dari perhitungan persamaan relaksasi Lagrangian serta jadwal yang dihasilkan. Gambar 5-2 menampilkan pengali Lagrange yang π dan λ yang dihasilkan selama proses. Dan gambar 5-3 menampilkan operasi – operasi yang dikerjakan pada tiap – tiap mesin disertai dengan waktu awal proses tiap operasi.

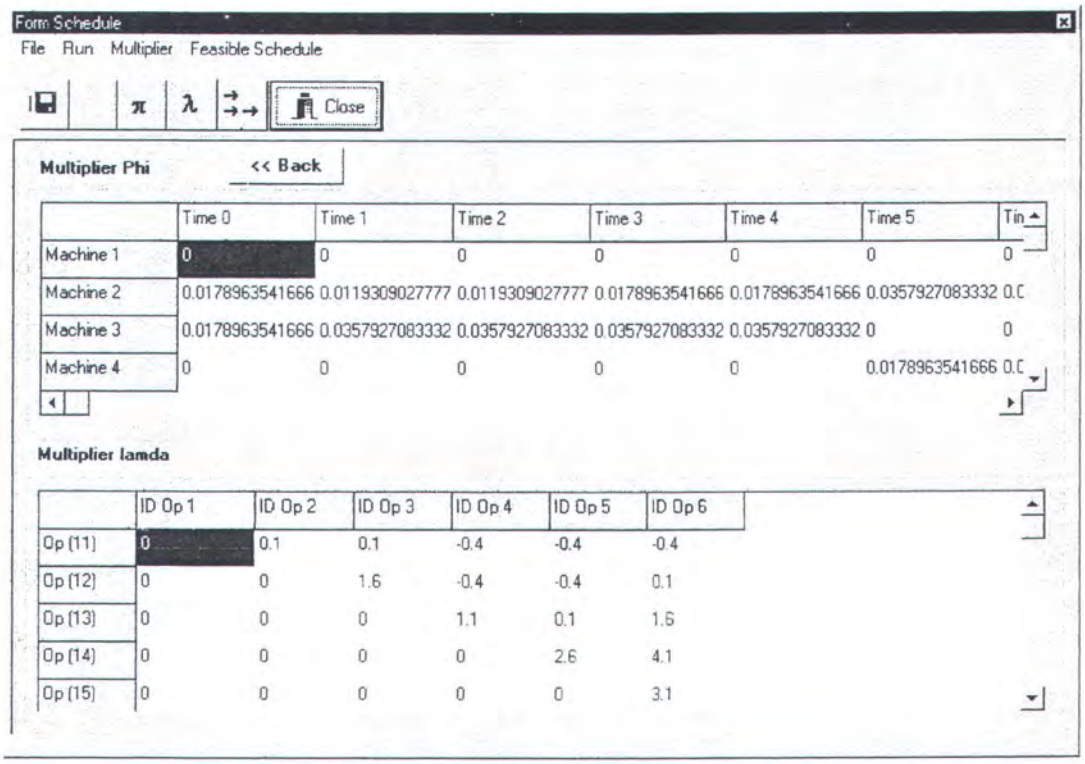


Gambar 5-1 Gambar form penyelesaian persamaan relaksasi lagrangian data 6x6

Data yang digunakan adalah 6x6. Iterasi yang dilakukan sebanyak 3 karena mengalami konvergen (bila adalah $\bar{b}_{ij} = b_{ij}$ untuk semua operasi). Nilai dual yang dicapai adalah 5434,28 dan nilai primalnya adalah 5435,03. Duality gap yang merupakan selisih antara fungsi dual dan primal, nilainya adalah 0,0138%. Waktu yang dibutuhkan untuk menyelesaikan persamaan relaksasi lagrangian dengan jumlah iterasi 3 adalah 1 detik (tergantung dengan lingkungan uji coba yang ada). Tabel sebelah kanan merupakan hasil akhir dari penjadwalan. Pekerjaan 1 mempunyai nilai keterlambatan (*tardiness*) 43, waktu kedatangan awal (*release time*) 0, waktu awal proses operasi 1 (*begin time*) 14, waktu penyelesaiannya (*complete time*

) adalah 43, bobot (*weight*) adalah 1 dan waktu seharusnya selesai (*due time*) 0. Demikian juga untuk pekerjaan – pekerjaan yang lain.

Nilai pengali lagrange yang dihasilkan merupakan hasil pembaruan tiap iterasi. Hasil dari pengali lagrange tersebut dapat dilihat pada gambar berikut ini :



	Time 0	Time 1	Time 2	Time 3	Time 4	Time 5	Tin
Machine 1	0	0	0	0	0	0	0
Machine 2	0.0178963541666	0.0119309027777	0.0119309027777	0.0178963541666	0.0178963541666	0.0357927083332	0.0
Machine 3	0.0178963541666	0.0357927083332	0.0357927083332	0.0357927083332	0.0357927083332	0	0
Machine 4	0	0	0	0	0	0.0178963541666	0.0

	ID Op 1	ID Op 2	ID Op 3	ID Op 4	ID Op 5	ID Op 6
Op (11)	0	0.1	0.1	-0.4	-0.4	-0.4
Op (12)	0	0	1.6	-0.4	-0.4	0.1
Op (13)	0	0	0	1.1	0.1	1.6
Op (14)	0	0	0	0	2.6	4.1
Op (15)	0	0	0	0	0	3.1

Gambar 5-2 Hasil dari pengali lagrange untuk data 6x6

Untuk lebih jelasnya, lihat tabel berikut ini.

Tabel 5-3 Tabel nilai π pada job shop 6x6

H/k	0	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0	0.0357
2	0.0178	0.0119	0.0119	0.0178	0.0178	0.0357	0.0357	0	0
3	0.0178	0.0357	0.0357	0.0357	0.0357	0	0	0	0.0178

H/k	0	1	2	3	4	5	6	7	8
4	0	0	0	0	0	0.0178	0.0357	0.0357	0.0357
5	0	0	0	0	0	0	0	0.0357	0.0357
6	0	0	0	0	0	0.0178	0.0178	0.0119	0.0119

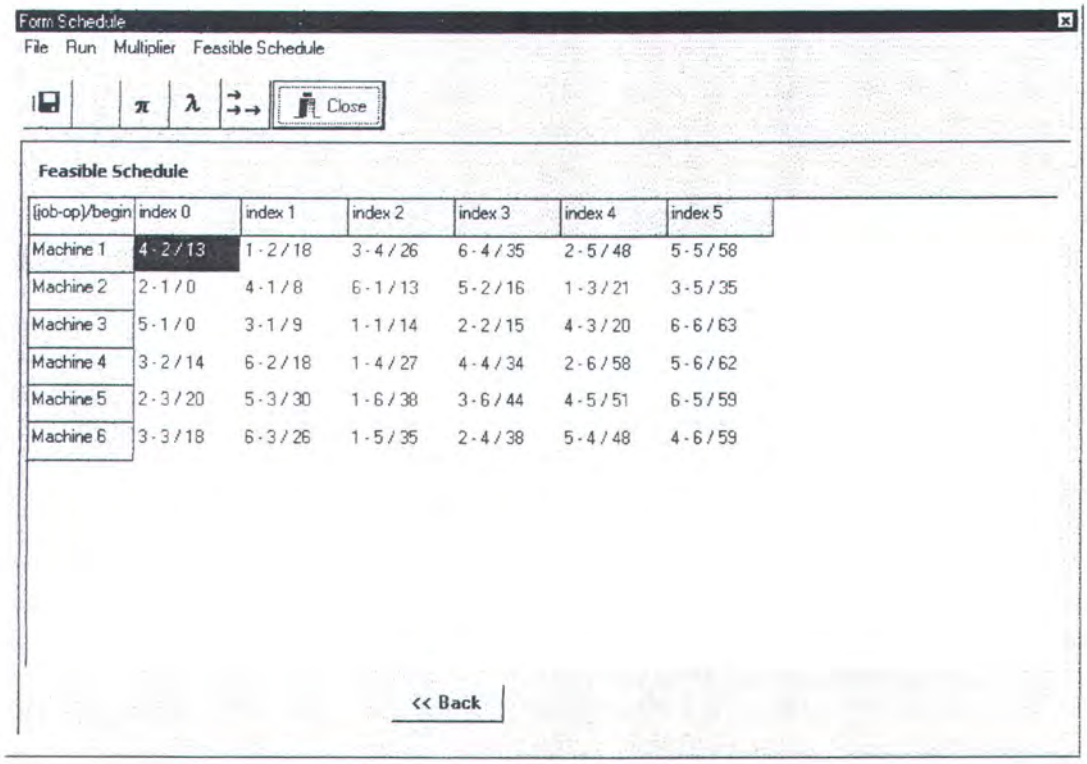
Pada tabel 5-3 di atas, nilai k hanya sampai 8 (Data selengkapnya lihat lampiran A tabel 2). Pada tabel dapat dilihat, bahwa nilai π untuk k=0 dan h=2 adalah 0,0178 yaitu terdapat lebih dari 1 operasi yang bekerja pada mesin 1 dengan waktu 0. Dan nilai π adalah 0, yaitu tidak ada operasi yang bekerja pada mesin 1 dengan waktu 0.

Dari nilai pengali λ dapat dilihat bahwa beberapa operasi yang berurutan telah mengalami *overlapping* dengan operasi – operasi yang lain. Nilai 0 menunjukkan bahwa operasi tersebut bukan operasi setelah operasi tersebut. Dan jika nilai pengali λ sama dengan inisialisasi maka tidak ada operasi yang *overlapping*. Untuk lebih jelasnya, lihat tabel berikut ini untuk operasi – operasi pada pekerjaan 1 (Data selengkapnya lihat lampiran A tabel 1).

Tabel 5-4 Tabel nilai pengali λ pekerjaan 1 untuk job shop 6x6

Op	1	2	3	4	5	6
1-1	0	0.1	0.1	-0.4	-0.4	-0.4
1-2	0	0	1.6	-0.4	-0.4	0.1
1-3	0	0	0	1.1	0.1	1.6
1-4	0	0	0	0	2.6	4.1
1-5	0	0	0	0	0	3.1
1-6	0	0	0	0	0	0

Hasil dari penyelesaian persamaan relaksasi lagrangian tidak menghasilkan jadwal yang fisibel karena masih adanya operasi – operasi yang *overlapping* dan menggunakan mesin yang sama pada saat yang bersamaan. Tahap akhir dari perangkat lunak ini adalah penyusunan jadwal tersebut (seperti telah disebutkan pada bab sebelumnya). Hasil akhir dapat dilihat pada gambar berikut ini.



Gambar 5-3 Hasil jadwal untuk data 6x6

Index menunjukkan urutan pengerjaan suatu operasi pada mesin tertentu. Index 0 pada mesin 1 adalah 4-2/13 yang artinya mesin 1 pertama kali mengerjakan pekerjaan 4 operasi 2 dengan waktu awal proses adalah 13.

Hasil dari metode di atas dibandingkan dengan perangkat lunak Lekin yang menggunakan prosedur yang umum total weighted tardiness dan shifting bottleneck total weighted tardiness, dapat dilihat pada tabel – tabel berikut ini.

Data awal untuk proses perbandingan ini adalah data 6x6 (pada sub bab 5.2.1). Dengan waktu awal kedatangan semua pekerjaan adalah 0, tiap pekerjaan mempunyai prioritas yang sama. Nilai batas waktu 0 berarti tidak ada batas waktu untuk semua pekerjaan. Hasil dari perbandingan adalah sebagai berikut :

Tabel 5-5 Tabel waktu awal proses untuk 3 metode

ID pekerjaan	Waktu awal proses		
	LR	Lekin	
		Shifting bottleneck	General Subroutine
		Sum (wT)	Sum (wT)
1	14	0	5
2	0	19	18
3	9	18	0
4	8	3	3
5	0	1	6
6	13	0	0

Waktu awal proses merupakan waktu awal suatu pekerjaan dimulai (waktu awal operasi pertama pada pekerjaan tersebut yang bekerja pada suatu mesin tertentu dimulai). Untuk pekerjaan 1, waktu awal proses dengan metode relaksasi Lagrangian (LR) adalah 14, shifting bottleneck 0 dan general subroutine adalah 5 dan seterusnya. Waktu awal proses ini tidak perlu diperbandingkan di antara 3 metode ini karena tabel ini hanya

merupakan keterangan untuk memperjelas waktu awal proses tiap pekerjaan berdasarkan ketiga metode itu.

Tabel 5-6 Tabel perbandingan waktu penyelesaian pekerjaan

ID pekerjaan	Waktu penyelesaian		
	LR	Lekin	
		Shifting bottleneck	General Subroutine
		Sum (wT)	Sum (wT)
1	43	39	41
2	61	73	82
3	50	65	48
4	67	40	42
5	62	29	31
6	63	34	36

Dari tabel di atas, waktu penyelesaian maksimum dengan metode relaksasi lagrangian (LR) adalah 67. Jika dibandingkan dengan perangkat lunak Lekin, shifting bottleneck total weighted tardiness adalah 73, dan prosedur umum total weighted tardiness adalah 82. Dari nilai itu dapat dilihat bahwa dengan metode LR, waktu penyelesaiannya lebih baik dibandingkan dengan 2 algoritma yang lain.

Tabel 5-7 Tabel perbandingan tardiness

ID pekerjaan	Nilai tardiness		
	LR	Lekin	
		Shifting bottleneck	General Subroutine
		Sum (wT)	Sum (wT)
1	43	39	41
2	61	73	82
3	50	65	48
4	67	40	42
5	62	29	31
6	63	34	36

Nilai tardiness dilihat dari waktu penyelesaian dan batas waktu. Dari perbandingan metode LR dengan ketiga algoritma pada perangkat lunak Lekin dapat dilihat bahwa pada pekerjaan 1, nilai tardiness dengan metode LR lebih besar daripada sum(wT). Akan tetapi pada pekerjaan 2, nilai tardiness dengan metode LR lebih kecil daripada kedua algoritma dari Lekin. Hal ini terjadi karena jika suatu pekerjaan didahulukan maka akan menunda pekerjaan lain.

Hasil penjadwalan metode LR dan kedua algoritma perangkat lunak Lekin dapat digambarkan dengan Gantt Chart pada lampiran A gambar 1 untuk metode relaksasi Lagrangian, gambar 2 untuk prosedur umum total weighted tardiness dan gambar 3 untuk metode shifting bottleneck.

BAB VI

PENUTUP

BAB VI
PENUTUP

Pada bab ini diuraikan beberapa hal yang dapat disimpulkan dari penggunaan teknik relaksasi lagrangian pada penjadwalan sistem manufaktur job shop dalam tugas akhir ini. Dan juga disertai kemungkinan pengembangan yang lebih lanjut dari perangkat lunak ini.

6.1 KESIMPULAN

Dari tugas akhir yang telah dikerjakan ini dapat ditarik beberapa kesimpulan mengenai penjadwalan dengan teknik relaksasi Lagrangian, yaitu antara lain :

1. Teknik relaksasi Lagrangian dapat digunakan untuk menyelesaikan permasalahan penjadwalan sistem manufaktur job shop yang mempunyai fungsi obyektif meminimumkan jumlah keterlambatan suatu pekerjaan, dengan memecah permasalahan menjadi sub permasalahan penjadwalan yang lebih sederhana yaitu masing – masing pekerjaan yang akan dijadwalkan. Batasan yang mempengaruhi fungsi obyektif tersebut adalah jumlah mesin yang tersedia dan urutan operasi – operasi pada tiap mesin.

2. Berdasarkan uji coba yang telah dibuat pada tugas akhir ini, suatu jadwal mencapai nilai optimal jika sesuai dengan fungsi obyektif yang digunakan, yaitu nilai tardiness (keterlambatan) yang minimum.
3. Dari perbandingan dengan beberapa algoritma pada perangkat lunak Lekin yang mempunyai fungsi obyektif yang sama, dapat dibuktikan bahwa perangkat lunak metode relaksasi Lagrangian menghasilkan jadwal yang lebih optimal.

6.2 KEMUNGKINAN PENGEMBANGAN LEBIH LANJUT

Perangkat lunak yang dibuat pada tugas akhir ini merupakan awal dari teknik – teknik selanjutnya khusus untuk teknik relaksasi Lagrangian terutama dalam penyelesaiannya. Masih ada beberapa cara dalam menyelesaikan persamaan relaksasi Lagrangian ini, misalnya penyelesaian persamaan dual Lagrangian dengan menggunakan metode surrogate gradient, penggabungan teknik relaksasi Lagrangian dengan dynamic programming.

Perlunya peningkatan efisiensi penggunaan waktu yang digunakan karena pemecahan persamaan dual Lagrangian dengan menggunakan metode subgradient membutuhkan waktu yang banyak khususnya jika jumlah pekerjaan lebih besar atau jika nilai waktu horizonnya lebih lama.

Teknik relaksasi Lagrangian juga dapat digunakan untuk menyelesaikan penjadwalan job shop yang mempunyai waktu proses yang tidak pasti, yaitu waktu proses berdasarkan mesin – mesin yang akan digunakan.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. [Art-96] Artiba A., Smaghraby E, "*The Planning and Scheduling of Production System*", Chapman & Hall, 1996
2. [Epp-93] Eppend G.D., Gould F.J., Schmidth C.P., "*Introduction Management Science*", Prentice - Hall International, Inc., 1993
3. [Gen-97] Gen Mitsuo, Cheng Runwei, " *Genetic Algorithms and Engineering Design* ", John Wiley & Sons, Inc., 1997
4. [Luh-93] Luh Peter B.,Hoitomt Debra J., "*Scheduling of Manufacturing System Using the Lagrangian Relaxation Technique*", IEEE Transactions on Automatic Control, Vol. 38, No. 7, Juli, 1993.
5. [Nas-96] Nash Stephen, Sofer Ariela, "*Linear and Nonlinear Programming*", Mc Graw-Hill, 1996.
6. [Pin-95] Pinedo Michael,"*Scheduling, Theory, Algortihms, and Systems*", Prentice Hall, 1995
7. [Wan-97] Wang Jihua, Luh P.B, Zhao Xing, "*An Optimization – Based Algorithm for Job Shop Scheduling*", Department of Electrical and Systems Engineering, University of Connecticut, 1997
8. [Zha-99] Zhao X., Luh P.B, Wang J., "*Surrogate Gradient Algorithm for Lagrangian Relaxation*", Journal of Optimization Theory and Applications, Vol. 100, No. 3, Mar.1999

LAMPIRAN

LAMPIRAN A
TABEL PENGALI LAGRANGE DAN GANTT CHART

Pada lampiran ini terdapat tabel pengali Lagrange λ dan π yang dihasilkan pada proses penyelesaian persamaan dengan relaksasi Lagrangian pada iterasi terakhir. Selain itu, juga terdapat Gantt Chart dari hasil penjadwalan dengan teknik relaksasi Lagrangian dan algoritma – algoritma yang terdapat pada perangkat lunak Lekin yaitu shifting bottleneck sum (wT) dan general subroutine sum (wT) yang digunakan sebagai pembanding.

Tabel 1
Tabel pengali lagrange λ data 6x6

Operasi	1	2	3	4	5	6
1-1	0	0.1	0.1	-0.4	-0.4	-0.4
1-2	0	0	1.6	-0.4	-0.4	0.1
1-3	0	0	0	1.1	0.1	1.6
1-4	0	0	0	0	2.6	4.1
1-5	0	0	0	0	0	3.1
1-6	0	0	0	0	0	0
2-1	0	0.1	-0.4	-0.4	-0.4	-0.4
2-2	0	0	1.1	-0.4	-0.4	0.1
2-3	0	0	0	1.6	1.6	4.1
2-4	0	0	0	0	5.1	7.6

Operasi	1	2	3	4	5	6
2-5	0	0	0	0	0	7.6
2-6	0	0	0	0	0	0
3-1	0	0.1	0.1	-0.4	-0.4	-0.4
3-2	0	0	2.1	0.6	-0.4	1.1
3-3	0	0	0	2.6	0.1	3.1
3-4	0	0	0	0	2.1	5.1
3-5	0	0	0	0	0	3.6
3-6	0	0	0	0	0	0
4-1	0	0.1	-0.4	-0.4	-0.4	0.1
4-2	0	0	1.1	-0.4	0.6	2.6
4-3	0	0	0	0.6	2.1	4.1
4-4	0	0	0	0	3.1	5.1
4-5	0	0	0	0	0	6.1
4-6	0	0	0	0	0	0
5-1	0	0.1	-0.4	-0.4	-0.4	0.1
5-2	0	0	1.1	-0.4	-0.4	1.6
5-3	0	0	0	1.1	1.1	3.1
5-4	0	0	0	0	2.1	4.1
5-5	0	0	0	0	0	3.6
5-6	0	0	0	0	0	0
6-1	0	.1	0.1	-0.4	-0.4	-0.4
6-2	0	0	1.6	-0.4	-0.4	-0.4
6-3	0	0	0	1.6	-0.4	1.1
6-4	0	0	0	0	3.1	4.6
6-5	0	0	0	0	0	3.6
6-6	0	0	0	0	0	0

Tabel 2

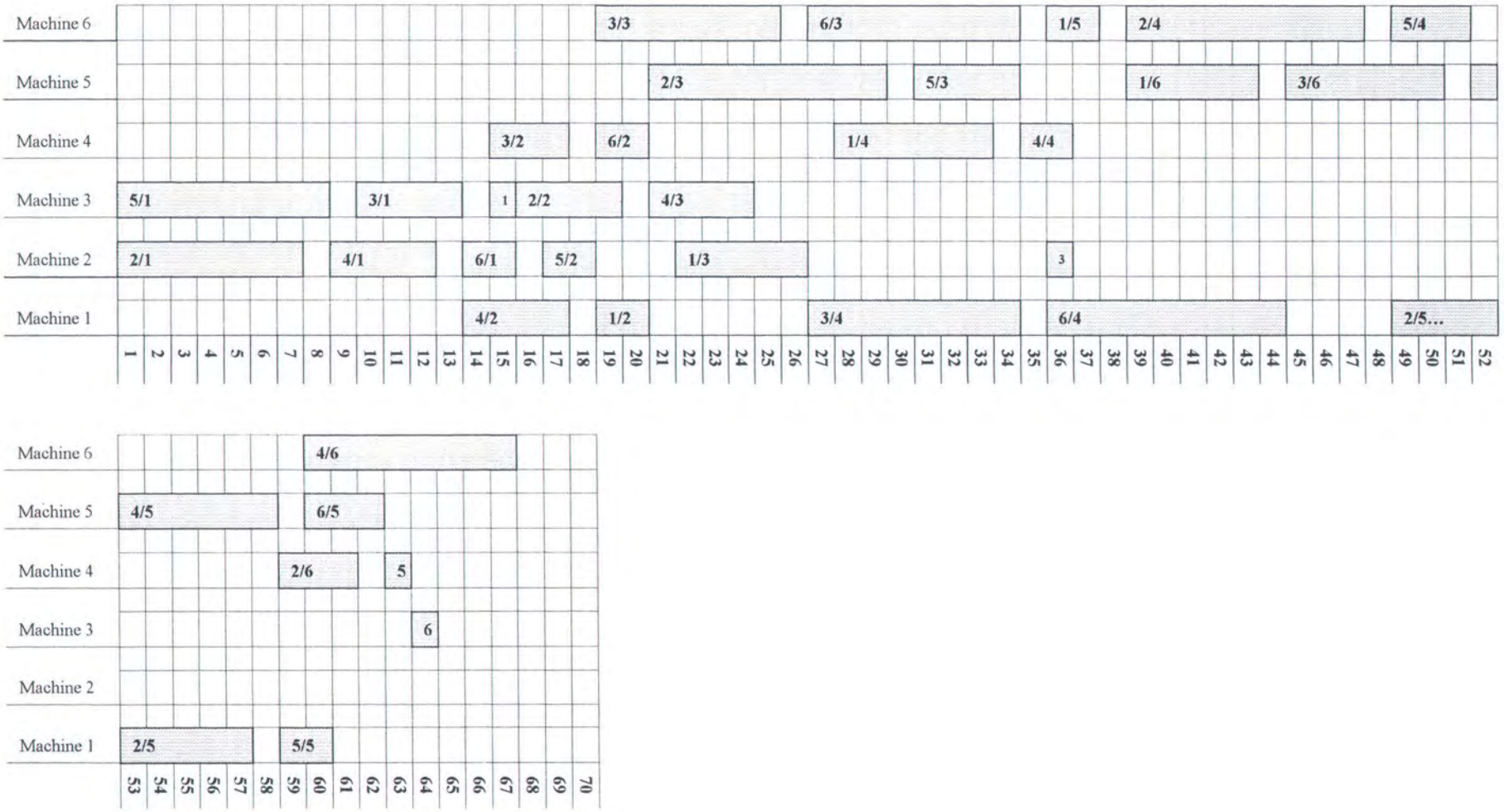
Tabel nilai lagrange π data 6x6 dengan waktu horison 100

h/k	0	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	27	28	29	30
1	0	0	0	0	0	0	0	0	0.04	0.02	0.04	0.04	0.04	0.02	0.03	0.02	0.04	0	0.03	0	0	0	0	0	0	0	0	0	0	0	0
2	0.02	0.01	0.01	0.02	0.02	0.04	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0.02	0.04	0.04	0.04	0.04	0	0	0	0.02	0.05	0.02	0.04	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0.02	0.04	0.04	0.04	0.04	0	0	0	0.04	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0.04	0.04	0.02	0.03	0.01	0.01	0.01	0.01	0.02	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0.02	0.02	0.01	0.01	0.01	0.02	0.02	0.04	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

h/k	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

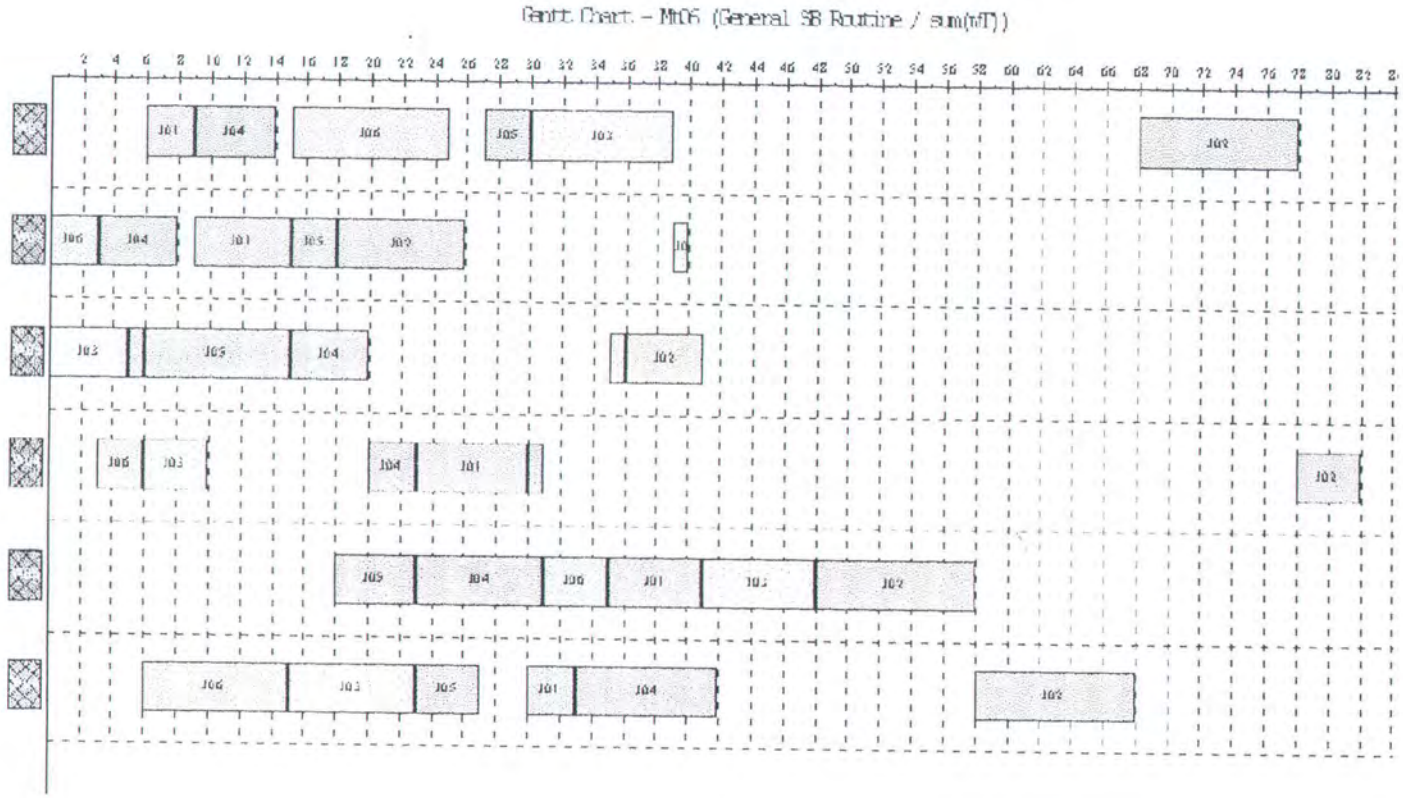
h/k	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 1
Gantt Chart data 6x6 metode relaksasi lagrangian



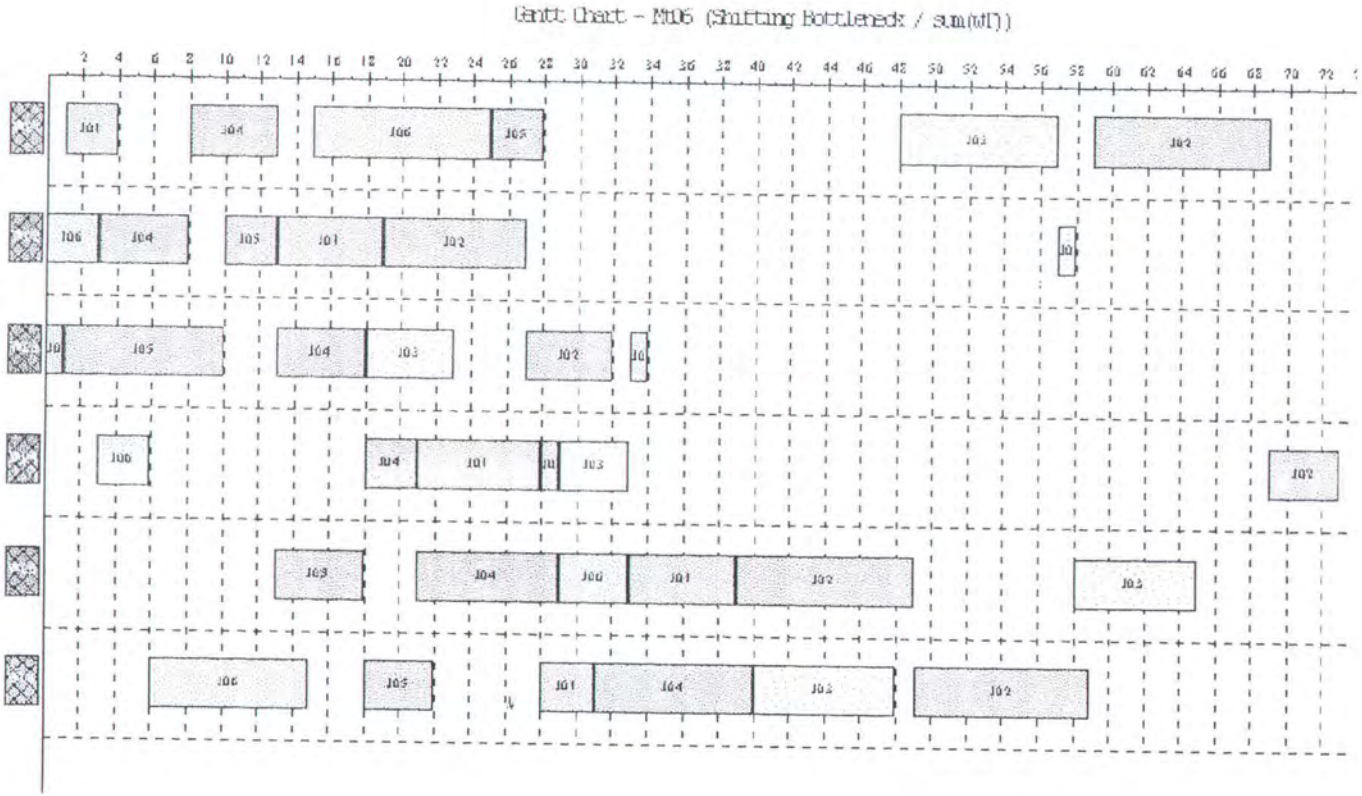
Gambar 2

Gantt Chart data 6x6 metode General Subroutine Sum(wT)



Gambar 3

Gantt Chart data 6x6 metode shifting bottleneck sum(wT)

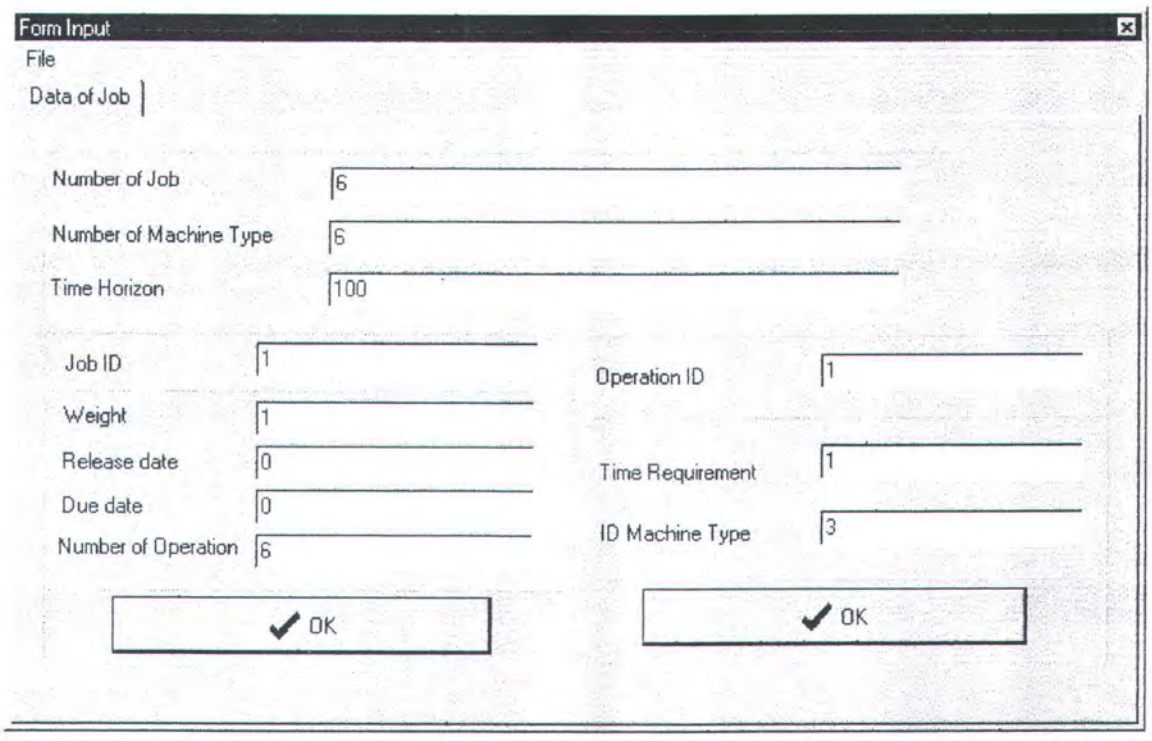


LAMPIRAN B

PETUNJUK PENGGUNAAN PERANGKAT LUNAK

Petunjuk penggunaan perangkat lunak ini terdiri dari petunjuk pemasukan data dan petunjuk penggunaan perangkat lunak untuk menyelesaikan persamaan relaksasi Lagrangian.

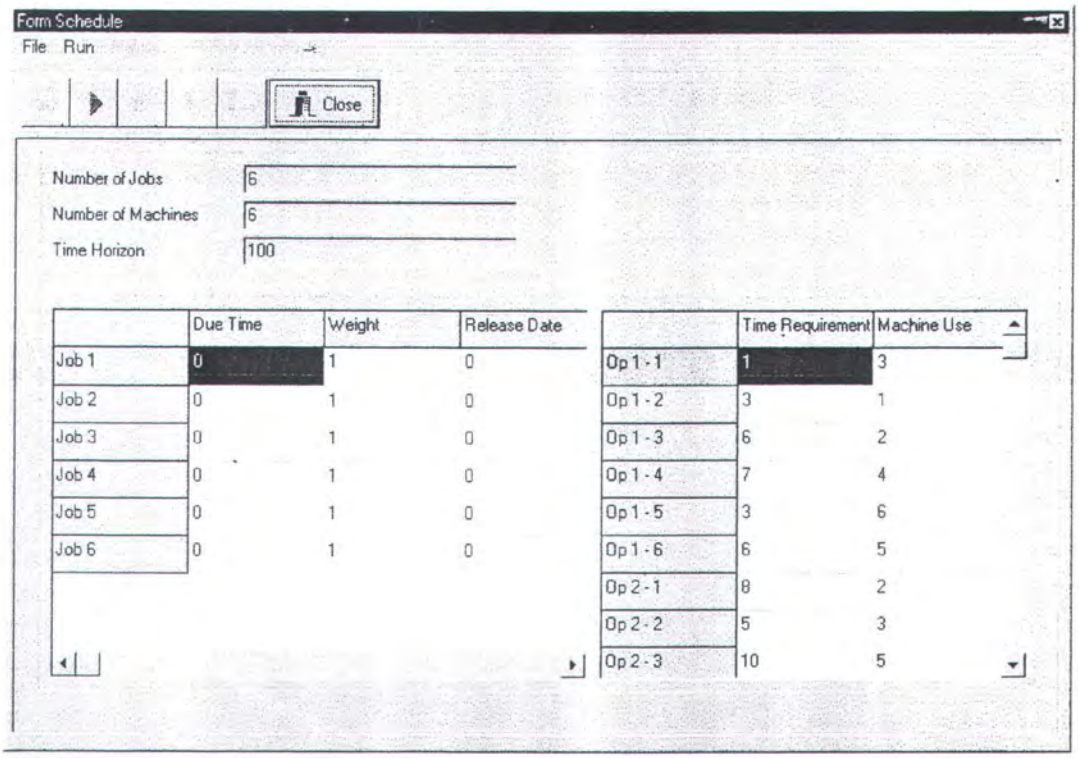
1. Petunjuk pemasukan data



- Untuk pemasukan data digunakan form frmInput (gambar di atas). Dari menu File dipilih New Data.
- Pertama kali, data yang diperlukan adalah jumlah pekerjaan, jumlah mesin dan waktu maksimum.

- Pengisian data yang berhubungan dengan pekerjaan (diulang sebanyak jumlah pekerjaan yang telah dimasukkan sebelumnya (sesuai dengan nilai *number of job*)).
- Pengisian data yang berhubungan dengan operasi (diulang sebanyak jumlah operasi yang telah dimasukkan (sesuai dengan nilai *number of operation*)).
- Setelah semua data selesai dimasukkan, keluar dari form dengan menekan tombol Exit pada menu File. Jika keluar, secara otomatis data disimpan pada suatu file (sebaiknya nama file disesuaikan dengan jumlah pekerjaan atau mesin).

2. Petunjuk penggunaan perangkat lunak untuk menyelesaikan persamaan relaksasi Lagrangian



- Form yang digunakan adalah frmSchedule (gambar di atas), dengan menekan tombol File | Schedule.
- Pilih file yang akan dijadwalkan.
- Data yang akan dijadwalkan akan muncul. Lalu pilih Schedule | Solving Dual Lagrange yaitu untuk menyelesaikan persamaan – persamaan yang berhubungan dengan relaksasi lagrangian ini.
- Setelah proses di atas selesai, pilih Schedule | Feasible Schedule untuk menampilkan hasil jadwal.
- Kedua langkah di atas dapat dipersingkat dengan menekan tombol Run.
- Hasil dari pengali Lagrange dapat dilihat dengan menekan menu Multiplier atau tekan tombol λ atau π) dan untuk melihat hasil dari penjadwalan pada menu Feasible Schedule atau tombolnya. Hasil penjadwalan yang ditampilkan adalah waktu awal proses tiap operasi berdasarkan mesin yang digunakan (telah dijelaskan pada Bab V).