PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENJADUALAN BANYAK PEKERJAAN DENGAN JARINGAN SARAF TIRUAN

Disusun oleh :

LUKMANUL HAKIM

NRP. 2693100037

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1998

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENJADUALAN BANYAK PEKERJAAN DENGAN JARINGAN SARAF TIRUAN

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Informatika
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Arif Djunaidy	Rully Soelaiman, S.Kom.
NIP. 131 633 403	NIP. 132 086 802

S U R A B A Y A
1998

ABSTRAK

Persoalan penjadualan (*scheduling*) semakin dirasakan dengan meningkatnya persaingan pasar bagi kalangan industri. Penjadualan yang tepat akan memberikan banyak keuntungan, antara lain meningkatkan efisiensi dengan mengurangi penggunaan waktu yang sia-sia, meningkatkan produktivitas, dan dapat memenuhi batas waktu (*dead line*) yang ditentukan oleh pelanggan (*customer*).

Tugas Akhir ini membahas aplikasi jaringan saraf tiruan untuk menyelesaikan persoalan penjadualan banyak pekerjaan (*multiple job scheduling*). Konstruksi jaringan saraf tiruan yang digunakan adalah model yang dirumuskan oleh Zhen-Ping Lo dan Behnam Bavarian. Model yang digunakan adalah jaringan Hopfield yang digunakan untuk menyelesaikan persoalan *traveling salesman* yang dikembangkan menjadi jaringan saraf tiga dimensi, yaitu *Neuro-Box Network* (NBN). Perumusan untuk persoalan *traveling salesman* diperluas untuk persoalan *multiple traveling salesmen* dan *multiple job scheduling* dengan tingkat kesulitan yang semakin bertambah. Model ini kemudian diuji dengan menerapkan beberapa persoalan dengan pembatas (*constraint*) yang bervariasi.

Hasil pengujian menunjukkan bahwa model NBN mengijinkan ekspansi ke ketiga arah dimensi. Sehingga model ini mampu mengakomodasi persoalan dengan jumlah pekerjaan, jumlah mesin, dan jumlah unit alokasi waktu yang berbeda. Konvergensi menuju suatu pemecahan yang *feasible* dapat dicapai tergantung pada persoalan yang ditangani. Hasil uji coba menunjukkan bahwa model ini tidak bisa mendapatkan pemecahan yang *feasible* untuk persoalan dengan pembatas (*constraint*) yang ketat.

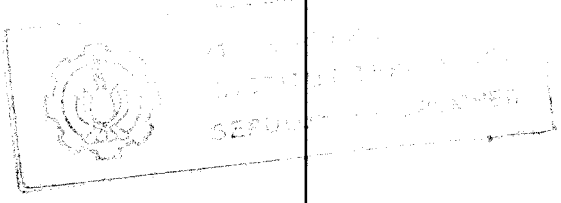
KATA PENGANTAR

Segala puji syukur penulis panjatkan kepada Allah swt., Tuhan semesta alam, karena jika tanpa kehendak-Nya, segala usaha untuk menyelesaikan Tugas Akhir ini tidak akan pernah terwujud. Ucapan terima kasih juga kami sampaikan kepada semua pihak yang telah membantu terselesaikannya Tugas Akhir ini.

Tugas Akhir yang dibuat untuk memenuhi persyaratan mendapatkan gelar Sarjana Teknik Informatika pada jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember ini, berjudul "Perancangan dan Pembuatan Perangkat Lunak Penjadualan Banyak Pekerjaan dengan Jaringan Saraf Tiruan". Tugas Akhir dibuat untuk memberi alternatif pemecahan persoalan penjadualan dengan merumuskan persoalan penjadualan tersebut dalam bentuk jaringan saraf tiruan. Studi literatur dilakukan melalui jurnal-jurnal internasional dan buku-buku referensi yang berkaitan dengan jaringan saraf tiruan dan persoalan optimasi khususnya penjadualan. Semua aktivitas ini memerlukan segala upaya yang tidak mudah. Sehingga dengan terselesaikaanya Tugas Akhir ini, penulis berharap dapat memberikan sumbangan yang berarti bagi semua pihak yang berkepentingan.

Secara khusus kami mengucapkan terima kasih kepada :

1. Dr. Ir. Arif Djunaidy selaku Dosen Pembimbing I, Dosen Wali, dan Ketua Jurusan Teknik Informatika.
2. Rully Soelaiman, S.Kom. selaku Dosen Pembimbing II.



3. Seluruh staf dosen dan karyawan Teknik Informatika atas bimbingan dan bantuannya selama perkuliahan.
4. Rekan-rekan senasib dan sepenanggungan di Lab. KOMISI, "kakak-kakak" dan "adik-adik", terima kasih atas perhatian yang telah kalian berikan.

Penulis menyadari bahwa Tugas Akhir ini bukanlah "karya yang terbaik" tetapi penulis telah mewujudkannya dengan segala "upaya yang terbaik". Segala kritik dan saran yang membangun penulis terima dengan senang hati.

Surabaya, 21 Agustus 1998

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Persoalan penjadualan (*scheduling*) semakin dirasakan dengan meningkatnya persaingan pasar bagi kalangan industri. Penjadualan yang tepat akan memberikan banyak keuntungan, antara lain meningkatkan efisiensi dengan mengurangi penggunaan waktu yang sia-sia, meningkatkan produktivitas, dan dapat memenuhi batas waktu (*dead line*) yang ditentukan oleh pelanggan (*customer*).

Studi mengenai persoalan penjadualan pekerjaan telah dilakukan selama beberapa tahun terakhir ini. Persoalan yang muncul biasanya berkaitan dengan adanya n buah pekerjaan yang akan diproses pada m buah mesin (atau *server*). Permasalahannya adalah bagaimana menemukan urutan pekerjaan dan alokasi waktu yang tepat untuk setiap pekerjaan pada setiap mesin sehingga diperoleh solusi yang optimal, yaitu waktu yang dibutuhkan untuk menyelesaikan semua pekerjaan diupayakan seminimal mungkin.

Tingkat kesulitan persoalan ditentukan oleh parameter yang dimiliki oleh pekerjaan dan mesin. Pendekatan heuristik klasik yang biasa dilakukan adalah menemukan operasi atau pekerjaan berikutnya yang akan diproses berdasarkan beberapa parameter dari pekerjaan atau mesin. Contoh dari parameter yang

digunakan adalah waktu pemrosesan (*processing time*), jatuh tempo (*due date*), biaya (*cost*), waktu penyiapan (*set-up time*), waktu kedatangan (*arrival time*), dan pemuatan mesin (*machine loading*). Beberapa aturan yang digunakan untuk menentukan operasi berikutnya adalah FIFO (*first in first out*), SPT (*shorthest processing time*), EDD (*earliest due date*).

Studi terbaru dalam jaringan saraf tiruan memberikan dasar dalam menyelesaikan persoalan optimasi *NP-complete*, seperti TSP (*traveling salesman problem*), MTSP (*multiple traveling salesman problem*), dan MJSP (*multiple job scheduling problem*). Penyelesaian persoalan *NP-complete* bersifat *nondeterministic polynomial*. Suatu persoalan *NP-complete* tidak mempunyai penyelesaian yang lebih baik selain mencoba semua kemungkinan dari penyelesaian yang ada. Kalau cara ini dilakukan, akan membutuhkan waktu komputasi yang tidak sedikit untuk mendapatkan penyelesaian masalah. Waktu komputasi akan bertambah seiring dengan bertambahnya satu faktor pada persoalan *NP-complete*.

Hopfield telah mendefinisikan arsitektur umum dari jaringan saraf tiruan untuk digunakan dalam menyelesaikan persoalan-persoalan optimasi. Hopfield dan Tank[1] telah merumuskan dan memecahkan *traveling salesman problem* dengan jaringan saraf dua dimensi (*2-D neural network*). Pendekatan ini kemudian dikembangkan lebih jauh untuk menyelesaikan *multiple traveling salesmen problem*. Kemudian, pendekatan di atas diperluas untuk menyelesaikan

[1] J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernet*, 52, 141-152, 1985.

persoalan penjadualan banyak pekerjaan dengan membangun jaringan saraf tiga dimensi (*3-D Neuro Box Network, NBN*).

1.2 Tujuan

Tugas Akhir ini bertujuan merancang dan membuat perangkat lunak penjadualan banyak pekerjaan dengan konstruksi jaringan saraf tiruan yang dirumuskan oleh Zhen-Ping Lo dan Behnam Bavarian[2] (*3-D Neuro Box Network, NBN*). Berdasarkan hasil pengujian terhadap perangkat lunak diharapkan dapat memberikan sebuah evaluasi terhadap konstruksi jaringan saraf tiruan tersebut.

1.3 Permasalahan

Permasalahan dalam Tugas Akhir ini adalah bagaimana konstruksi jaringan saraf tiruan yang digunakan bisa memberikan penyelesaian yang *feasibel*. Penyelesaian tersebut berupa jadual yang diperoleh, yaitu hasil penugasan n buah pekerjaan pada m buah mesin. Pengertian *feasibel* disini adalah penyelesaian tersebut bisa diterima atau valid dan optimal, yaitu waktu yang dibutuhkan untuk menyelesaikan semua pekerjaan diusahakan seminimal mungkin.

---

[2] Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling with Artificial Neural Networks", University of California, Irvine, 1991.

1.4 Pembatasan Masalah

Pada dasarnya, Tugas Akhir ini menitikberatkan pada persoalan penjadualan klasik dengan memperhatikan jatuh tempo (*deadline requirement*). Namun, persoalan dirumuskan dalam bentuk jaringan saraf tiruan dengan menggunakan model NBN.

Pembatasan masalah diuraikan dalam bentuk asumsi-asumsi terhadap pekerjaan beserta panjang proses dan jatuh temponya, mesin beserta kapasitasnya, dan alokasi waktu. Setiap pekerjaan yang ditangani diasumsikan memiliki tipe yang sama, tetapi memiliki panjang proses (*length*) dan jatuh tempo (*due date*) yang berbeda. Panjang proses dan jatuh tempo berupa nilai konstan yang dapat ditentukan. Setiap mesin diasumsikan memiliki tipe yang sama, tetapi memiliki kapasitas yang tidak sama yang nilainya juga dapat ditentukan. Setiap mesin juga diasumsikan hanya dapat memroses satu pekerjaan pada satu alokasi waktu dan tidak terjadi kerusakan selama perencanaan. Pada jadual akhir, setiap pekerjaan hanya ditangani oleh satu mesin.

1.5 Metodologi Penelitian

Metodologi yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

♦Studi literatur dari berbagai buku dan jurnal

Studi dilakukan dengan mempelajari literatur berkaitan dengan persoalan optimasi khususnya *NP-complete*, jaringan saraf tiruan khususnya jaringan Hopfield yang digunakan untuk menyelesaikan persoalan *traveling salesman*, dan pemrograman berorientasi obyek.

♦Perancangan perangkat lunak dan implementasi struktur data

Perancangan perangkat lunak dilakukan dengan pendekatan pemrograman berorientasi obyek. Berdasarkan hasil rancangan tersebut, dibuat implementasi struktur data yang digunakan dalam perangkat lunak.

♦Pembuatan perangkat lunak

Perangkat lunak dibuat berdasarkan hasil perancangan dengan menggunakan bahasa pemrograman *C++*.

♦Pengujian perangkat lunak

Pengujian dilakukan dengan menerapkan persoalan dengan pembatas (*constraint*) yang bervariasi.

♦Perbaikan untuk meningkatkan kinerja

Perbaikan yang dilakukan meliputi pengecekan masukan, penyesuaian bentuk tampilan, dan mengurangi *bug* perangkat lunak.

♦Pembuatan laporan

Laporan ini berupa dokumentasi terhadap metodologi yang dilakukan yang terangkum dalam naskah Tugas Akhir.

1.6 Sistematika Penulisan

Sistematika yang digunakan dalam Tugas Akhir dijelaskan berikut ini.

- Bab I Pendahuluan, menjelaskan mengenai latar belakang, permasalahan dan batasannya, tujuan, metodologi penelitian, dan sistematika penulisan.

- Bab II Jaringan Saraf Tiruan, membahas pengertian jaringan saraf tiruan, elemen pengolah, penetapan nilai bobot, dan jaringan Hopfield.
- Bab III Penjadualan Banyak Pekerjaan, membahas persoalan penjadualan banyak pekerjaan dengan didahului penjelasan mengenai traveling salesman problem dan multiple traveling salesmen problem yang mendasari perumusan persoalan penjadualan banyak pekerjaan.
- Bab IV Perancangan dan Pembuatan Perangkat Lunak, membahas sistem perangkat lunak penjadualan banyak pekerjaan, perancangan dan implementasi struktur data yang digunakan oleh perangkat lunak, serta hirarki proses dan implementasinya.
- Bab V Uji Coba dan Evaluasi Perangkat Lunak, membahas hasil uji coba perangkat lunak yang sudah jadi dan menguji kemampuan perangkat lunak.
- Bab VI Kesimpulan dan Saran, menguraikan kesimpulan dari bab-bab sebelumnya serta saran berkaitan dengan perancangan dan pembuatan perangkat lunak.

BAB II

JARINGAN SARAF TIRUAN

Bab ini menjelaskan tentang jaringan saraf tiruan meliputi pengertian jaringan saraf tiruan, elemen pengolah, nilai bobot, serta fungsi keaktifan sel saraf. Pada akhir bab dijelaskan mengenai jaringan Hopfield sebagai salah satu topologi jaringan saraf yang banyak digunakan dalam menyelesaikan persoalan optimasi.

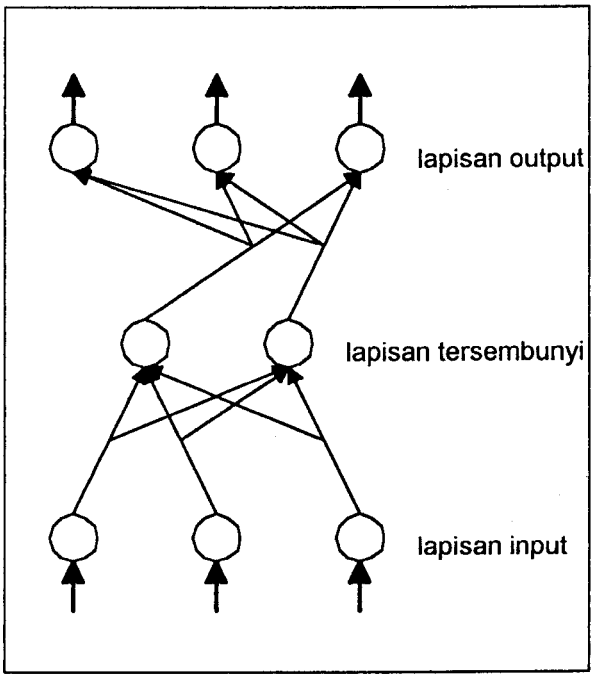
2.1 Pengertian Jaringan Saraf Tiruan[3]

Sebuah jaringan saraf tiruan (JST) adalah sebuah grup yang terdiri atas elemen pengolah (*processing element*). JST memiliki ciri khas, yaitu sebuah subgrup akan melakukan perhitungan secara terpisah (*independent*) dan memberikan hasil dari perhitungan tersebut ke sebuah subgrup yang lain. Pada akhirnya, sebuah subgrup yang terdiri atas satu atau lebih elemen pengolah akan menentukan nilai output dari JST. Setiap elemen pengolah melakukan perhitungan berdasarkan penjumlahan dari masing-masing nilai inputnya dikalikan dengan sebuah nilai bobot (*weight*). Sebuah subgrup dari elemen pengolah dikenal dengan lapisan (*layer*). Lapisan pertama merupakan lapisan input dan lapisan yang terakhir adalah lapisan output. Lapisan yang berada di antara lapisan input dan

[3] Valluru B. Rao and Hayagriva V. Rao, "C++ Neural Networks and Fuzzy Logic", 68, 1993.

lapisan output dikenal dengan lapisan tersembunyi (*hidden*). Elemen pengolah adalah sebuah unit yang memiliki kemiripan fungsi dengan sel saraf pada otak manusia. Oleh sebab itu elemen pengolah diibaratkan sebagai sebuah *cell*, *neuromimes*, atau sel saraf tiruan (*artificial neuron*). Sebuah fungsi *threshold* sering digunakan untuk menentukan nilai output dari sebuah sel saraf pada lapisan output. Nilai yang dikeluarkan dari fungsi *threshold* menentukan karakteristik dari sel saraf. Karakteristik tersebut adalah sel saraf aktif (*fired*) atau sel saraf tidak aktif. Dalam Tugas Akhir ini digunakan istilah sel saraf pada elemen pengolah meskipun elemen pengolah merupakan sel saraf tiruan. Istilah koneksi (*connection*) juga digunakan untuk *synapses*, yaitu hubungan di antara sel saraf. Dalam bentuk *graph* sebuah koneksi disimbolkan dengan *edge* dan sebuah sel saraf dengan *node*. Gambar 2.1 adalah sebuah model jaringan saraf tiruan. Dalam gambar ini, setiap sel saraf digambarkan dengan sebuah lingkaran (*circular node*). Model tersebut terdiri atas tiga lapisan, yaitu sebuah lapisan input, sebuah lapisan tersembunyi, dan sebuah lapisan output. Koneksi yang ada menunjukkan hubungan antar lapisan.

Berdasarkan jumlah lapisan yang dapat dibentuk, JST dapat dibedakan atas lapisan tunggal (*single layer*), yaitu jaringan yang hanya terdiri atas satu lapisan, dan lapisan ganda (*multilayer*), yaitu jaringan yang memiliki lebih dari satu lapisan. Berdasarkan arah aliran sinyal, JST dapat dibedakan atas *feedforward* dan *recurrent*. Pada *feedforward* arah aliran sinyal berasal dari lapisan input ke lapisan output. Sedangkan pada *recurrent* terdapat arah aliran sinyal yang berasal dan kembali ke unit yang sama.



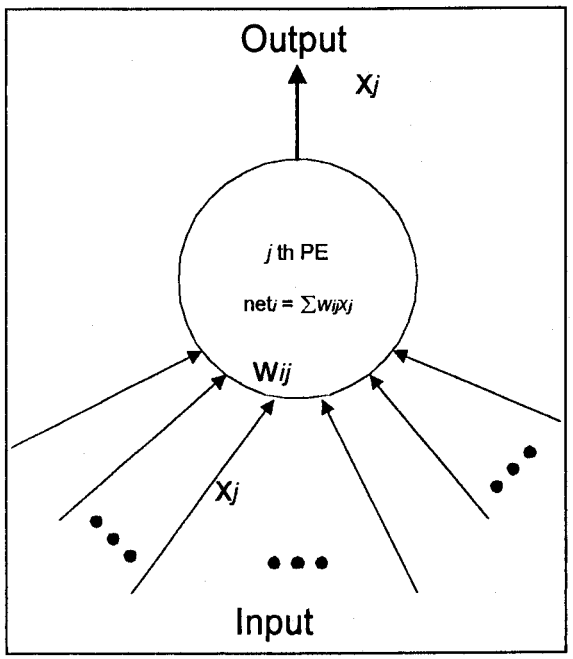
Gambar 2.1 Sebuah Model Jaringan Saraf Tiruan

2.1.1 Elemen Pengolah

Seperti dijelaskan sebelumnya elemen pengolah (*processing element*, disingkat PE) merupakan unit terkecil dari JST. Gambar 2.2 menunjukkan model PE secara garis besar. Setiap PE mempunyai nomor tersendiri. PE mempunyai banyak input, tetapi hanya memiliki satu output. Setiap koneksi ke PE mempunyai satu nilai bobot (*connection strength*, atau *weight*). Nilai bobot dari node ke-*j* ke node ke-*i* dinotasikan dengan w_{ij}. Suatu input dapat berfungsi sebagai *excitatory* atau *inhibitory*. Input yang *excitatory* mempunyai nilai bobot positif sehingga menguatkan energi sel saraf, sedangkan input yang *inhibitory* mempunyai nilai bobot negatif sehingga melemahkan energi sel saraf. Nilai input dari setiap PE atau unit ke-*i* ditetapkan sebagai berikut:

$$net_i = \sum x_j w_{ij}$$

dimana x_j menyatakan nilai output sel saraf ke-j dan w_{ij} menyatakan nilai bobot koneksi antara sel saraf ke-i dan sel saraf ke-j.



Gambar 2.2 Model Elemen Pengolah

2.1.2 Penetapan nilai bobot

Penetapan nilai bobot dilakukan dengan pelatihan (*training*). Proses pelatihan inilah yang membedakan fungsi dari berbagai jenis JST. Nilai bobot yang tepat akan menghasilkan nilai output yang diharapkan. Beberapa aturan perlu ditetapkan untuk memperbarui nilai bobot dan untuk menentukan kapan proses memperbarui nilai bobot bisa dihentikan. Proses pembaruan nilai bobot dikenal dengan proses belajar (*learning*). Jika proses belajar sedang terjadi pada

suatu jaringan maka, jaringan tersebut dikatakan sedang melakukan proses pelatihan (*training*).

Secara umum proses belajar dapat dibedakan menjadi dua jenis, yaitu dengan pembimbing (*supervised*) dan tanpa pembimbing (*unsupervised*). Proses belajar dengan pembimbing dilakukan dengan memberikan berbagai macam pola pelatihan (*traing pattern* atau *training vector*). Masing-masing beserta nilai output yang diinginkan (*training set*). Selanjutnya nilai bobot diatur berdasarkan algoritma yang ada. Proses belajar dihentikan bilamana jaringan telah memberikan hasil yang diinginkan untuk sejumlah nilai output yang diberikan. Pada saat itu, dapat dikatakan nilai bobot yang diberikan merupakan nilai yang paling sesuai untuk JST yang digunakan. JST yang dilatih untuk menghasilkan nilai output tertentu berdasarkan nilai input tertentu dinamakan *associative memory*. Untuk model ini, jika nilai output yang diinginkan sama dengan nilai input yang diberikan, maka *associative memory* disebut sebagai *autoassociative memory*, sedangkan bila tidak disebut sebagai *heteroassociative memory*.

Proses belajar tanpa pembimbing tidak memerlukan sejumlah *training set* untuk mengatur besarnya nilai bobot. Hal ini disebabkan adanya pemantauan internal yang berfungsi mengawasi tingkah laku jaringan. Jaringan akan mencari kecenderungan dari sinyal input dan akan menyesuaikan dirinya berdasarkan suatu fungsi tertentu. Jaringan harus mempunyai informasi tentang cara untuk mengatur dirinya. Proses belajar tanpa pembimbing ada yang menekankan pada adanya suatu keterkaitan atau kerja sama antar elemen pengolah. Pada beberapa susunan jaringan, bila ada suatu input dari luar yang mengaktifkan sebuah elemen

pengolah pada suatu lapisan, maka elemen pengolah-elemen pengolah yang lain pada lapisan tersebut juga akan aktif dan mengakibatkan nilai keaktifan pada lapisan tersebut. Demikian juga sebaliknya, bila ada suatu input dari luar yang membuat sebuah elemen pengolah pada suatu lapisan tidak aktif, maka elemen pengolah-elemen pengolah yang lain pada lapisan tersebut juga tidak akan aktif.

2.1.3 Fungsi keaktifan

Operasi dasar yang dilakukan oleh sel saraf adalah menghitung nilai output yang merupakan hasil dari fungsi keaktifan terhadap jumlah input yang masuk pada sebuah sel saraf. Semua sel saraf pada satu lapisan tertentu mempunyai fungsi keaktifan yang sama. Fungsi keaktifan tersebut, antara lain:

(i) Fungsi Identitas:

$f(x) = x$, dimana x menyatakan nilai input sel saraf.

(ii) Fungsi Binary Step (*binary step function*) dengan nilai ambang batas θ.

$$f(x) = \begin{cases} 1, x \geq \theta \\ 0, x < \theta \end{cases}$$

Dalam fungsi ini, x menyatakan nilai input sel saraf dan θ menyatakan nilai ambang batas (*threshold*). Jaringan dengan lapisan tunggal menggunakan fungsi ini untuk mengubah input menjadi output yang merupakan bilangan biner (1 atau 0) atau bipolar (1 atau -1). Nama lain dari fungsi ini adalah *threshold function* atau *heaviside function*.

(iii) Fungsi Binary Sigmoid:

$$f(x) = \frac{1}{1+\exp(-\sigma x)}$$

Dalam fungsi ini x menyatakan nilai input sel saraf dan σ merupakan konstanta yang mempengaruhi lebar kurva (*width curve*). Fungsi logistik dan fungsi tangen hiperbolik merupakan fungsi *binary sigmoid* yang sering digunakan.

(iv) Fungsi Bipolar Sigmoid:

$$g(x) = 2f(x) - 1 = \frac{2}{1+\exp(-\sigma x)} - 1 = \frac{1-\exp(-\sigma x)}{1+\exp(-\sigma x)}$$

Dalam fungsi ini x juga menyatakan nilai input sel saraf dan σ juga merupakan konstanta yang mempengaruhi lebar kurva (*width curve*). Fungsi logistik sigmoid dapat diubah jangkauannya sesuai dengan persoalan yang dihadapi. Jangkauan yang sering digunakan adalah antara –1 sampai 1. Sigmoid dengan jangkauan antara –1 dan 1 dinamakan *bipolar sigmoid*. Selain *bipolar sigmoid*, fungsi tangen hiperbolik juga sering digunakan untuk menghasilkan nilai output antara –1 dan 1. Untuk menghasilkan nilai output antara 0 dan 1, kedua fungsi tersebut juga sering digunakan.

2.2 Jaringan Hopfield[4]

Model sel saraf yang digunakan dalam jaringan hopfield adalah elemen pengolah yang menggunakan variabel biner (*two-state*)[5]. Perhitungan yang dilakukan pada sel saraf ini adalah penjumlahan dari setiap nilai input yang berasal dari sel saraf yang lain dikalikan dengan masing-masing nilai bobot pada setiap koneksi. Sebuah fungsi *threshold* menentukan keaktifan dari sel saraf. Jika hasil dari perhitungan lebih besar atau sama dengan nilai *threshold* maka sel saraf menjadi aktif, tetapi jika hasil perhitungan kurang dari *threshold* maka sel saraf menjadi tidak aktif. Jaringan saraf yang diusulkan oleh Hopfield mempunyai hubungan atau koneksi yang bersifat *feedback*. Jadi nilai output dari sel saraf bisa menjadi nilai input pada sel saraf itu sendiri pada iterasi berikutnya. Hal inilah yang membedakan jaringan Hopfield dengan model *perceptron* pada jaringan *feedforward*[6]. Gambar 2.3 menampilkan arsitektur jaringan Hopfield secara garis besar.

Lingkaran pada setiap pertemuan koneksi merupakan bobot dari koneksi tersebut. Perilaku dari neuron ke-*i* diberikan oleh persamaan beda orde pertama sebagai berikut:

$$v_i(k+1) = f\left[\sum_{j=1}^{n} w_{ij} v_j(k) + I_i(k)\right], \qquad (2.1)$$

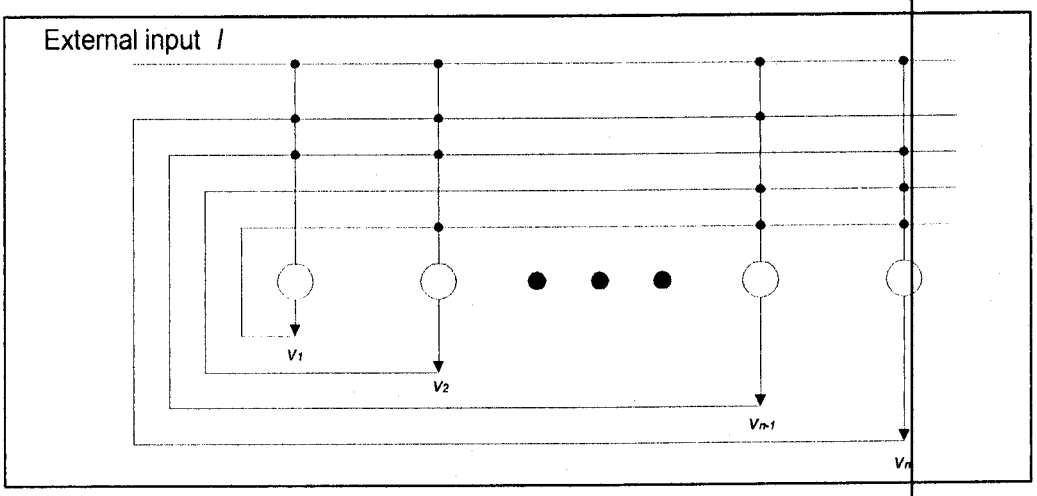
---

[4] Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling with Artificial Neural Networks", University of California, Irvine,1991.

[5] J.J. Hopfield, "Neural Network and Physical systems with Emergent Collective Computational Abilities", Proc. Natl. Acad. Sci, U.S.A. 79, 2554-2558,1982.

[6] R. Rosenblatt, "Principles of Neurodynamics", New York, Spartan Books ,1959.

dimana k adalah indek dari waktu, v_i adalah nilai aktivasi dari sel saraf ke-i, w_{ij} adalah nilai bobot dari koneksi antara sel saraf ke-j ke sel saraf ke-i, I_i adalah input eksternal dari sel saraf ke-i, dan $f(.)$ adalah *binary threshold function.*



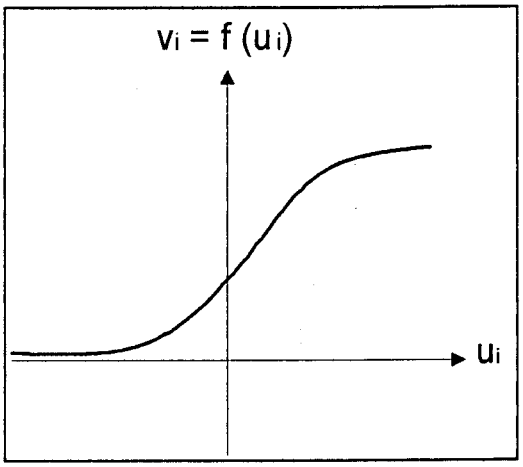
Gambar 2.3 Arsitektur Jaringan Hopfield

Pada umumnya, pembaruan/peremajaan status dari sel saraf dilakukan secara *asynchronous* dan acak pada sebuah nilai rata-rata (*mean rate*). Hopfield sudah membuktikan global konvergensi dari jaringan untuk matriks bobot dengan koneksi yang simetris dengan nilai nol pada elemen diagonal (dalam hal ini tidak ada *self activation*). Nilai bobot dari koneksi menggabungkan semua properti informasi pemrosesan dari jaringan. Kemudian nilai bobot ini diatur untuk menyelesaikan persoalan yang spesifik dengan suatu pertimbangan tertentu. Secara praktis, jaringan Hopfield sudah pernah digunakan untuk memodelkan *associatif memory*. Dalam hal ini informasi yang diingat adalah data yang tidak lengkap (*noisy data*). Matriks nilai bobot dihitung berdasarkan matriks korelasi dari suatu pola (*pattern*) yang menjadi target. Pada umumnya, untuk sebuah persoalan yang spesifik, sebuah *computational energy*, E, didefinisikan, untuk

mengatur pola dari matriks nilai bobot. Angka perubahan (*rate of change*) tingkat aktivasi dari sel saraf diatur sama dengan minus dari *partial derivatif* energi berkenaan dengan variabel sel saraf (metode optimasi gradien diferensial). Kemudian dengan menyamakan hasil persamaan ke sisi kanan dari persamaan (2.1), didapatkan suatu nilai yang spesifik dari nilai bobot koneksi. Respon yang bertingkat dari sel saraf memotivasi Hopfield mengembangkan arsitektur jaringannya untuk mempunyai sel saraf dengan nilai output yang kontinyu. Karakteristik sel saraf didefinisikan dengan persamaan diferensial orde pertama seperti berikut.

$$C_i = \frac{du_i}{dt} = -\frac{u_i}{R_i} + \sum_{j=1}^{n} w_{ij} v_j + I_i, \qquad (2.2)$$

Dalam persamaan di atas, u_i merupakan potensial aktivasi internal sel saraf, R_iC_i adalah nilai resistansi/kapasitansi yang mendefinisikan waktu konstan dari rangkaian RC, w_{ij} adalah nilai bobot koneksi (konduktansi) dari nilai output sel saraf ke-j sebagai nilai input dari sel saraf ke-i, I_i adalah nilai input eksternal, dan v_j mendefinisikan nilai output dari sel saraf yang berkaitan dengan potensial aktivasi internal melalui sebuah fungsi *threshold* $v_j = f(u_j)$. Pada umumnya, fungsi ini akan meningkat secara monoton seperti fungsi sigmoid dalam Gambar 2.4.

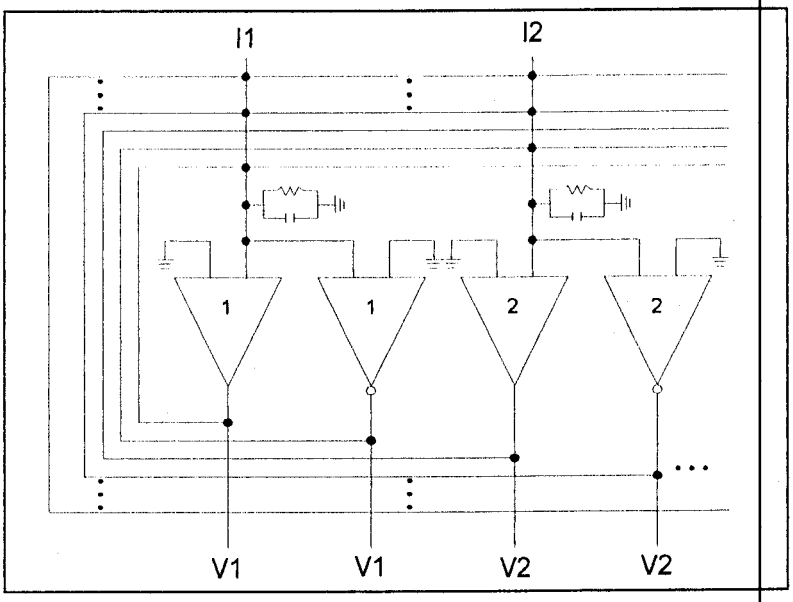


Gambar 2.4 Fungsi Sigmoid *Nonlinear*, Menggambarkan Kondisi Internal Sel Saraf dan Potensial Nilai Outputnya

Arsitektur jaringan Hopfield yang kontinyu dapat direalisasikan dengan rangkaian aktif orde pertama, seperti diilustrasikan dalam Gambar 2.5. Nilai output dari operasional amplifier dapat berupa nilai positif atau negatif. Fungsi energi kuadratik yang secara umum digunakan untuk membuktikan konvergensi global dari jaringan ditunjukkan dalam persamaan berikut:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} v_i v_j - \sum_{i=1}^{n} I_i v_i + \sum_{i=1}^{n}\left(\frac{1}{R_i}\right)\int_0^{v_i} f^{-1}(v)dv, \quad \text{.....(2.3)}$$

dimana $f^{1}(v)$ adalah invers dari fungsi *sigmoid nonlinier*. Kondisi konvergensi tergantung pada kesimetrisan matriks nilai bobot dari koneksi yang meningkat secara monoton (*bounded sigmoid nonlinearity*). Fungsi energi atau fungsi biaya di atas hanya sesuai untuk persoalan yang spesifik, artinya untuk setiap persoalan perlu dirumuskan fungsi energi tersendiri sesuai dengan pola matriks nilai bobot yang diinginkan. Persamaan differensial dari sel saraf atau matriks nilai bobot dari koneksi dihitung dengan menurunkan persamaan (2.3).



Gambar 2.5 Arsitektur Rangkaian Jaringan Hopfield

Kondisi sel saraf berevolusi menurut waktu, bergerak ke arah penurunan permukaan energi (*energy surface*) ke arah konvergensi menuju ke sebuah minimum lokal. Jadi, untuk persoalan yang diberikan, fungsi energi harus dikonstruksi sedemikian rupa sehingga konfigurasi dari kondisi jaringan mewakili pemecahan-pemecahan yang mungkin pada minimum lokal.

BAB III

PENJADUALAN BANYAK PEKERJAAN

Bab ini menjelaskan persoalan penjadualan banyak pekerjaan (*multiple job scheduling*) beserta perumusannya dalam bentuk jaringan saraf tiruan. Bab ini didahului dengan penjelasan mengenai persoalan *traveling salesman problem* dan *multiple traveling salesmen problem* yang mendasari perumusan persoalan penjadualan banyak pekerjaan. Kemudian, dilanjutkan dengan penjelasan mengenai persoalan penjadualan banyak pekerjaan berikut perumusannya dalam bentuk jaringan saraf tiruan, yaitu *Neuro-Box Network* (NBN).

3.1 Traveling Salesman Problem[7]

Proses optimasi pada *traveling salesman problem* (TSP) merupakan topik klasik dari riset operasional. Persoalan tersebut dapat didefinisikan seperti berikut. Seorang salesman akan mengunjungi beberapa kota. Data yang diketahui adalah jumlah kota yang harus dikunjungi beserta jarak antar kotanya. Permasalahannya adalah bagaimana menemukan rute terpendek yang mengunjungi setiap kota dengan syarat, setiap kota hanya dikunjungi satu kali dan kembali ke kota dimana rute tersebut dimulai.

[7] Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling with Artificial Neural Network", University of California, Irvine, 1991.

Penyelesaian persoalan ini dengan pendekatan langsung adalah dengan menghitung semua kemungkinan rute yang ada, kemudian dipilih satu rute yang terpendek. Jika ada n kota yang harus dikunjungi maka ada $n!/(2n)$ rute yang harus diselidiki. Dengan cara ini, jumlah waktu komputasi yang diperlukan meningkat seiring dengan bertambahnya ukuran dari persoalan, yaitu jumlah kota. Sebagai contoh, untuk 15 kota akan didapat kurang lebih $4{,}4 \times 10^{10}$ rute, yang mencerminkan banyaknya rute yang harus diselidiki untuk mendapatkan rute yang optimal sehingga sulit diselesaikan dengan bantuan komputer secara efisien.

Hopfield dan Tank[8] telah memetakan TSP ke dalam arsitektur jaringan saraf tiruan. Mereka telah mendefinisikan konfigurasi jaringan dan fungsi energi atau fungsi biaya yang berkorespondensi dengan persoalan tersebut. Dalam hal ini, untuk n kota dipilih $N = n^2$ sel saraf dalam bentuk matriks 2-D $n \times n$. Setiap baris dari matriks $n \times n$ ini berkorespondensi dengan satu kota tertentu dan setiap kolom berkorespondensi dengan satu posisi tertentu. Keadaan akhir dipaksakan sedemikian rupa sehingga hanya ada satu output sel saraf yang bernilai satu dan yang lain bernilai nol untuk setiap baris dan kolom. Karena setiap kota hanya dikunjungi satu kali, setiap kolom hanya mempunyai satu masukan, demikian juga untuk setiap baris. Jumlah seluruh masukan sama dengan n. Sebagai contoh, untuk konfigurasi jaringan 4 x 4 kondisi akhir dari matriks ditunjukkan dalam Tabel 3.1.

---

[8] J.J. Hopfield and D.W. Tank, "Neural Computation of Decision in Optimization Problem", Biol. Cybernet. 52, 141-152, 1985.

Tabel 3.1
Kondisi Akhir untuk Konfigurasi Jaringan 4 x 4

	1	2	3	4
C_1	0	1	0	0
C_2	0	0	1	0
C_3	1	0	0	0
C_4	0	0	0	1

Dari Tabel 3.1, terlihat secara jelas hasil rute yang diperoleh, yaitu $C_3 \rightarrow C_1 \rightarrow C_2 \rightarrow C_4 \rightarrow C_3$ dengan panjang rute $l_{dist} = d_{C3C1} + d_{C1C2} + d_{C2C4} + d_{C4C3}$ dimana d_{CiCj} adalah jarak antara kota C_i dan C_j. Sedang fungsi energi yang memaksa semua pembatas ke dalam kondisi ini adalah sebagai berikut:

$$E = \frac{1}{2}A\sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} v_{xi}v_{xj} + \frac{1}{2}B\sum_{i=1}^{n}\sum_{x=1}^{n}\sum_{\substack{y=1 \\ y\neq x}}^{n} v_{xi}v_{yi} + \frac{1}{2}C\left(\sum_{x=1}^{n}\sum_{i=1}^{n} v_{xi} - n\right)^2, \quad \text{.....(3.1)}$$

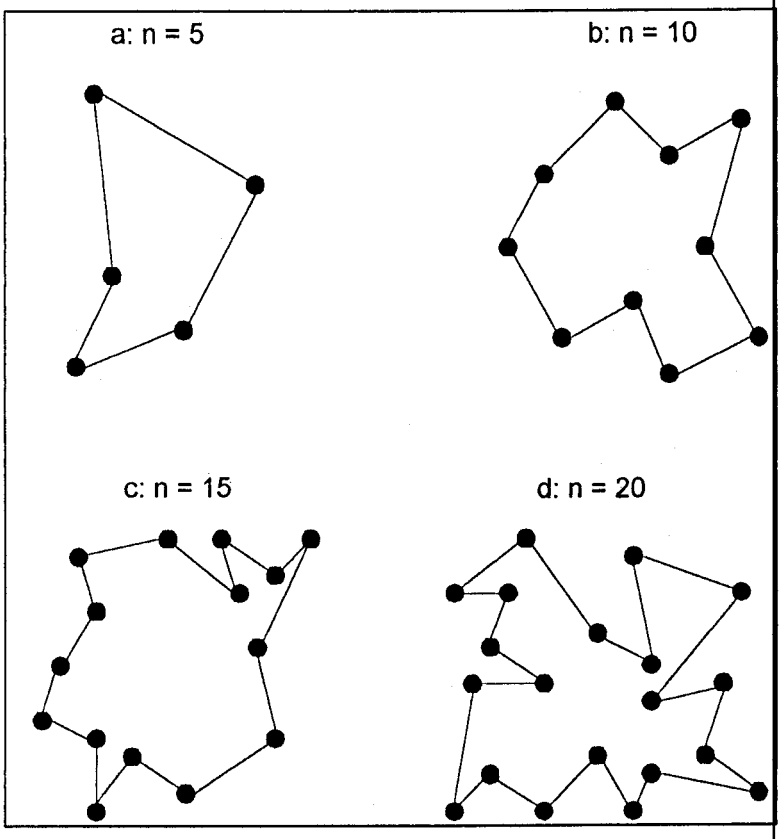
dimana v_{xi} adalah output sel saraf, indek x dan y mengacu pada kota atau baris pada matriks serta indek i dan j mengacu pada posisi rute atau kolom pada matriks. Tahap yang pertama memaksa setiap baris mempunyai satu nilai bukan nol. Tahap kedua berperilaku sama, tetapi diperlakukan untuk setiap kolom. Tahap ketiga merupakan faktor skala umum. Tahap inilah yang memaksa hanya ada n nilai bukan nol pada matriks, karena pemecahan nol memenuhi kebutuhan dari dua tahap sebelumnya. Kemudian, jarak rute ditambahkan pada fungsi energi, sehingga selama proses minimisasi sebuah rute dengan jarak minimal ditemukan. Sehingga total fungsi biaya dapat dirumuskan menjadi:

$$E_t = E + \frac{1}{2}D\sum_{x=1}^{n}\sum_{\substack{y=1 \\ y\neq x}}^{n}\sum_{i=1}^{n} d_{xy}v_{xi}(v_{y,i+1} + v_{y,i-1}), \quad \text{.....(3.2)}$$

Dalam persamaan (3.1) dan (3.2) di atas, koefisien A, B, C, dan D merupakan parameter *gain* yang memberi bobot pada setiap tahap pada fungsi energi. Seperti dijelaskan oleh Hopfield dan Tank persamaan diferensial yang mendefinisikan karakteristik dari sel saraf dapat diturunkan dari E_t sebagai berikut:

$$\frac{du}{dt} = -\frac{u_{xi}}{\tau} - A\sum_{\substack{j=1 \\ j\neq i}}^{n} v_{xj} - B\sum_{\substack{y=1 \\ y\neq x}}^{n} v_{yi} - C\left(\sum_{x=1}^{n}\sum_{j=1}^{n} v_{xj} - n\right) - D\sum_{y=1}^{n} d_{xy}(v_{y,i+1} + v_{y,i-1}),\ldots\ldots(3.3)$$

dimana u_{xi} adalah kondisi internal atau nilai aktivasi dari sel saraf, dan v_{xi} adalah output dari sel saraf seperti dijelaskan sebelumnya. Model ini kemudian dicoba dengan simulasi komputer, yang hasilnya ditunjukkan dalam Gambar 3.1.



Gambar 3.1
Beberapa Contoh Hasil Simulasi TSP

Konvergensi simulasi menuju sebuah rute dengan panjang rute kecil yang valid sangatlah sensitif pada *gain* A, B, C, dan D untuk semua tahap dalam fungsi energi. Parameter A, B, C, D yang diusulkan oleh Hopfield dan Tank dalam model TSP tidak selalu memberikan hasil yang optimal, dimana hasil eksperimen hanya 10% dari keseluruhan solusi yang memberikan rute valid. Dengan menyesuaikan parameter A, B, C, D kinerja konvergensi dapat dikembangkan. Konfigurasi akhir yang tidak valid dari solusi dapat dikelompokkan sebagai berikut:

1. Sebuah baris atau beberapa baris tidak memiliki masukan atau memiliki masukan lebih dari satu (berkorespondensi dengan sebuah rute yang mengunjungi kota tertentu beberapa kali).
2. Sebuah kolom atau beberapa kolom tidak memiliki masukan atau memiliki masukan lebih dari satu (berkorespondensi dengan sebuah rute yang mengunjungi beberapa kota secara bersama-sama).

Fungsi energi yang sudah dirumuskan di atas dapat dikelompokkan menjadi dua bagian. Bagian pertama, yaitu tahap-tahap dengan parameter *gain* A, B, dan C yang memaksa matriks permutasi untuk hanya mempunyai sebuah masukan pada setiap baris dan kolom dengan total n buah masukan. Hal ini akan menghasilkan rute yang valid. Bagian kedua, yaitu tahap dengan parameter *gain* D yang berfungsi untuk meminimalkan fungsi biaya. Kedua bagian ini bersaing satu sama lain. Nilai yang besar untuk D, *gain* untuk fungsi biaya akan memaksa mendapatkan jarak minimal, tetapi sebagian besar rute yang diperoleh tidak valid. Dengan mengurangi nilai D struktur valid dipaksakan. Tetapi, rute yang diperoleh

memiliki kualitas yang jelek untuk total jarak. Dengan demikian terdapat *trade-off* untuk menemukan nilai optimal dari parameter-parameter ini. Beberapa pendekatan sudah diusulkan untuk menyelesaikan persoalan ini[9].

Ide dari pemecahan TSP dapat diperluas untuk persoalan pengurutan atau penjadualan dimana *n* buah kota identik dengan *n* buah pekerjaan dan seorang salesman identik dengan sebuah mesin. Permasalahannya adalah bagaimana menemukan urutan dari pekerjaan ini yang memiliki waktu minimal atau membutuhkan usaha yang lebih sedikit.

Pada kenyataanya proses manufaktur melibatkan banyak mesin industri, sehingga pada pembahasan berikut dijelaskan mengenai *multiple travelling salesmen problem* (MTSP) yang secara analog mirip persoalan tersebut. Sehingga ide dari pemecahan MTSP dapat diperluas untuk persoalan penjadualan dimana *n* buah kota identik dengan *n* buah pekerjaan dan *m* salesman identik dengan *m* buah mesin.

---

[9] R. D. Brandt, Y. Wang, A. J. Laub, dan S. K. Mitra, "Alternative Networks for Solving the Traveling Salesman Problem and the List-matching Problem", Proc. Int. Conf. On Neural Network, Vol. 3, pp. 333-340, 1988.

3.2 Multiple Traveling Salesmen Problem[10]

Multiple traveling salesmen problem (MTSP) merupakan perluasan dari TSP dengan tingkat kesulitan yang lebih tinggi. Persoalan tersebut dapat didefinisikan seperti berikut. Sejumlah salesman akan mengunjungi beberapa kota. Data yang diketahui adalah jumlah kota beserta jarak antar kotanya. Permasalahannya adalah bagaimana menemukan sejumlah rute untuk m salesman yang dimulai dan diakhiri dari satu kota tertentu sehingga total jarak untuk semua rute adalah minimal dengan syarat setiap salesman mengunjungi paling sedikit r kota.

Alasan utama kesulitan persoalan optimasi ini berkaitan dengan adanya banyak kemungkinan pemecahannya. Sebagai contoh, untuk kasus sederhana dengan jumlah salesman, $m = 2$, batas jumlah kota yang harus dikunjungi pada satu rute, $r = 1$ dan jumlah kota n, kemungkinan pemecahannya adalah sebagai berikut[11]:

$$\binom{n}{1}\frac{(n-1)!}{2(n-1)}+\binom{n}{2}\frac{(n-2)!}{2(n-2)}+\ldots+\binom{n}{\aleph}\frac{(n-\aleph)!}{2(n-\aleph)}$$

dimana $\aleph$ adalah nilai pembulatan dari $n/2$. Dengan rumusan ini, jika $n = 10$ (persoalan untuk 10 kota), maka kemungkinan pemecahannya kurang lebih mencapai 400.000, sebagai perbandingan, untuk persoalan yang sama, TSP biasa akan memberikan kemungkinan pemecahan kurang lebih sebesar 200.000.

---

[10] Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling with Artificial Neural Network", University of California, Irvine, 1991.

[11] E. Wacholder, J. Han and R.C. Momn, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem", Biol. Cybernet. 61, 11-19, 1989.

Akibat adanyan ledakan kemungkinan pemecahan yang dapat timbul, maka Wacholder[12] mengembangkan *neuromorphic* kota-posisi TSP dari Hopfield dan Tank dengan merumuskan persoalan menjadi (n+m-1) buah kota dengan seorang salesman fiktif. Skema representasi dari persoalan akhirnya sebagai berikut, untuk n kota dan m salesman dipilih $N = (n+m-1)^2$ sel saraf dan membentuk sebuah matriks kota-posisi. Sebuah kota dipilih sebagai basis untuk menghitung jarak. Kemudian, dengan pendekatan TSP dicari pemecahan berupa rute valid dengan sebuah jarak minimal yang mengunjungi (n+m-1) buah kota dan kembali ke kota basis. Tabel 3.2 menampilkan contoh representasi untuk 10 kota, dan 4 salesman.

Rute valid terdiri atas m sub-rute berupa lintasan tertutup yang melibatkan kota basis. Dalam hal ini jika kota fiktif muncul ditengah-tengah rute, maka salesman tersebut harus kembali ke kota basis. Jadi seorang salesman akan mempunyai rute tersendiri. A dipilih sebagai kota basis, sedangkan K, L, dan M adalah kota fiktif. Rute pemecahan adalah ACELGFDJMBKIHA. Sehingga didapat sub-rute pemecahan ACEA, AGFDJA, ABA, dan AIHA. Dari Tabel 3.2 terlihat bahwa semua salesman berangkat dari kota basis, melengkapi rutenya dan kembali ke kota basis. Rute pertama mempunyai dua kota C dan E dalam rutenya, sedangkan rute ketiga hanya mempunyai satu kota yaitu B.

---

[12] E. Wacholder, J. Han and R.C. Momn, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem", Biol. Cybernet. 61, 11-19, 1989.

Tabel 3.2 Contoh Pemetaan Kota-posisi untuk MTSP

	1	2	3	4	5	6	7	8	9	10		11	12	13
A	1													
B										1				
C		1												
D							1							
E			1											
F						1								
G					1									
H														1
I													1	
J								1						

K	0											1		0
L	0			1										0
M	0								1					0

Berikut ini adalah pemecahan TSP dari Hopfield dan Tank yang didefinisikan untuk MTSP. Fungsi biaya yang memaksa semua pembatas seperti representasi pada gambar di atas memiliki bentuk kuadratik[13].

$$E_1 = \frac{1}{2}\sum_{k=1}^{n+m-1}\sum_{i=1}^{n+m-1}\sum_{\substack{j=1 \\ j\neq i}}^{n+m-1} v_{ki}v_{kj}, \quad \text{.....(3.4)}$$

$$E_2 = \frac{1}{2}\sum_{i=1}^{n+m-1}\sum_{k=1}^{n+m-1}\sum_{\substack{l=1 \\ l\neq k}}^{n+m-1} v_{ki}v_{li}, \quad \text{.....(3.5)}$$

$$E_3 = \frac{1}{2}\left[\left(\sum_{k=1}^{n+m-1}\sum_{i=1}^{n+m-1} v_{ki}\right) - (n+m-1)\right]^2, \quad \text{.....(3.6)}$$

[13] E. Wacholder, J. Han and R.C. Momn, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem", Biol. Cybernet. 61, 11-19, 1989.

$$E_4 = \frac{1}{2}\left[\sum_{k=n+1}^{n+m-1}\sum_{\substack{l=n+1\\ l\neq k}}^{n+m-1}\sum_{i=1}^{n+m-1} v_{ki}\sum_{s=1}^{r}(v_{l,i+s}+v_{l,i-s})\right], \quad (3.7)$$

$$E_5 = \frac{1}{2}\left[\left(\sum_{k=n+1}^{n+m-1}\sum_{i=1}^{n+m-1} v_{ki}\right)-(m-1)\right]^2, \quad (3.8)$$

$v_{1,1}=1,$

$v_{1,2}=v_{1,3}=\ldots=v_{1,n+m-1}=0,$

$v_{2,1}=v_{3,1}=\ldots=v_{n+m-1,1}=0,$

$v_{k,2}=v_{k,3}=\ldots=v_{k,r+1}=0, \quad (3.9)$

$v_{k,n+m-r}=v_{k,n+m+1-r}=\ldots=v_{k,n+m-1}=0, \quad \forall k=n+1,\ldots,n+m-1 \quad (3.10)$

E_1, E_2, dan E_3 sama seperti pada model TSP untuk membentuk matriks permutasi dan menjamin bahwa setiap kota (termasuk kota fiktif) hanya dikunjungi satu kali. Persamaan berikutnya diperlukan agar kota nomor 1 menjadi kota basis dan agar pemecahan mempunyai m lintasan tertutup, dengan setiap salesman paling sedikit mengunjungi r kota. Persamaan (3.8) berangkat dari kenyataan bahwa ada (m-1) '1' pada (m-1) baris terakhir yang mewakili kejadian salesman berangkat dan kembali ke kota basis.

Panjang rute diberikan sebagai berikut:

$$E_d = \frac{1}{2}\sum_{k=1}^{n+m-1}\sum_{\substack{l=1\\ l\neq k}}^{n+m-1}\sum_{i=1}^{n+m-1} d_{kl}v_{ki}(v_{l,i+1}+v_{l,i-1}), \quad (3.11)$$

dengan total fungsi biaya sebagai pembatas dari optimasi dirumuskan dengan *Lagrange Multiplier*:

$$E = E_d + \sum_{\alpha=1}^{5} \lambda_\alpha E_\alpha, \quad (3.12)$$

dan λ_α dibiarkan sebagai salah satu *state* dari 5 sel saraf tambahan dan mencoba menemukan nilai optimalnya daripada memberikan suatu nilai konstanta.

Persamaan diferensial yang mendefinisikan persamaan sel saraf diberikan sebagai berikut:

$$\frac{du_{ki}}{dt} = -\left(\frac{\partial E_d}{\partial v_{ki}} + \sum_{\alpha=1}^{5} \lambda_\alpha \frac{\partial E_\alpha}{\partial v_{ki}}\right), \quad (3.13)$$

$$\frac{d\lambda_\alpha}{dt} = E_\alpha, \quad (3.14)$$

dan agar konvergensi sistem mencapai titik keseimbangan yang stabil, persamaan sel saraf dimodifikasi sebagai berikut:

$$\frac{du_{ki}}{dt} = -\frac{u_{ki}}{\tau} - \left(\frac{\partial E_d}{\partial v_{ki}} + \sum_{\alpha=1}^{5} \lambda_\alpha \frac{\partial E_\alpha}{\partial v_{ki}}\right), \quad (3.15)$$

$$\frac{d\lambda_\alpha}{dt} = E_\alpha, \quad (3.16)$$

atau secara eksplisit setelah dideferensial menjadi:

$$\frac{du_{ki}}{dt} = -\left\{ \frac{u_{ki}}{\tau} + \frac{1}{2}\sum_{\substack{l=1 \\ l\neq k}}^{n+m-1} d_{kl}(v_{l,i+1} + v_{l,i-1}) + \frac{\lambda_1}{2}\sum_{\substack{j=1 \\ j\neq i}}^{n+m-1} v_{kj} + \frac{\lambda_2}{2}\sum_{\substack{l=1 \\ l\neq k}}^{n+m-1} v_{li} \right.$$

$$+ \lambda_3\left[\left(\sum_{l=1}^{n+m-1}\sum_{j=1}^{n+m-1} v_{lj}\right) - (n+m-1)\right] + \frac{\lambda_4}{2}\sum_{\substack{l=n+1 \\ l\neq k}}^{n+m-1}\sum_{s=1}^{r}(v_{l,i+s} + v_{l,i+s})$$

$$\left. + \lambda_5\left[\left(\sum_{l=n+1}^{n+m-1}\sum_{j=1}^{n+m-1} v_{ij}\right) - (m-1)\right]\right\}, \quad \text{.....(3.17)}$$

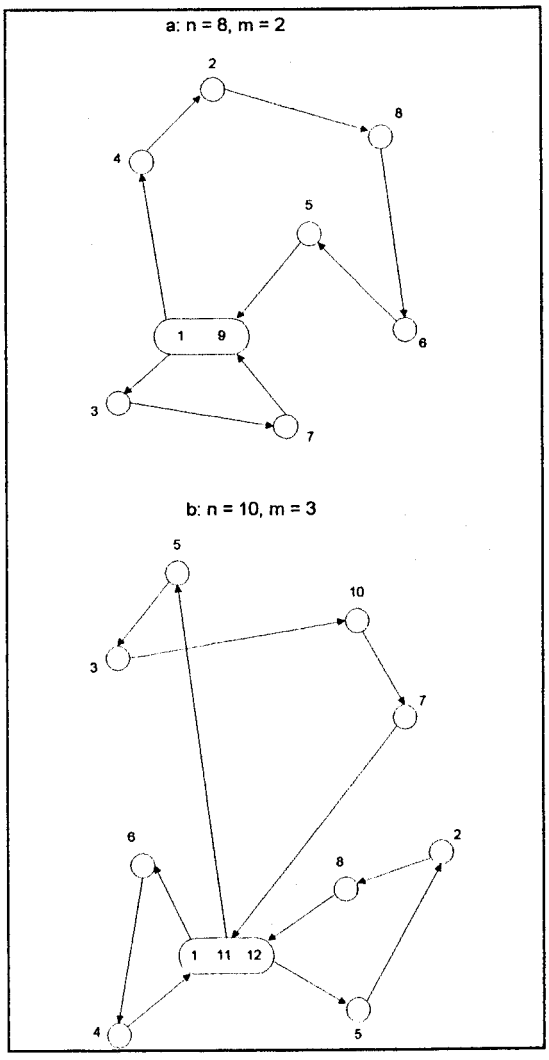
$\forall k, i = 1,2,...,n+m-1,$

$$\frac{d\lambda_\alpha}{dt} = E_\alpha,$$

$\forall \alpha = 1,2,...,5$ .....(3.18)

Gambar 3.2 menampilkan dua hasil simulasi dari MTSP. Kasus pertama dengan 8 kota dan 2 salesman serta kasus kedua 10 kota dengan 3 salesman. Persamaan sel saraf yang digunakan dalam simulasi tersebut menggunakan metode Euler orde satu dengan nilai awal yang sudah ditentukan. Konvergensi dari jaringan sensitif terhadap nilai awal dari *Lagrange multiplier*, $\lambda_\alpha(0)$. Riset lebih jauh menghendaki konvergensi yang baik. Memperkenalkan *Lagrange multiplier* ke dalam persamaan diferensial dari sel saraf juga merupakan deviasi dari penjumlahan bobot dari karakteristik input yang merupakan perilaku alami dari pemrosesan pararel terdistribusi dari jaringan saraf. Pada kasus ini variabel *Lagrange* mewakili beberapa penggerak umum yang memodulasi penjumlahan bobot dari sinyal input, dan variasinya harus diperlakukan sebagai *long term memory* daripada dengan *short term memory*.

Pada banyak kasus perluasan dari perumusan TSP menjadi MTSP akan memberikan dasar untuk persoalan penjadualan pada proses manufaktur yang melibatkan banyak pekerjaan dan banyak mesin, dengan beberapa perumusan yang diperlukan sabagai pembatas. Pembahasan berikut menyelidiki perumusan dari persoalan penjadualan.



Gambar 3.2
Beberapa Contoh Hasil Simulasi MTSP

3.3 Penjadualan Banyak Pekerjaan[14]

Seperti dijelaskan dalam sub bab sebelumnya, persoalan penjadualan memiliki kemiripan dengan persoalan MTSP dimana *n* buah kota identik dengan *n* buah pekerjaan dan *m* buah salesman identik dengan *m* buah mesin. Bagian ini didahului dengan penjelasan mengenai deskripsi persoalan penjadualan banyak pekerjaan. Kemudian, dilanjutkan dengan penjelasan mengenai perumusan persoalan dalam bentuk *neuro-box network* (NBN)

3.3.1 Deskripsi Persoalan Penjadualan Banyak Pekerjaan

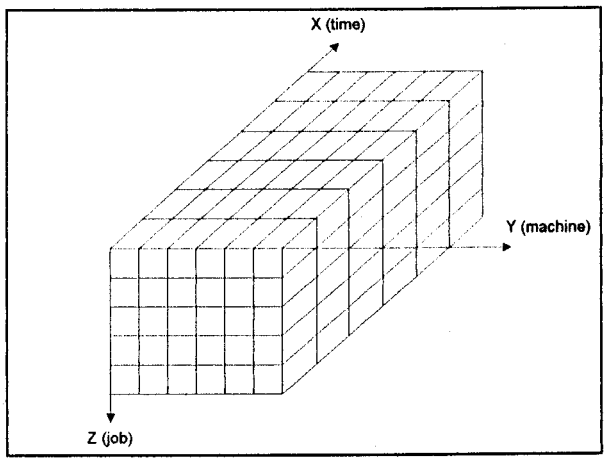
Persoalan penjadualan banyak pekerjaan (*multiple job scheduling problem,* MJSP) merupakan perluasan dari MTSP dengan tingkat kesulitan yang lebih tinggi lagi. Untuk memperjelas persoalan yang ada, diasumsikan terdapat *n* buah pekerjaan yang masing-masing dinotasikan dengan *Ji* (i=1,2,...,n). Masing-masing pekerjaan diasumsikan memiliki tipe yang sama dan panjang proses yang berbeda. Sejumlah pekerjaan tadi akan diproses pada *m* buah mesin yang dinotasikan dengan *Mj* (j=1,2,...,m) dimana masing-masing mesin diasumsikan memiliki tipe yang sama, tetapi dengan kapasitas yang berbeda. Setiap mesin juga diasumsikan hanya dapat memroses paling banyak satu pekerjaan dalam satu alokasi waktu dan tidak terjadi kerusakan selama perencanaan.

---

[14] Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling with Artificial Neural Network", University of California, Irvine, 1991.

Permasalahannya adalah bagaimana menentukan urutan dari pekerjaan pada setiap mesin yang mempunyai biaya dan waktu minimal untuk menyelesaikan semua pekerjaan. Pada jadual akhir setiap pekerjaan hanya berkorespondensi dengan satu mesin. Sehingga mesin yang diinginkan harus dapat mengerjakan paling sedikit satu pekerjaan. Setiap pekerjaan memiliki jatuh tempo (*due date*) dan dinotasikan dengan d_i di mana pekerjaan tersebut harus diselesaikan atau jika tidak maka sebuah penalti akan dikenakan pada pekerjaan tersebut.

Pemecahan untuk persoalan m buah mesin dan dan n buah pekerjaan ini terdiri atas sebuah daftar terurut dari n pekerjaan dan m mesin. Untuk memetakan persoalan ini ke dalan jaringan saraf, jaringan Hopfield 2-D untuk TSP dan MTSP dikembangkan menjadi sebuah jaringan saraf 3-D yang dikenal dengan *neuro-box network* (NBN) seperti ditunjukkan pada Gambar 3.3, dimana sel saraf beroperasi pada *high gain limit* untuk mengkonvergensi ke bentuk digital output (1 dan 0). Pengembangan dari NBN ini didorong oleh pendekatan MTSP, tetapi daripada menambahkan baris dan kolom fiktif pada topologi persoalan 2-D, sebuah dimensi ketiga yang membawa faktor waktu (*time*) pada penjadualan ditambahkan dalam pendekatan ini. Sumbu z mewakili pekerjaan, sumbu y mewakili mesin, dan sumbu x mewakili alokasi waktu. Ketiga sumbu ini merupakan unit diskrit, sehingga setiap unit menyajikan pekerjaan, mesin, dan alokasi waktu yang berkorespondensi. Unit waktu tergantung pada persoalan yang ditangani, mungkin bisa hari, minggu atau jam. Jadual akhir harus dikerjakan dalam satuan waktu tersebut.



Gambar 3.3
Jaringan Saraf 3D untuk Penjadualan

Representasi ini memberikan dasar penjadualan *on-demand* yang dinamis, karena sumbu x yang mewakili alokasi waktu merupakan variabel utama dalam perumusan persoalan ini. Ketika suatu pekerjaan baru datang atau ada perubahan dalam batas waktu yang diperlukan untuk pekerjaan-pekerjaan tersebut, NBN akan diinisialisasi dan disimulasi untuk menemukan jadual baru untuk memenuhi permintaan, dan seiring dengan bergeraknya waktu NBN berubah mengikuti waktu. Karena jumlah mesin dan jumlah pekerjaan diketahui, sejumlah waktu yang dibutuhkan dialokasikan untuk penjadualan tersebut, sehingga dimensi dari NBN terbentuk. Setiap kubus adalah sebuah sel saraf pada jaringan dan mungkin mempunyai koneksi *excitory* atau *inhibitory* untuk semua sel saraf lain pada NBN. Jika *state* dari sel saraf v_{ijl} adalah satu, maka pekerjaan J_j diberikan pada mesin M_i pada alokasi waktu yang ke- l.

Untuk representasi persoalan ini maka pada bidang xy untuk setiap pekerjaan seharusnya hanya ada satu sel saraf pada *state* 1 dan sisanya pada *state* 0, karena suatu pekerjaan tertentu harus diberikan pada sebuah mesin pada waktu

mulai (*starting time*) yang diberikan. Pada bidang *yz* terdapat paling banyak *m* buah (jumlah mesin) sel saraf *high* dengan syarat bahwa hanya ada satu sel saraf pada baris atau kolom yang dapat *high*, karena pada waktu yang ditentukan, pekerjaan J_j hanya dapat diberikan pada satu mesin (pembatas dari baris) dan dua pekerjaan tidak dapat diberikan pada satu mesin (pembatas dari kolom). Selanjutnya, bidang *xz*, yang menampilkan jadual dari sebuah mesin, paling banyak hanya mempunyai satu sel saraf *high* pada baris ke-*l*, karena setiap pekerjaan hanya ditugaskan satu kali (pembatas baris) dan dua pekerjaan tidak bisa dimulai pada waktu yang sama (pembatas kolom). Karena setiap mesin harus mengerjakan paling sedikit satu pekerjaan, paling sedikit satu sel saraf harus *high* pada bidang *zx*. Juga, jika ada lebih dari satu pekerjaan diberikan pada suatu mesin, perbedaan posisi dari pekerjaan tersebut pada sumbu *x* seharusnya lebih besar dari panjang waktu yang diambil oleh pekerjaan yang pertama untuk menyelesaikan pekerjaanya pada mesin ini. Jumlah total dari sel saraf bernilai '1' pada NBN harus sama dengan *n*, yaitu jumlah pekerjaan.

Tabel 3.3 menampilkan contoh penugasan untuk 2 mesin, 4 pekerjaan dengan 6 alokasi waktu. Jadual untuk M_1 adalah J_2 dan J_3, masing-masing membutuhkan 3 alokasi waktu untuk menyelesaikan pekerjaannya. Jadual untuk M_2 adalah J_4 dan J_1 dan masing-masing membutuhkan 2 dan 4 alokasi waktu.

Tabel 3.3
Sebuah Contoh Penugasan untuk 2 Mesin, 4 Pekerjaan, dan 6 Unit Alokasi Waktu

J1	0	0	0	0	0	0
J2	1	0	0	0	0	0
J3	0	0	0	1	0	0
J4	0	0	0	0	0	0

J1	0	0	1	0	0	0
J2	0	0	0	0	0	0
J3	0	0	0	0	0	0
J4	1	0	0	0	0	0

3.3.2 Perumusan Neuro-Box Network (NBN)

Berdasarkan penurunan fungsi biaya untuk TSP dan MTSP seperti dijelaskan sebelumnya, berikut ini dirumuskan fungsi biaya untuk NBN, sehingga keadaan jaringan dengan nilai fungsi biaya paling rendah berkorespondensi dengan jadual yang paling baik. Waktu yang minimal dipilih sebagai fungsi tujuan (*object function*). Penjelasan berikut akan mendaftar semua tahap fungsi energi dari jaringan. E_t fungsi tujuan, yang meminimalkan waktu yang dibutuhkan untuk menyelesaikan semua pekerjaan, diberikan sebagai berikut:

$$E_t = \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}\left(\frac{l+T_{ij}-1}{C_k}\right), \quad \text{.....(3.19)}$$

sedangkan fungsi energi untuk pembatas baris dan kolom diberikan sebagai berikut:

$$E_1 = \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{k}\sum_{j=1}^{m}\sum_{\substack{l_1=1\\ l_1\neq l}}^{k}\sum_{\substack{j_1=1\\ j_1\neq j}}^{m} v_{ijl}\left(v_{ij_1l} + v_{ijl_1} + v_{ij_1l_1}\right), \quad \text{.....(3.20)}$$

$$E_2 = \frac{1}{2}\sum_{j=1}^{m}\sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{\substack{i_1=1\\ i_1\neq i}}^{n} v_{ijl}v_{i_1jl}, \quad \text{.....(3.21)}$$

$$E_3 = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k} v_{ijl} - n\right)^2, \quad \text{.....(3.22)}$$

$$E_4 = \frac{1}{2}\sum_{j=1}^{m} f^2{}_j H(f_j), \quad \text{.....(3.23)}$$

$$E_5 = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij_1} - m\right)^2, \quad \text{.....(3.24)}$$

$$E_6 = \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}\, g^2{}_{ijl}\, H(g_{ijl}), \quad \text{.....(3.25)}$$

$$E_7 = \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{k}\sum_{\substack{i_1=1\\ i_1\neq i}}^{n} v_{ijl}\left[h^2{}_1 H(h_1)\sum_{l_1=l}^{k} v_{i_1 j l_1} + h^2{}_2 H(h_2)\sum_{\substack{l_2=1\\ l_2\neq l}}^{l-1} v_{i_1 j l_2} \right], \quad \text{.....(3.26)}$$

$$T_{ij} = \frac{L_i}{m_j},$$

$$f_j = 1 - \sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl},$$

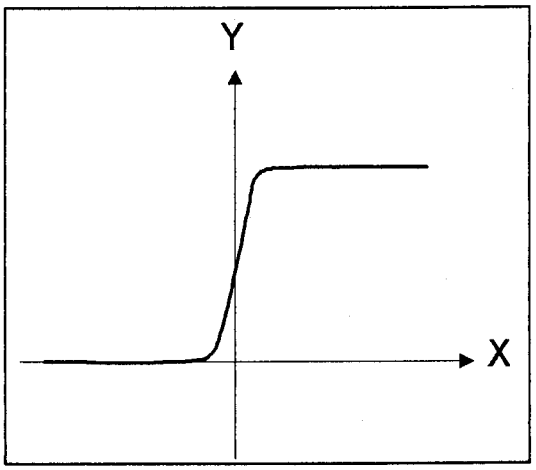
$$g_{ijl} = l - 1 + T_{ij} - d_i,$$

$$h_1 = (T_{ij} + l) - \sum_{l_1=l}^{k} l_1 v_{i_1 j l_1},$$

$$h_2 = \sum_{\substack{l_2=1\\ l_2\neq l}}^{l-1} (T_{i_1 j} + l_2) v_{i_1 j l_2} - l,$$

dimana *i, j, l, i_1, j_1, l_1, l_2* adalah indek sedangkan C_k adalah konstanta untuk skala, v_{ijl} adalah output sel saraf, m_j adalah rate operasi dari mesin ke-*j*. *Li* adalah panjang pekerjaan ke-*i*. *Tij* adalah waktu yang dibutuhkan oleh mesin ke-*j* untuk menyelesaikan pekerjaan ke-*i*. d_i adalah jatuh tempo pekerjaan ke-*i*. *H(f)* adalah fungsi Heaviside dengan sebuah nilai sama dengan satu jika $f > 0$; dan sebaliknya bernilai nol. Untuk metode gradien proyeksi dibutuhkan aproksimasi *H(f)*, yang juga perlu dapat dideferensial. Pendekatan yang dipakai adalah menggunakan fungsi sigmoid *high-gain*. Aproksimasi bisa dilihat pada Gambar 3.4, dan dihitung dengan rumus:

$$H(f) = \frac{1}{1 + e^{-\gamma f}}, \quad \text{.....(3.27)}$$



Gambar 3.4 Fungsi Sigmoid *High Gain*

dimana γ adalah konstanta. Pembatas (3.20), (3.21), (3.21), dan (3.23) diperlukan untuk membentuk matriks permutasi. Pembatas tersebut menjamin bahwa setiap pekerjaan hanya ditugaskan pada satu mesin dan setiap mesin paling sedikit mengerjakan satu pekerjaan dalam seluruh urutan penjadualan. Pembatas (3.24) menjamin bahwa setiap mesin memerlukan satu pekerjaan pada alokasi waktu ke-1. Pembatas (3.25) memaksa setiap mesin menyelesaikan pekerjaan sesuai jatuh tempo. Pembatas (3.26) menjamin bahwa jika pekerjaan ke-*i* ditugaskan pada mesin ke-*j* pada alokasi waktu ke-*l*, maka tidak ada pekerjaan lain ditugaskan pada mesin tersebut sampai $l + Tij$, dan juga pekerjaan ke-i_1 tidak dapat ditugaskan sebelum $l_2 + Ti_1j$.

Sekali lagi, persoalan optimasi yang mempunyai pembatas dikonversi dalam bentuk tidak mempunyai pembatas dengan memperkenalkan *Lagrange multiplier* λ_α seperti pada pemecahan MTSP[15], λ_α dibiarkan sebagai *state* dari

[15] E. Wacholder, J. Han and R.C. Momn, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem", Biol. Cybernet. 61, 11-19, 1989.

tujuh sel saraf tambahan. Tujuannya adalah untuk meminimalkan total fungsi biaya, yaitu:

$$E = E_t + \sum_{\alpha=1}^{7} \lambda_\alpha E_\alpha \quad \text{.....(3.28)}$$

Kemudian, dengan mengaplikasikan teknik gradien didapatkan sebuah sistem persamaan diferensial biasa:

$$\frac{du_{ijl}}{dt} = -\left(\frac{\partial E_t}{\partial v_{ijl}} + \sum_{\alpha=1}^{7} \lambda_\alpha \frac{\partial E_\alpha}{\partial v_{ijl}} \right), \text{.....(3.29)}$$

Untuk membuat sistem konvergen ke suatu titik keseimbangan yang stabil, persamaan (3.29) dan (3.30) dimodifikasi seperti pada MTSP, yaitu menguatnya sebuah tahap damping akan memecahkan persamaan (3.30) seiring waktu:

$$\frac{du_{ijl}}{dt} = -\frac{u_{ijl}}{\tau} - \left(\frac{\partial E_t}{\partial v_{ijl}} + \sum_{\alpha=1}^{7} \lambda_\alpha \frac{\partial E_\alpha}{\partial v_{ijl}} \right), \text{.....(3.31)}$$

$$\frac{d\lambda_\alpha}{dt} = -E_\alpha, \forall \alpha = 1,2,...,7, \text{.....(3.32)}$$

dengan menurunkan E_t dan $E_1,...,E_7$, didapatkan persamaan diferensial sel saraf sebagai berikut:

$$\frac{du_{ijl}}{dt} = -\frac{u_{ijl}}{\tau} - \frac{l + T_{ij} - 1}{C_k} - \lambda_1 \sum_{\substack{j_1=1 \\ j_1 \neq j}}^{m} \sum_{\substack{l_1=1 \\ l_1 \neq l}}^{k} (v_{ij_1l} + v_{ijl_1} + v_{ij_1l_1})$$

$$- \lambda_2 \sum_{\substack{i_1=1 \\ i_1 \neq i}}^{n} v_{i_1jl} - \lambda_3 \left(\sum_{i_1=1}^{n} \sum_{j_1=1}^{m} \sum_{l_1=1}^{k} v_{i_1j_1l_1} - n \right)$$

$$+ \lambda_4 \left[f_j H(f_j) + \frac{1}{2} H^2(f_j) f^2{}_j A e^{-Af_j} \right]$$

$$- \lambda_5 \left(\sum_{i_1=1}^{n} \sum_{j_1=1}^{m} v_{i_1j_1l} - m \right) \delta(l-1) - \lambda_6 H(g_{ijl}) g^2{}_{ijl}$$

$$- \lambda_7 \sum_{\substack{i_1=1 \\ i_1 \neq i}}^{n} \left[h^2{}_1 H(h_1) \sum_{l_1=1}^{k} v_{i_1jl_1} + h^2{}_2 H(h_2) \sum_{\substack{l_2=1 \\ l_2 \neq l}}^{l-1} v_{i_1jl_2} \right], \quad (3.33)$$

$$\frac{d\lambda_\alpha}{dt} = -E_\alpha, \forall \alpha = 1,2,...,7, \quad (3.34)$$

dimana f_j, g_{ijl}, T_{ij}, dan h_2 sudah didefinisikan sebelumnya, sedangkan:

$$\delta(l-1) = \begin{cases} 1 & l=1, \\ 0 & otherwise. \end{cases}$$

$$v_{ijl} = \frac{1}{2}\left[1 + \tanh\left(\frac{u_{ijl}}{u_0}\right)\right],$$

Dalam hal ini, hubungan antara sel saraf-sel saraf tidak simetris seperti pada jaringan Hopfield sehingga global konvergensi tidak bisa dijamin, walau demikian konvergensi ke beberapa sub optimal pemecahan valid dan dapat diterima untuk beberapa keperluan praktis.

Persamaan-persamaan ini disimulasikan menggunakan hasil dari teknik BDMM[16]. Teknik tersebut menunjukkan konvergensi dari persoalan optimasi umum berikut ini dengan kondisi-kondisi tertentu. Untuk beberapa persoalan optimasi umum diberikan sebagai berikut:

$$\varepsilon = f(x) + \sum_{\alpha} \lambda_{\alpha} g_{\alpha}(x), x \in \Re^{n}, \quad \text{.....(3.35)}$$

yang menghasilkan persamaan diferensial:

$$\frac{dx_i}{dt} = -\frac{\partial f}{dx_i} - \sum_{\alpha} \lambda_{\alpha} \frac{\partial g_{\alpha}}{\partial x_i}, \quad \text{.....(3.36)}$$

$$\frac{d\lambda_{\alpha}}{dt} = g_{\alpha}(x), \quad \text{.....(3.37)}$$

dengan total energi sistem dapat dikonstruksi melalui penjumlahan dari energi kinetik dan energi potensial:

$$E = \sum_{i} \frac{1}{2}\left(\frac{dx_i}{dt}\right)^2 + \sum_{\alpha} \frac{1}{2}[g_{\alpha}(x)]^2, \quad \text{.....(3.38)}$$

dengan derivasi waktu dari E diberikan sbb:

$$\frac{dE}{dt} = \sum_{i} \frac{d^2 x_i}{dt^2}\left(\frac{dx_i}{dt}\right) + \sum_{\alpha} g_{\alpha}(x) \frac{\partial g_{\alpha}}{\partial x_i} \frac{dx_i}{dt}, \quad \text{.....(3.39)}$$

$$= -\sum_{ij} \frac{dx}{dt}\left(\frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{\alpha} \lambda_{\alpha} \frac{\partial^2 g_{\alpha}}{\partial x_i \partial x_j}\right)\frac{dx_j}{dt}, \quad \text{.....(3.40)}$$

$$= -\sum_{ij} \frac{dx_i}{dt} A_{ij} \frac{dx_j}{dt}, \quad \text{.....(3.41)}$$

---

[16] C. P. John and H. B. Alan, "Constrained Differential Optimization", pp. 612-621, American Institute of Physics, 1988.

Jika matriks damping A_{ij} positif definitif, maka sistem akan konvergen ke sebuah pemecahan valid yang memuaskan semua pembatas.

Algoritma penjadualan n buah pekerjaan dan m buah mesin dengan jaringan saraf hanya sebuah kasus dari persoalan optimasi di atas. Jadi konvergensi dijamin. Simulasi-simulasi sudah menunjukkan konvergensi dari jaringan. Derivasi dari range A_{ij} memungkinkan pencapaian konvergensi dan kondisi secara langsung.

BAB IV

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Bab ini menjelaskan bagaimana perangkat lunak penjadualan banyak pekerjaan dikembangkan. Penjelasan ini meliputi kebutuhan sistem, deskripsi sistem, perancangan, dan implementasinya. Perangkat lunak dibuat dengan menggunakan bahasa pemrograman *C*, dalam hal ini adalah *Borland C++ Builder Professional*[17], yang dijalankan pada sistem operasi *Windows 95*.

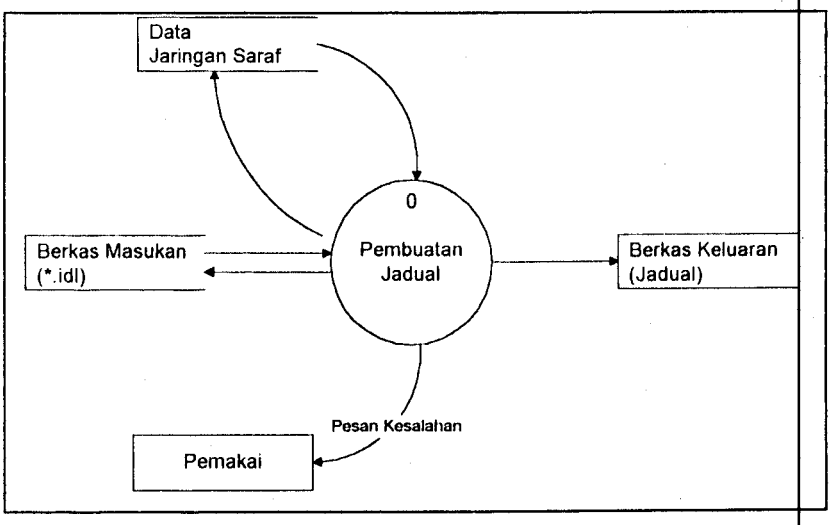
4.1 Kebutuhan Sistem

Perangkat lunak yang dibuat hanya dapat berjalan pada sistem operasi *Windows 95* dan memerlukan *run-time-library* berupa file-file dalam direktori *vcl* yang disediakan oleh *Borland C++ Builder Professional*. Memori yang dibutuhkan cukup besar karena harus bisa menampung matriks tiga dimensi sebagai implementasi dari sebuah topologi jaringan saraf tiga dimensi. Selanjutnya manajemen memori diserahkan pada sistem operasi *Windows 95*. Kebutuhan minimum yang diperlukan agar waktu eksekusi relatif lebih cepat adalah komputer dengan prosesor *Pentium 133 MHz* dengan *16 MB* memori.

[17] http://www.inprise.com

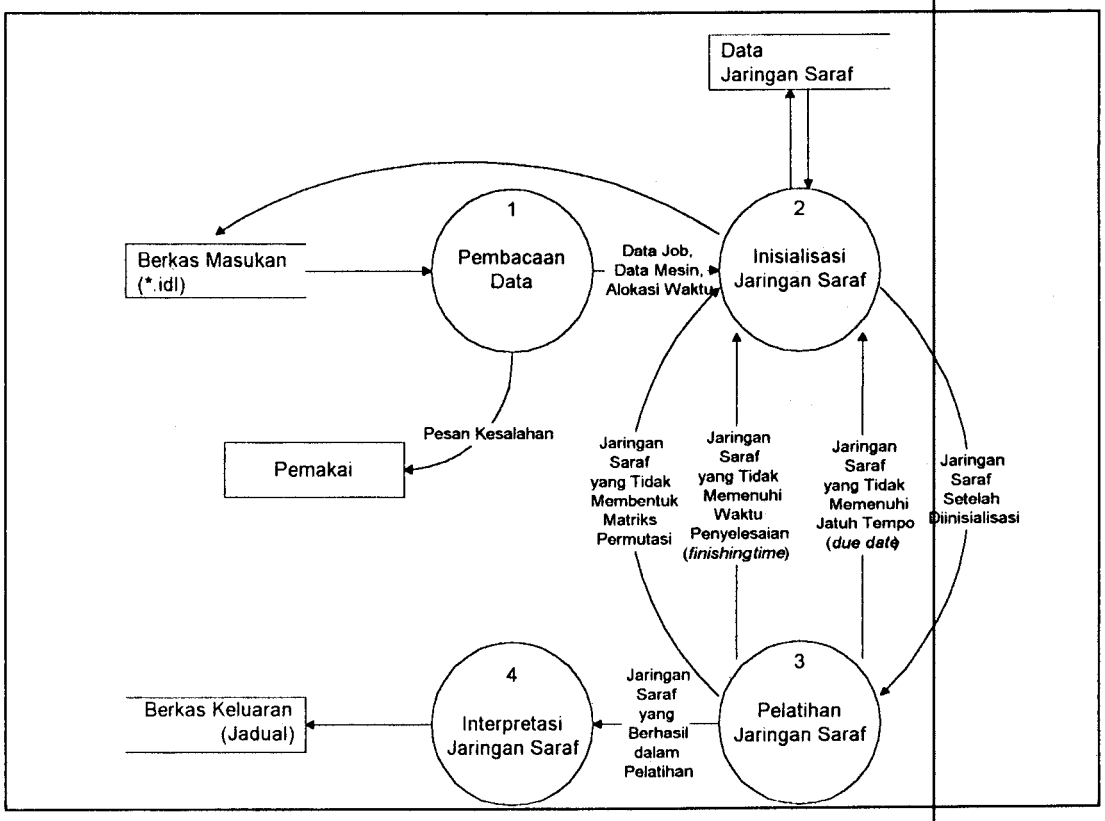
4.2 Deskripsi Sistem

Deskripsi sistem perangkat lunak dijelaskan dengan menggunakan diagram aliran data (*data flow diagram*, DFD) berikut ini. Gambar 4.1 menampilkan DFD level 0 dari aplikasi Penjadualan Banyak Pekerjaan dengan Jaringan Saraf Tiruan. Aplikasi menerima berkas masukan berupa file teks dan menghasilkan berkas keluaran yang berisi jadual urutan pemrosesan. Dalam proses pembuatan jadual, sistem membaca data jaringan saraf dan mengirim pesan kesalahan kepada pemakai jika dalam proses tersebut terdapat kesalahan. Berkas masukan berupa file teks yang berisi data jumlah pekerjaan, data mesin, dan alokasi waktu. Data pekerjaan terdiri dari jumlah pekerjaan, panjang proses dan jatuh tempo masing-masing pekerjaan. Data mesin terdiri dari jumlah mesin dan kapasitas masing-masing mesin. Berkas keluaran yang dihasilkan dapat ditampilkan dalam bentuk tabel ataupun dicetak printer.



Gambar 4.1 DFD Level 0, Aplikasi Penjadualan Banyak Pekerjaan dengan Jaringan Saraf Tiruan

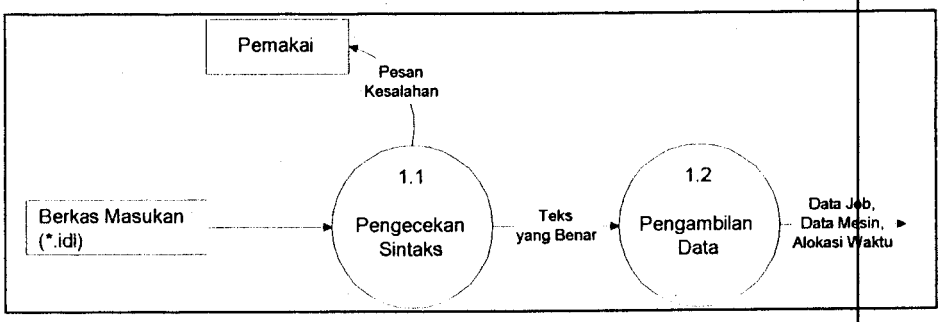
Penjelasan lebih detail dapat dilihat dalam Gambar 4.2. Proses pembuatan jadual di atas diuraikan dalam bentuk DFD level 1. Proses pembuatan jadual terdiri atas empat proses utama, yaitu pembacaan data, inisialisasi jaringan saraf, pelatihan jaringan saraf, dan interpretasi jaringan saraf. Proses pelatihan dilakukan melalui proses inisialisasi jaringan saraf berdasarkan pada data jaringan saraf dan data hasil proses pembacaan data (yaitu data pekerjaan, data mesin, dan alokasi waktu).



Gambar 4.2 DFD Level 1, Pembuatan Jadual

Dalam proses pembacaan data dilakukan proses pengecekan sintaks. Data teks dari berkas masukan dicek sintaksnya sebelum dilakukan pengambilan data. Jika ada kesalahan maka sistem mengirim pesan kesalahan kepada pemakai. Jadi

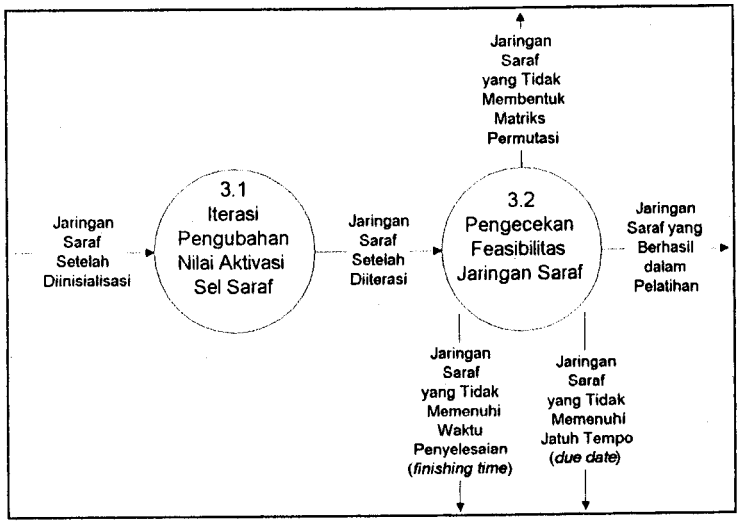
proses pengambilan data hanya dilakukan pada berkas masukan yang sudah benar. Hasil dari proses pengambilan data adalah data pekerjaan, data mesin, dan alokasi waktu. Data pekerjaan tersebut berupa panjang proses dan jatuh tempo dari masing-masing pekerjaan. Sedangkan data mesin tersebut berupa kapasitas dari masing-masing mesin. Gambar 4.3 menjelaskan alur data dan alur proses dalam pembacaan data.



Gambar 4.3 DFD Level 2, Pembacaan Data

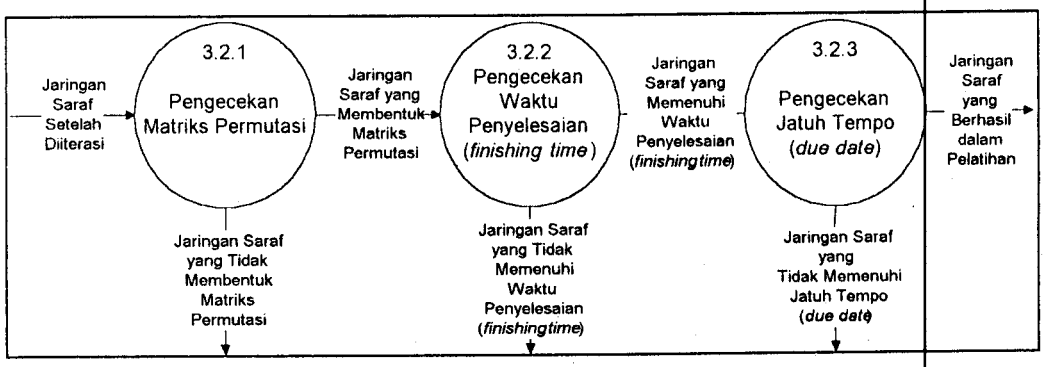
Proses pelatihan jaringan saraf diuraikan melalui DFD level 2 dalam Gambar 4.4. Proses pelatihan ini mempunyai dua proses utama, yaitu iterasi pengubahan nilai aktivasi sel saraf dan pengecekan feasibilitas jaringan saraf. Jumlah iterasi yang dilakukan dalam proses pengubahan nilai aktivasi sel saraf tergantung pada kompleksitas permasalahan yang ditangani. Iterasi dilakukan untuk membuat sistem konvergen. Jumlah iterasi dibatasi untuk menghindari komputasi yang sia-sia. Salah satu syarat suatu sistem konvergen adalah selisih energi dari setiap sel saraf pada iterasi sekarang dengan iterasi sebelumnya sangat kecil atau nol. Dalam kasus ini untuk mencapai kondisi tersebut dibutuhkan jumlah iterasi yang besar. Padahal dengan sejumlah iterasi yang tidak terlalu besar dapat diketahui kecenderungan dari sistem. Jika sistem divergen atau konvergen

tetapi tidak dalam daerah yang *feasibel* maka dilakukan iterasi ulang. Jika jumlah iterasi tidak memungkinkan suatu feasibilitas tercapai maka jumlah iterasi dapat ditambah. Seperti dijelaskan dalam Gambar 4.4 proses pengecekan feasibilitas jaringan saraf dilakukan setelah proses iterasi pengubahan nilai aktivasi sel saraf. Jika feasibilitas tidak terpenuhi maka dilakukan iterasi ulang. Dengan kata lain proses pelatihan baru berhenti jika feasibilitas jaringan saraf terpenuhi atau sistem cenderung divergen. Jika sistem cenderung divergen maka dilakukan pengubahan data jaringan saraf dalam proses inisialisasi.



Gambar 4.4 DFD Level 2, Pelatihan Jaringan Saraf

Proses pengecekan feasibilitas jaringan saraf diuraikan melalui DFD level 3 dalam Gambar 4.5. Proses ini terdiri atas tiga tahap pengecekan, yaitu pengecekan matriks permutasi, pengecekan waktu penyelesaian (*finishing time*), dan pengecekan jatuh tempo (*due date*). Jika salah satu pengecekan menghasilkan nilai salah maka sistem kembali ke proses inisialisasi. Kemudian dilakukan proses pelatihan kembali sampai menemukan suatu nilai yang *feasibel*.



Gambar 4.5 DFD Level 3, Pengecekan Feasibilitas Jaringan Saraf

Pengecekan matriks permutasi dilakukan untuk memenuhi syarat kondisi akhir dari jaringan. Jumlah sel saraf yang aktif dalam kondisi akhir harus sama dengan jumlah pekerjaan yang ditangani. Kemudian untuk setiap arah dalam matriks tiga dimensi tersebut jumlah sel saraf yang aktif sama dengan satu atau nol. Jadi pengertian matriks permutasi disini agak berbeda dengan pengertian matriks permutasi pada umumnya, yaitu matriks bujursangkar dimana jumlah nilai untuk setiap baris dan kolom sama dengan satu. Pada persoalan ini matriks permutasi merupakan matriks permutasi yang sudah dimodifikasi. Karena matriks tersebut merupakan matriks tiga dimensi dan strukturnya tidak mesti berbentuk kubus. Sehingga ada baris atau kolom dalam ketiga arah matriks jumlah nilainya sama dengan nol.

Pengecekan waktu penyelesaian dilakukan jika jaringan saraf membentuk matriks permutasi. Jika ada satu pekerjaan yang tidak memenuhi waktu penyelesaian maka dilakukan iterasi ulang terhadap jaringan saraf. Kondisi ini menunjukkan bahwa solusi yang diperoleh tidak *feasibel* walaupun jaringan saraf konvergen. Solusi yang diperoleh merupakan lokal optimal tetapi tidak berada di dalam daerah yang *feasibel* karena ada pembatas yang masih dilanggar.

Pengecekan jatuh tempo berperilaku sama dengan pengecekan waktu penyelesaian. Suatu jadual mungkin terbentuk tetapi jatuh temponya dilanggar. Dalam perancangan perangkat lunak ini sistem dibuat untuk menghasilkan solusi yang memenuhi semua pembatas. Jika ada salah satu pembatas yang tidak terpenuhi maka dilakukan proses pelatihan kembali dengan didahului proses inisialisasi untuk mengubah data jaringan saraf atau berkas masukan jika diperlukan.

4.3 Perancangan Perangkat Lunak

Dalam bagian ini dijelaskan mengenai perancangan sistem perangkat lunak. Perancangan tersebut meliputi spesifikasi dan pendekatan yang digunakan dalam sistem perangkat lunak.

4.3.1 Perancangan Data Masukan

Perangkat lunak ini menggunakan data masukan berupa file teks. Informasi yang disimpan adalah jumlah pekerjaan, panjang proses masing-masing pekerjaan, jatuh tempo masing-masing pekerjaan, jumlah mesin, kapasitas masing-masing mesin dan alokasi waktu yang digunakan. Tata bahasa yang digunakan dalam penulisan data masukan dijelaskan dalam Gambar 4.6.

S	→ *header P*
header	→ *"MJSP_ Application_ (*.idl)"*
P	→ *job machine time*
job	→ *numjob joblen jobduedate*
numjob	→ *int*
joblen	→ *int A*
A	→ *int* | ε
jobduedate	→ *int A*
machine	→ *nummachine machinecap*
nummachine	→ *int*
machinecap	→ *int A*
time	→ *numtime*

Gambar 4.6 Tata Bahasa yang Digunakan dalam Data Masukan

Identitas file disimpan dalam header file (*"MJSP_ Application_ (*.idl)"*). Data pekerjaan, data mesin, dan alokasi waktu, masing-masing merupakan bilangan bulat lebih besar dari nol.

4.3.2 Perancangan Data Keluaran

Perangkat lunak ini menampilkan data keluaran berupa jadual yang diperoleh berdasarkan data masukan yang diberikan. Selanjutnya pemakai dapat menyimpannya dalam suatu berkas teks. Jadual disajikan dalam bentuk urutan-urutan pekerjaan pada setiap mesin dan ditampilkan dengan tabel. Perubahan nilai dari *Lagrange multiplier* juga ditampilkan pada saat proses pelatihan. Pemakai juga dapat memperoleh informasi lain melalui pengaturan pilihan. Informasi itu adalah nilai aktivasi masing-masing sel saraf dan penugasan pekerjaan (*job assignment*).

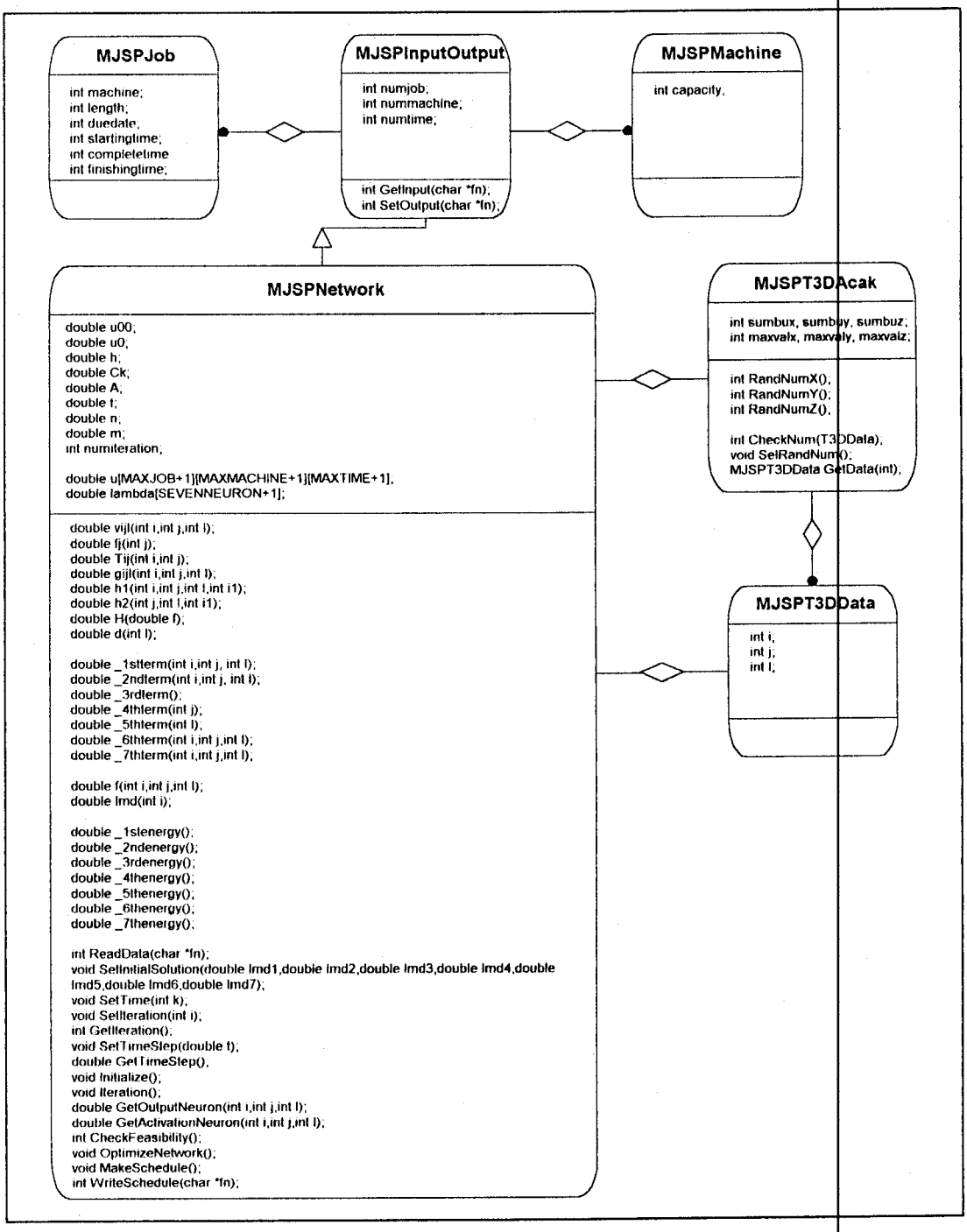
4.3.3 Perancangan Obyek dan Implementasi Struktur Data

Dalam bagian ini dijelaskan mengenai perancangan obyek sistem. Pendekatan yang digunakan adalah pemrograman berorientasi obyek (*Object Oriented Programming*)[18]. Perancangan dideskripsikan dengan menggunakan *C++-like*. Obyek-obyek yang terlibat dalam perangkat lunak ini ditunjukkan dalam Gambar 4.7. Obyek-obyek tersebut adalah *MJSPJob*, *MJSPMachine*, *MJSPInputOutput*, *MJSPNetwork*, *MJSPT3DAcak*, dan *MJSPT3DData*. Berikut ini dijelaskan tentang masing-masing obyek berikut atribut yang dimiliki dan operasi yang ditangani. Beberapa atribut dan operasi mengacu pada persamaan matematis dalam bab sebelumnya. Penjelasan tentang penamaannya bisa dilihat dalam bab tersebut.

MJSPJob merupakan representasi dari pekerjaan, *MJSPMachine* merupakan representasi dari mesin, *MJSPInputOutput* merupakan representasi dari input/output perangkat lunak, dan *MJSPNetwork* merupakan representasi dari sistem jaringan saraf yang digunakan perangkat lunak. *MJSPT3DAcak* digunakan untuk membangkitkan dan menyimpan sejumlah bilangan acak. *MJSPT3DData* digunakan untuk meyimpan data acak indek tiga dimensi dari matriks.

---

[18] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, "Object-Oriented Modelling and Design", Prentice Hall, Englewood Cliffs, New Jersey 07632, 1991.



Gambar 4.7 Perancangan Obyek

MJSPJob mempunyai atribut *machine*, *length*, *duedate*, *startingtime*, *completetime* dan *finishingtime*. Atribut *machine* digunakan untuk menyimpan indeks mesin dimana pekerjaan tersebut diproses, atribut *length* digunakan untuk menyimpan nilai panjang proses, atribut *duedate* digunakan untuk menyimpan data jatuh tempo, atribut *startingtime* digunakan untuk menyimpan waktu mulai proses, atribut *completetime* digunakan untuk menyimpan waktu yang dibutuhkan untuk menyelesaikan pekerjaan berdasarkan kapasitas mesin yang menangani, dan atribut *finishingtime* digunakan untuk menyimpan waktu penyelesaian proses. *MJSPMachine* mempunyai atribut *capacity* untuk menyimpan data kapasitas mesin. *MJSPInputOutput* mempunyai atribut *numjob*, *nummachine*, dan *numtime* serta mempunyai operasi *GetInput* dan *SetOutput*. Atribut *numjob* digunakan untuk menyimpan nilai jumlah pekerjaan yang ditangani, atribut *nummachine* digunakan untuk menyimpan nilai jumlah mesin yang memroses pekerjaan-pekerjaan tersebut, dan atribut *numtime* untuk menyimpan nilai jumlah waktu yang dialokasikan. Atribut *Job* dan *Machine* adalah sebuah matriks satu dimensi yang terdiri dari obyek *MJSPJob* dan *MJSPMachine*. Operasi *GetInput* digunakan untuk mengambil data input dan operasi *SetOutput* digunakan untuk mengatur output. *MJSPInputOutput* terdiri dari *MJSPJob* dan *MJSPMachine*. Hal ini memungkinkan sistem mampu mengakomodasi persoalan dengan banyak pekerjaan dan banyak mesin.

MJSPNetwork merupakan turunan dari *MJSPInputOutput*. *MJSPNetwork* mampu membaca data pekerjaan dan data mesin dalam *MJSPJob* dan *MJSPMachine* melalui operasi *GetInput* dan *SetOutput* yang disediakan oleh

MJSPInputOutput. Atribut dan operasi yang dimiliki oleh *MJSPNetwork* berkaitan dengan pemodelan MJSP[19] dengan jaringan saraf tiruan. Jadi atribut dan operasi yang disediakan oleh *MJSPNetwork* merupakan bentuk pemodelan dari perumusan *neuro-box network* (NBN) dari persoalan tersebut.

Atribut *u00* merupakan nilai rata-rata yang digunakan sebagai acuan bilangan acak untuk nilai aktivasi awal sel saraf. Jadi nilai aktivasi awal sel saraf merupakan nilai acak yang berkisar pada *u00*. Atribut *u0* merupakan nilai yang digunakan dalam fungsi sigmoid. Nilai *u0* berpengaruh pada lebar kurva. Atribut *h* digunakan untuk menyimpan nilai *time-step*. Nilai ini berpengaruh pada tingkat ketelitian. Atribut *Ck* merupakan faktor skala waktu. Atribut *A* dan *t* digunakan untuk menyimpan nilai yang diperlukan dalam persamaan differensial. Atribut *t* berpengaruh terhadap kecepatan konvergensi. Atribut *m* dan *n* digunakan untuk menyimpan nilai yang diperlukan dalam *global inhibition* sel saraf. Atribut *numiteration* digunakan untuk menyimpan jumlah iterasi yang dilakukan. Atribut *dataindek* dan *dataacak* digunakan untuk menyimpan indeks sel saraf yang aktif. Atribut *u* digunakan untuk menyimpan nilai aktivasi sel saraf. Atribut ini merupakan matriks tiga dimensi dengan indek maksimum sebesar *MAXJOB*, *MAXMACHINE*, dan *MAXTIME*. Atribut *lambda* digunakan untuk menyimpan tujuh buah nilai *Lagrange multiplier*. Atribut ini merupakan matriks satu dimensi dengan tujuh elemen. Atribut ini diperlakukan seperti tujuh buah sel saraf dengan nilai aktivasinya sama dengan nilai tujuh buah *Lagrange multiplier*. Nilai-nilai ini akan terus berubah sampai mendapatkan nilai optimalnya, yaitu pada saat sistem

[19] Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling Problem with Artificial Neural Networks", University of California, Irvine, 1991.

cenderung konvergen dan semua pembatas terpenuhi. Atribut *lambda()* digunakan untuk menyimpan nilai/solusi awal tujuh buah nilai *Lagrange multiplier*. Sehingga atribut ini juga merupakan matriks satu dimensi dengan tujuh elemen.

Operasi *vijl(i,j,l)* digunakan untuk menghitung nilai aktivasi sel saraf dengan indeks *i, j,* dan *l*. Operasi *fj(j), Tij(i,j), gijl(i,j,l), h1(i,j,l,i1), h2(i,l,i1), H(j),* dan *d(l)* merupakan fungsi-fungsi yang digunakan untuk menyelesaikan persamaan diferensial yang merupakan persamaan karakteristik sel saraf. Operasi *_1stterm(i,j,l), _2ndterm(i,j,l), _3rdterm(), _4thterm(j), _5thterm(l), 6thterm(i,j,l),* dan *_7thterm(i,j,l)* merupakan fungsi-fungsi yang mengembalikan nilai masing-masing *term* dalam persamaan karakteristik sel saraf. Operasi *f(i,j,l)* merupakan fungsi untuk menghitung nilai aktivasi sel saraf berdasarkan persamaan karakteristiknya dan operasi *lmd(i)* merupakan fungsi untuk menghitung nilai aktivasi tujuh buah sel saraf tambahan yang digunakan untuk mendapatkan nilai dari *Lagrange multiplier*. Persamaan karakteristik dari tujuh buah sel saraf tambahan ini merupakan energi masing-masing pembatas. Operasi *_1stenergy(), _2ndenergy(), _3rdenergy(), _4thenergy(), _5thenergy(), _6thenergy(),* dan *7thenergy()* merupakan fungsi untuk mendapatkan besar energi masing-masing pembatas berdasarkan fungsi energi. Operasi *ReadData(fn)* merupakan fungsi untuk membaca data yang digunakan dalam proses pembuatan jadual. Argumen yang dikirim adalah sebuah nama file dari berkas masukan. Operasi *SetInitialSolution(lmd1,lmd2,lmd3,lmd4,lmd5,lmd6,lmd7)* merupakan fungsi untuk memberikan nilai awal pada tujuh buah nilai *Lagrange multiplier*. Operasi *SetTime(k)* merupakan fungsi untuk mengatur jumlah alokasi waktu yang

disediakan. Operasi *SetIteration(i)* merupakan fungsi untuk mengatur jumlah iterasi yang akan dilakukan, sedangkan operasi *GetIteration()* merupakan fungsi untuk mendapatkan jumlah iterasi. Operasi *SetTimeStep(t)* merupakan fungsi untuk mengatur nilai *time-step*, sedangkan operasi *GetTimeStep()* merupakan fungsi untuk mendapatkan nilai *time-step*. Operasi *Initialize()* merupakan fungsi inisialisasi jaringan saraf sebelum dilakukan proses iterasi. Operasi *Iteration()* merupakan fungsi untuk iterasi pengubahan sel saraf. Operasi *GetOutputNeuron(i,j,l)* merupakan fungsi untuk mendapatkan output sel saraf, sedangkan operasi *GetActivationNeuron(i,j,l)* merupakan fungsi untuk mendapatkan nilai aktivasi sel saraf. Operasi *CheckFeasibility()* merupakan fungsi untuk mengecek konvergensi jaringan beserta feasibilitasnya. Operasi *OptimizeNetwork()* berfungsi untuk mengoptimasi jaringan *feasibel* yang sudah terbentuk. Optimasi dilakukan untuk menghilangkan waktu menunggu (*idle time*) dari masing-masing mesin. Operasi *MakeSchedule()* merupakan fungsi untuk membuat jadual melalui interpretasi terhadap kondisi akhir jaringan saraf. Operasi *WriteSchedule(fn)* merupakan fungsi untuk menulis jadual yang diperoleh dalam suatu berkas. Argumen yang dikirim merupakan nama file.

Obyek *MJSPT3DData* mempunyai atribut *i, j,* dan *l* yang digunakan untuk menyimpan indeks matriks tiga dimensi. Obyek *MJSPT3DCcak* mempunyai atribut *sumbux, sumbuy,* dan *sumbuz* yang digunakan untuk menyimpan besar indeks masing-masing sumbu *x*, *y*, dan *z*. Atribut *maxvalx, maxvaly,* dan *maxvalz* digunakan untuk menyimpan nilai maksimum pada masing-masing sumbu *x*, *y*, dan *z*. Atribut *data* adalah sebuah matriks yang terdiri atas obyek *MJSPT3DData*.

Obyek ini juga mempunyai operasi *RandNumX()*, *RandNumY()*, dan *RandNumZ()* yang berfungsi membangkitkan bilangan acak antara *0* sampai dengan *maxvalx-1*, *0* sampai dengan *maxvaly-1*, serta *0* sampai dengan *maxvalz-1*. Operasi *CheckNum(value)* digunakan untuk mengecek apakah *value* ada pada data yang disimpan. Operasi *SelRandNum()* menggunakan operasi *RandNumX()*, *RandNumY()*, *RandNumZ()* untuk membangkitkan bilangan acak dan menyimpannya pada data. Operasi *GetData(i)* berfungsi mengambil elemen ke-*i* dari data.

Berikut ini adalah implementasi struktur data dari masing-masing obyek dan desain hubungan antar obyek. Deklarasi obyek menggunakan *class* pada C++.

- ***Obyek MJSPJob***

```
class MJSPJob
{
public:
int   machine;
int   length;
int   duedate;
int   startingtime;
int   completetime;
int   finishingtime;
MJSPJob();
~MJSPJob();
};
```

- ***Obyek MJSPMachine***

```
class MJSPMachine
{
public:
int   capacity;
MJSPMachine();
~MJSPMachine();
};
```

- ***Obyek MJSPInputOutput***

```
class MJSPInputOutput
{
public:
int   numjob;
int   nummachine;
int   numtime;
MJSPInputOutput();
~MJSPInputOutput();
int   GetInput(char *fn);
int   SetOutput(char *fn);
};
```

- ***MJSPT3DData***

```
class MJSPT3DData
{
public:
int   i;
int   j;
int   l;
MJSPT3DData();
~MJSPT3DData();
};
```

♦ ***Obyek MJSPT3DAcak***

```
class MJSPT3DAcak
{
// number index of each direction x,y,z
int   sumbux, sumbuy, sumbuz;
// maximum value of each index
int   maxvalx, maxvaly, maxvalz;
// saves random 'MJSPT3DData'
MJSPT3DData *data;
// generate random value beetwen 0 to (maxvalx-1)
int   RandNumX();
// generate random value beetwen 0 to (maxvaly-1)
int   RandNumY();
// generate random value beetwen 0 to (maxvalz-1)
int   RandNumZ();
public:
MJSPT3DAcak(int,int,int,int,int,int);
~MJSPT3DAcak();
// is there 'MJSPT3DData' in 'data'
int CheckNum(T3DData);
// select a number of random 'MJSPT3Data' and save in 'data'
void SelRandNum();
// get 'MJSPT3DData' from 'data'
T3DData GetData(int);
};
```

♦ ***Obyek MJSPNetwork***

```
class MJSPNetwork:public MJSPInputOutput
{
protected:
// mean value of initial input voltages
double u00;
// this parameter effects the output voltages of the amplifier,
// increasing it gives a broader curve
double u0;
// time step
double h;
// constant for scaling
double Ck;
// constants are needed for differensial equation
double A;
double t;
// this term effects the global inhibition
double n;
double m;
// number of iteration
int numiteration;
// saves indek that will be processed asynchronously
MJSPT3DData dataindek;
MJSPT3DAcak *dataacak;
```

```
// this function calculates output voltages of the amplifier
double vijl(int i,int j,int l);
// functions are needed for differential equation
double fj(int j);
double Tij(int i,int j);
double gijl(int i,int j,int l);
double h1(int i,int j,int l,int i1);
double h2(int j,int l,int i1);
double H(double f);
double d(int l);
// terms in differential equation
double _1stterm(int i,int j, int l);
double _2ndterm(int i,int j, int l);
double _3rdterm();
double _4thterm(int j);
double _5thterm(int l);
double _6thterm(int i,int j,int l);
double _7thterm(int i,int j,int l);

// this is the differential equation for objective function
double f(int i,int j,int l);
// this is the differensial equation for additional seven neurons
double lmd(int i);

// terms in objective function
double _1stenergy();
double _2ndenergy();
double _3rdenergy();
double _4thenergy();
double _5thenergy();
double _6thenergy();
double _7thenergy();
public:

// class constructor
MJSPNetwork();
// class destructor
~MJSPNetwork();

// input voltages
double u[MAXJOB+1][MAXMACHINE+1][MAXTIME+1];
// additional seven neuron
double lambda[SEVENNEURON+1];
// this term save initial value for lagrange multiplier
double lambda0[SEVENNEURON+1];

// read data for training
int ReadData(char *fn);
// set initial solution
void SetInitialSolution(double lmd1,double lmd2,double lmd3,double
lmd4,double lmd5,double lmd6,double lmd7);
// set time allocation
void SetTime(int k);
// set number of iteration
void SetIteration(int i);
```

```
// get number of iteration
int GetIteration();
// set time step
void SetTimeStep(double t);
// get time step
double GetTimeStep();
// initialize input of the network
void Initialize();
// training network
void Iteration();
// get output neuron
double GetOutputNeuron(int i,int j,int l);
// get activation neuron
double GetActivationNeuron(int i,int j,int l);
// check feasibility
int CheckFeasibility();
// optimize network
void OptimizeNetwork();
// make schedule
void MakeSchedule();
// write schedule
int WriteSchedule(char *fn);
};
```

Dalam Gambar 4.7 obyek *MJSPInputOutput* terdiri dari obyek *MJSPJob* dan *MJSPMachine*. Implementasi dari desain tersebut ditunjukan dalam *constructor* dan *destructor class MJSPInputOutput*.

```
MJSPInputOutput::MJSPInputOutput()
{
      job          = new MJSPJob[MAXJOB+1];
      machine      = new MJSPMachine[MAXMACHINE+1];
}


MJSPInputOutput::~MJSPInputOutput()
{
      delete       job;
      delete       machine;
}
```

Obyek *MJSPT3DAcak* terdiri dari dari obyek *MJSPT3DData*. Implementasi dari desain ditunjukkan dalam *constructor* dan *destructor class MJSPT3DAcak*.

```
MJSPT3DAcak::MJSPT3DAcak()
{
      ...
      data        = new MJSPT3DData[MAXNUM];
}
MJSPT3DAcak::~MJSPT3DAcak()
{
      delete      data;
}
```

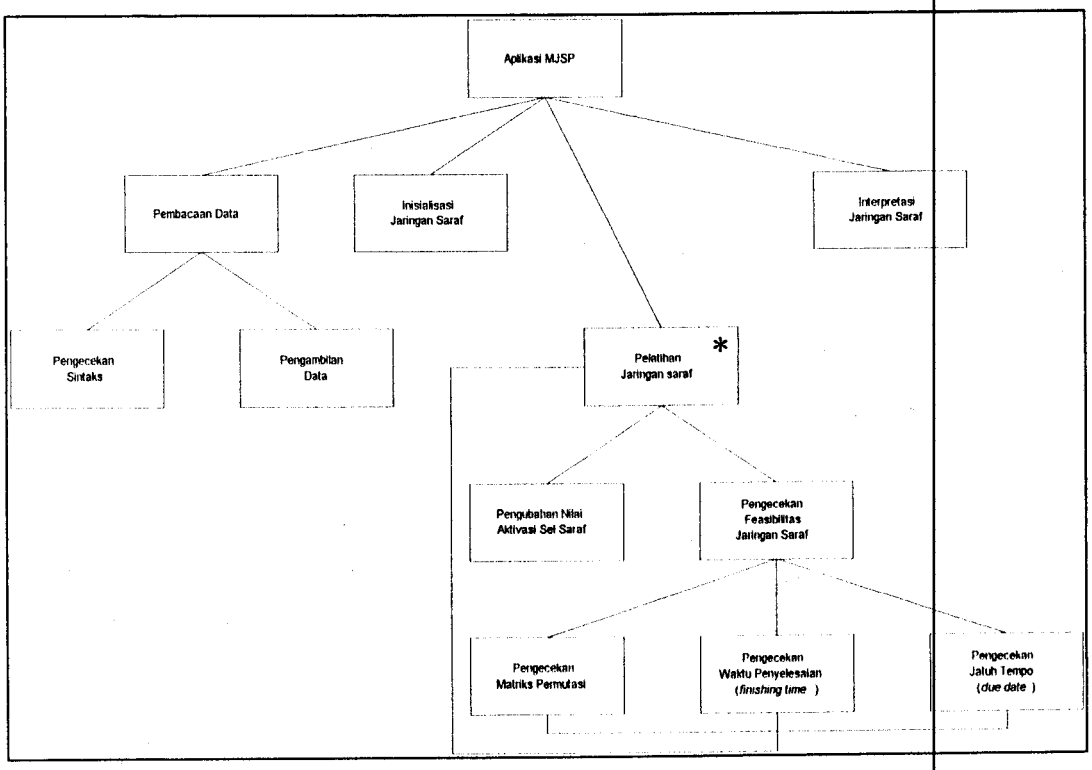
Obyek *MJSPNetwork* merupakan turunan dari *MJSPInputOutput* dan terdiri dari sebuah obyek *MJSPT3DAcak* dan sebuah obyek *MJSPT3DData*. Deklarasi penurunan obyek ditunjukkan dalam struktur data dan implementasi dari desain ditunjukkan dalam *construktor* dan *destructor class MJSPNetwork*.

```
class MJSPNetwork:public MJSPInputOutput
{
      ...
      MJSPT3DData dataindek;
      MJSPT3DAcak *dataacak;
}


MJSPNetwork::MJSPNetwork:MJSPInputOutput
{
      ...
      dataacak    = new MJSPT3DAcak;
}


MJSPNetwork::~MJSPNetwork
{
      ...
      delete      dataacak;
}
```

Obyek *MJSPT3DData* diimplementasikan secara statis sedangkan obyek *MJSPT3DAcak* diimplementasikan secara dinamis. Meskipun diimplementasikan secara dinamis, berdasarkan perancangan obyek dan desain hubungan antar obyek, *dataacak* hanya terdiri dari sebuah obyek *MJSPT3DAcak*.

4.3.4 Hirarki Proses dan Implementasinya

Bagian ini menjelaskan tentang hirarki proses dan implementasinya dari perangkat lunak yang dibuat. Hirarki proses ditunjukkan dalam Gambar 4.8. Implementasi dari masing-masing proses disajikan dalam bentuk *C++-like*. Hirarki proses dibuat berdasarkan DFD yang sudah dijelaskan sebelumnya dimana Aplikasi MJSP mempunyai empat proses utama, yaitu pembacaan data, inisialisasi jaringan saraf, pelatihan jaringan saraf, dan interpretasi jaringan saraf.



Gambar 4.8 Hirarki Proses

♦ ***Proses Pembacaan Data***

Dalam proses pembacaan data dilakukan pengecekan sintaks berkas masukan. Kemudian dilanjutkan dengan pembacaan data. Kedua proses ini ditangani oleh operasi *GetInput(fn)* dari *MJSPInputOutput*. Operasi ini dipanggil oleh operasi *ReadData(fn)* dari *MJSPNetwork*. Aplikasi menggunakan operasi *ReadData(fn)* dari *MJSPNetwork* untuk melakukan pembacaan data. Berikut ini implementasi dari proses pembacaan data.

```
void Read_Data()
{
  mjspnetwork.ReadData(filename);
}

int MJSPNetwork::ReadData(filename)
{
  return GetInput(filename);
}

int MJSPInputOutput::GetInput(char *fn)
{
  FILE *fp;
  char msg[256];
  int status,i;

  status = 0;
  /* memory allocation for file stream */
  if((fp=fopen(fn,"rt"))!=NULL)
  {
    if(fscanf(fp,"%s\n",msg))
    {
     /* check and get header file */
     if(strcmp(msg,"\"Mjsp_Application_(*.idl)\"")==0)
      {
        /* check and get number of job */
        if(fscanf(fp,"%d\n",&numjob))
        {
          /* check and get job length */
          for(i=1; i<numjob; i++)
          {
            if(!fscanf(fp,"%d",&job[i].length)) break;
          }
          if(fscanf(fp,"%d\n",&job[numjob].length))
          {
            for(i=1; i<numjob; i++)
            {
```

```
            /*check and get job duedate */
            if(!fscanf(fp,"%d",&job[i].duedate)) break;
          }
          if(fscanf(fp,"%d\n",&job[numjob].duedate))
          {
            /* check and get number of machine */
            if(fscanf(fp,"%d\n",&nummachine))
            {
              for(i=1; i<nummachine; i++)
              {
                /* check and get machine capacity */
                if(!fscanf(fp,"%d",&machine[i].capacity)) break;
              }

              if(fscanf(fp,"%d\n",&machine[nummachine].capacity))
              {
                /* check and get time allocation */
                if(fscanf(fp,"%d\n",&numtime)) status = 1;
              }
            }
          }
        }
      }
    }
  }
 }
 fclose(fp);
 return status;
}
```

- ***Proses Inisialisasi dan Pelatihan Jaringan Saraf***

Proses pelatihan menggunakan operasi *Initialize()*, *Iteration()* dan *CheckFeasibility()* dari *MJSPNetwork*. Operasi *Initialize()* dipanggil terlebih dahulu sebelum dilakukan iterasi untuk melakukan inisialisasi. Inisialisasi dilakukan terhadap semua parameter yang dibutuhkan untuk pelatihan jaringan saraf dan *Lagrange multiplier*. Operasi *Iteration()* akan dieksekusi sebanyak jumlah iterasi yang diinginkan. Kemudian dilakukan pengecekan konvergensi dan feasibilitas dari jaringan saraf. Langkah-langkah ini akan diulangi terus sampai kondisi *feasibel* tercapai atau jaringan cenderung divergen atau tidak bisa memberikan suatu pemecahan. Jika hal ini terjadi maka harus dilakukan pengubahan terhadap data dari berkas masukan. Kompromi yang bisa dipilih

adalah penambahan jumlah mesin atau perpanjangan jatuh tempo dengan konsekuensi penambahan alokasi waktu. Operasi *Iteration()* melakukan pengubahan nilai aktivasi sel saraf. Jumlah sel saraf yang terlibat sama dengan jumlah pekerjaan dikalikan dengan jumlah mesin dikalikan dengan alokasi waktu. Dalam operasi *CheckFeasibility()* nilai *status* sama dengan satu jika *CheckMatrixPermutation()*, *CheckFinishingTime()*, dan *CheckDueDate()* bernilai benar. Aplikasi memanggil *Start_Simulation()* untuk memulai proses pelatihan. Berikut ini implementasi dari proses inisialisasi dan pelatihan jaringan saraf.

```
void Start_Simulation()
{
  iteration = mjspnetwork.GetIteration();
  do
  {
    mjspnetwork.Initialize();
    for(indeks=1; indeks <= iteration ; indeks++)
    {
      mjspnetwork.Iteration();
    }
  }while(!mjspnetwork.CheckFeasibility() && !DIVERGEN);
}

void MJSPNetwork::Initialize()
{
  SetAllParameters();
  SetLagrangeMultipliers();
}

void MJSPNetwork::Iteration()
{
  numcell = numjob*nummachine*numtime;
  for(indeks=1; indeks <= numcell; indeks++)
  {
    dataindek = dataacak.GetData(indeks);
    u[dataindek.i][dataindek.j][dataindek.l] +=
    GetTimeStep()*f([dataindek.i][dataindek.j][dataindek.l]);
  }
}
```

```
int MJSPNetwork::CheckFeasibility()
{
  status = 0;
  if(CheckNumOfActiveNeuron() != numjob) return status;
  if(CheckMatrixPermutation())
    if(CheckFinishingTime())
      if(CheckDueDate())
        status = 1;
  return status;
}
```

Proses pengecekan matriks permutation dilakukan dengan menghitung jumlah sel saraf yang aktif pada sumbu *x*, sumbu *y*, dan sumbu *z* yang mewakili waktu, mesin, dan pekerjaan. Jumlah sel saraf yang aktif pada masing-masing arah tersebut harus sama dengan nol atau satu. Dengan catatan pada setiap arah waktu sama dengan satu, jumlah sel saraf yang aktif harus sama dengan satu dan tidak boleh nol. Hal ini untuk memastikan bahwa pada alokasi waktu ke-1 setiap mesin harus memroses satu pekerjaan. Dalam proses pengecekan feasibilitas, sebelum dilakukan proses pengecekan matriks permutasi harus dipastikan dahulu jumlah sel saraf yang aktif sama dengan jumlah pekerjaan. Jika tidak sama maka sistem tidak *feasibel* dan proses pengecekan feasibilitas tersebut mengembalikan nilai salah.

```
int MJSPNetwork::CheckMatrixPermutation()
{
  status = 0;
  if(CheckNumOfActiveNeuronAtJobDirection())
    if(CheckNumOfActiveNeuronAtMachineDirection())
      if(CheckNumOfActiveNeuronAtTimeDirection())
        status = 1;
  return status;
}
```

Proses pengecekan waktu penyelesaian ini dilakukan untuk memastikan bahwa dalam satu mesin, jika suatu pekerjaan sedang diproses maka tidak boleh ada pekerjaan baru yang dialokasikan pada mesin tersebut sebelum pekerjaan

yang pertama kali dialokasikan selesai diproses. Dalam kode tersebut bisa dilihat, jika waktu mulai pekerjaan baru kurang dari waktu penyelesaian pekerjaan sebelumnya maka operasi tersebut mengembalikan nilai salah. Waktu penyelesaian sama dengan waktu mulai proses ditambah dengan panjang proses pekerjaan dikurangi satu.

```
int MJSPNetwork::CheckFinishingTime()
{
  status = 1;
  for(j=1; j<=nummachine; j++)
  {
    finishingtime = 0;
    for(l=1; l<=numtime; l++)
    {
      for(i=1; i<=numjob; i++)
      {
        /*check active neuron*/
        if(vijl(i,j,l) > 0.5)
        {
          /*check starting time*/
          if(l < finishingtime)
          {
            status = 0;
            break;
          }
          finishingtime = l + job[i].length - 1;
        }
      }
      if(status==0) break;
    }
    if(status==0) break;
  }
  return status;
}
```

Proses pengecekan jatuh tempo dilakukan dengan menghitung waktu penyelesaian pekerjaan. Jika waktu penyelesaian pekerjaan lebih dari jatuh tempo pekerjaan tersebut, maka jatuh tempo telah dilanggar. Sehingga operasi tersebut mengembalikan nilai salah.

```
int MJSPNetwork::CheckDueDate()
{
  status = 1;
  for(j=1; j<=nummachine; j++)
  {
    for(l=1; l<=numtime; l++)
    {
      for(i=1; i<=numjob; i++)
      {
        /*check active neuron*/
        if(vijl(i,j,l) > 0.5)
        {
          finishingtime = l + job[i].length - 1;

          /*check duedate*/
          if(finishingtime > job[i].duedate)
          {
            status = 0;
            break;
          }
        }
      }
      if(status==0) break;
    }
    if(status==0) break;
  }
  return status;
}
```

- ***Proses Interpretasi Jaringan saraf***

Proses interpretasi jaringan saraf menggunakan operasi *OptimizeNetwork()* dan *MakeSchedule()* dari *MJSPNetwork*. Proses ini melakukan pencarian sel saraf yang aktif (*fired*). Untuk mengetahui apakah suatu sel saraf pada indeks *i, j*, dan *l* aktif, operasi ini memanggil operasi *vijl(i,j,l)*. Jika operasi ini mengembalikan nilai lebih dari 0.5 maka sel saraf pada indeks *i, j*, dan *l* aktif. Nilai 0.5 dipilih untuk menjamin validitas dari solusi. Berikut ini implementasi proses interpretasi jaringan saraf.

```
void Schedule_Result()
{
  mjspnetwork.OptimizeNetwork();
  mjspnetwork.MakeSchedule();
}


void MJSPNetwork::OptimizeNetwork()
{
  for(j=1; j<=nummachine; j++)
  {
    for(i=1; i<=numjob; i++)
    {
      if(vijl(i,j,1) > 0.5)
      {
        if(fmod(job[i].length,machine[j].capacity)==0)
        {
          ljobp.completetime = job[i].length /
          machine[j].capacity;
        }
        else
        {
          ljobp.completetime = job[i].length / machine[j].capacity
          + 1;
        }
        ljobp.finishingtime = ljobp.completetime;
      }
    }
```

```
for(l=2; l<=numtime; l++)
{
  for(i=1; i<=numjob; i++)
  {
    if(vijl(i,j,l) > 0.5)
    {
      if(l == (ljobp.finishingtime + 1))
      {
        ljobp.startingtime = l;

        if(fmod(job[i].length,machine[j].capacity)==0)
        {
          ljobp.completetime = job[i].length /
          machine[j].capacity;
        }
        else
        {
          ljobp.completetime = job[i].length /
          machine[j].capacity + 1;
        }
        ljobp.finishingtime = ljobp.startingtime +
        ljobp.completetime - 1;
      }
      else
      {
        activation = u[i][j][ljobp.finishingtime + 1];
        u[i][j][ljobp.finishingtime + 1] = u[i][j][l];
        u[i][j][l] = activation;
        ljobp.startingtime = ljobp.finishingtime + 1;

        if(fmod(job[i].length,machine[j].capacity)==0)
        {
          ljobp.completetime = job[i].length /
          machine[j].capacity;
        }
        else
        {
          ljobp.completetime = job[i].length /
          machine[j].capacity + 1;
        }
        ljobp.finishingtime = ljobp.startingtime +
        ljobp.completetime - 1;
      }
    }
  }
}
}
```

```
void MJSPNetwork::MakeSchedule()
{
  for(i=1; i<=numjob; i++)
  {
    for(j=1; j<=nummachine; j++)
    {
      for(l=1; l<=numtime; l++)
      {
        if(vijl(i,j,l) > 0.5)
        {
          job[i].machine = j;
          job[i].startingtime = l;

          if(fmod(job[i].length,machine[j].capacity)==0)
          {
            job[i].completetime = job[i].length /
            machine[j].capacity;
          }
          else
          {
            job[i].completetime = job[i].length /
            machine[j].capacity + 1;
          }
          job[i].finishingtime = l + job[i].completetime - 1;
        }
      }
    }
  }
}
```

BAB V

UJI COBA DAN EVALUASI

PERANGKAT LUNAK

Bab ini menjelaskan hasil uji coba perangkat lunak dan evaluasi terhadap hasil uji coba tersebut berkenaan dengan studi literatur yang telah dilakukan. Uji coba dilakukan terhadap suatu persoalan dengan beberapa kasus yang berbeda. Kemudian dilakukan evaluasi terhadap pemecahan yang dihasilkan dari uji coba tersebut.

5.2 Data Uji Coba

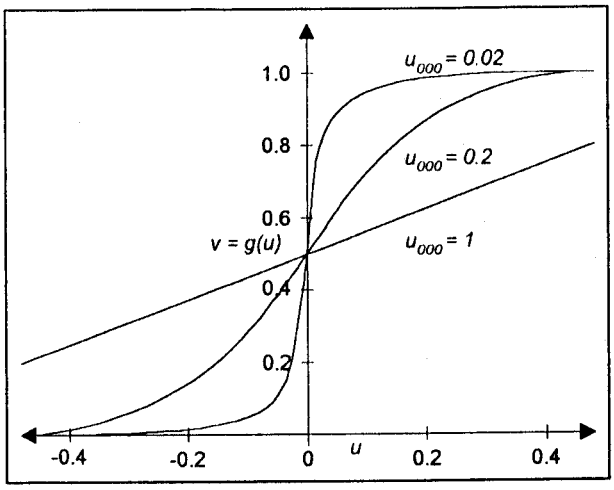
Persoalan yang diujikan terhadap perangkat lunak ditunjukkan dalam Tabel 5.1. Persoalan tersebut terdiri dari 6 pekerjaan. Berturut-turut persoalan tersebut diselesaikan dengan 3 mesin dan 6 alokasi waktu dengan kapasitas masing-masing mesin sama dengan 1. Kemudian persoalan tersebut diselesaikan dengan 3 mesin dan 4 alokasi waktu dengan kapasitas masing-masing mesin sama dengan 2. Terakhir, persoalan tersebut diselesaikan dengan 2 mesin dan 4 alokasi waktu dengan kapasitas masing-masing mesin sama dengan 2. Pengertian alokasi waktu dalam hal ini adalah waktu yang dialokasikan terhadap sistem yang memaksa setiap pekerjaan harus sudah ditangani oleh sebuah mesin dalam alokasi waktu tersebut.

Tabel 5.1 Contoh Persoalan dengan 6 Pekerjaan

Pekerjaan	**1**	**2**	**3**	**4**	**5**	**6**
Panjang Proses	2	3	1	3	3	1
Jatuh Tempo	3	6	5	5	6	2

5.2 Uji Coba dan Evaluasi Hasil

Dalam uji coba ini persamaan diferensial *nonlinier* dari persamaan karakteristik sel saraf diselesaikan dengan menggunakan metode Euler orde satu. Nilai awal aktivasi sel saraf $u_{ijl}(0)$ dan *Lagrange multiplier* $\lambda_{\alpha}(0)$ dipilih dalam kisaran yang telah diusulkan oleh Hopfield[1] dan Wacholder[2], yaitu $u_{ijl} = u_{000} + \delta u_{ijl}$ dan $-0,1u_{000} \leq \delta u_{ijl} \leq 0,1u_{000}$ dimana $u_{000} = [0,02;0,05]$ dan $\lambda_{\alpha}(0) > 0$. Gambar 5.1 menunjukkan fungsi sigmoid yang dihasilkan.

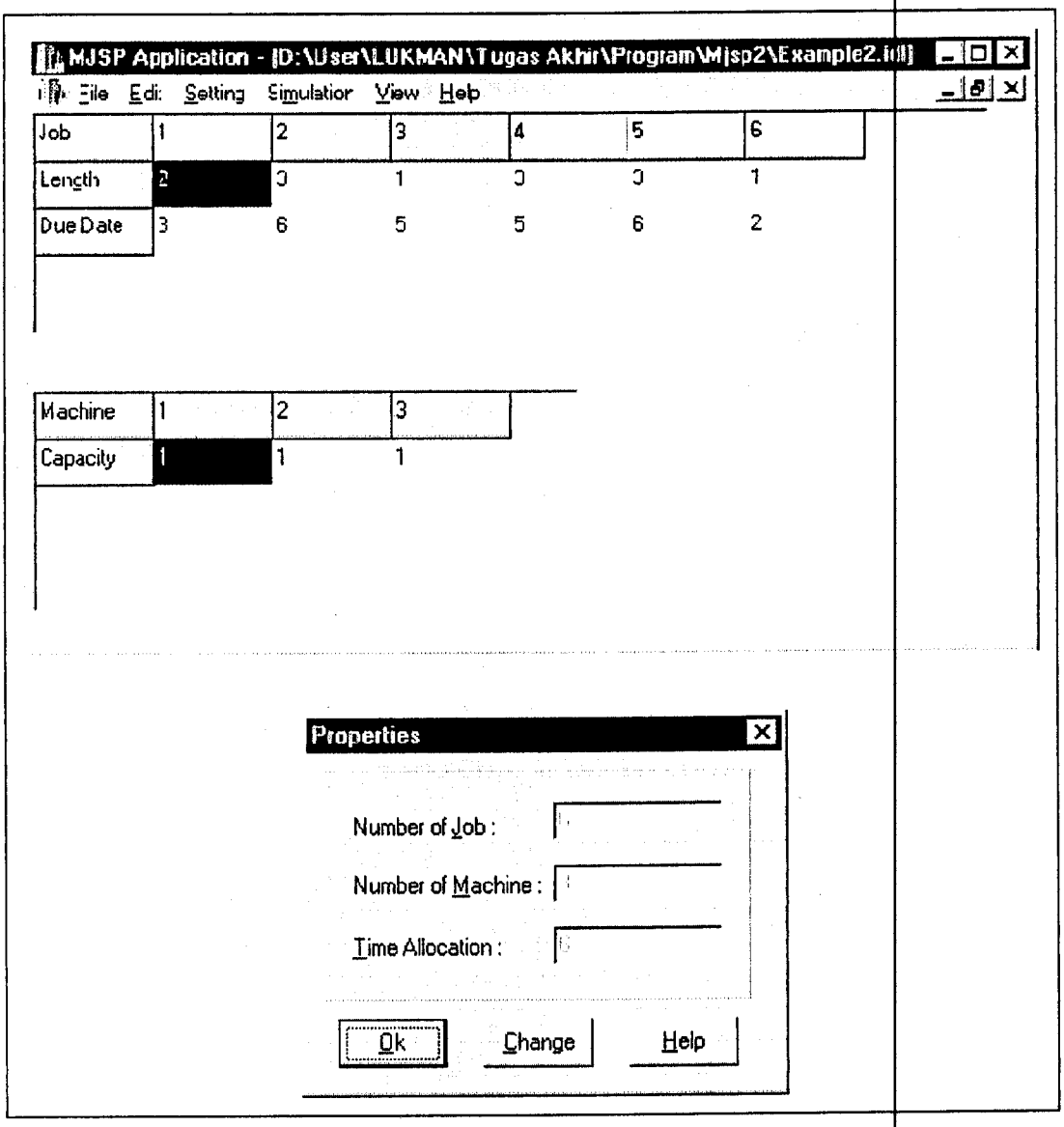


Gambar 5.1 Aproksimasi Kurva Sigmoid dengan Beberapa Nilai u_{000}

[1] J.J. Hopfield dan D.W. Tank, "Neural Computation of Decision in Optimization Problem", Biol. Cybernet. 52, 141-152, 1985.
[2] E. Wacholder, J. Han, dan R.C. Momn, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem", Biol. Cybernet. 61, 11-19, 1989.

Perangkat lunak ini menggunakan nilai 0,02 untuk u_{000} dan $\lambda_\alpha(0) > 0$. Seperti dijelaskan dalam bab sebelumnya, λ_α adalah *Lagrange multiplier*. Nilai standard yang dipilih untuk λ_α adalah 1. Dengan demikian sistem menganggap λ_α tidak berpengaruh pada awal iterasi. Kemudian nilai ini akan berubah mencari nilai optimalnya. Seperti dijelaskan juga dalam bab sebelumnya angka perubahan untuk λ_α sama dengan E_α, yaitu energi untuk masing-masing pembatas (fungsi kendala). Hasil uji coba menunjukkan bahwa berapapun nilai $\lambda_\alpha(0)$ asal lebih dari 0 sistem bisa mendapatkan hasil yang optimal. Perbedaannya terletak pada waktu komputasi untuk mendapatkan hasil optimal tersebut. Sekumpulan nilai $\lambda_\alpha(0)$ mungkin mendapatkan hasil optimal cukup dengan satu kali pelatihan saja sedangkan sekumpulan nilai yang lain memerlukan beberapa kali pelatihan. Bahkan untuk persoalan yang sama sekumpulan nilai $\lambda_\alpha(0)$ bisa mendapatkan hasil yang berbeda dengan waktu komputasi yang berbeda pula bila dilakukan pelatihan kembali. Jadi waktu komputasi untuk mendapatkan suatu pemecahan yang valid sangat sensitif terhadap pemilihan nilai $\lambda_\alpha(0)$.

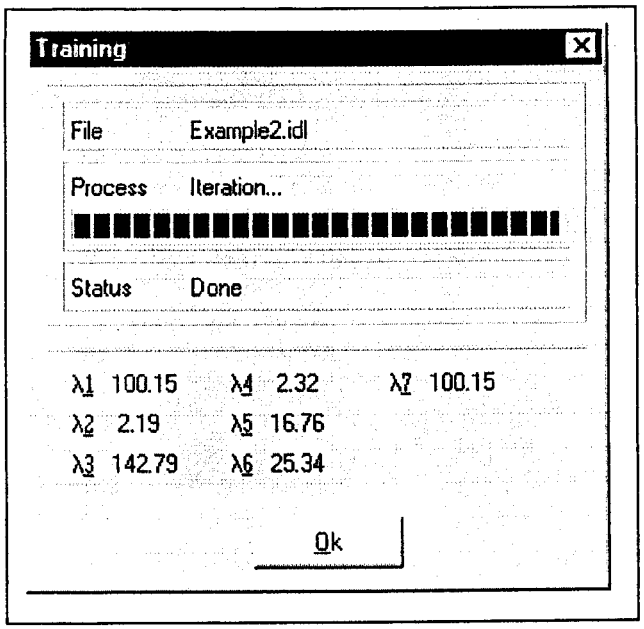
Gambar 5.2 merupakan bentuk tampilan data uji coba yang dikeluarkan perangkat lunak jika persoalan tersebut diujikan pada 3 mesin dengan 6 alokasi waktu. Data uji coba tersebut berupa data pekerjaan, yaitu jumlah pekerjaan, panjang proses, dan jatuh tempo masing-masing pekerjaan. Kemudian data mesin, yaitu jumlah mesin dan kapasitas masing-masing mesin serta data waktu berupa jumlah waktu yang dialokasikan.



Gambar 5.2 Data Uji Coba untuk
Persoalan 6 Pekerjaan dan 3 Mesin dengan 6 Alokasi Waktu

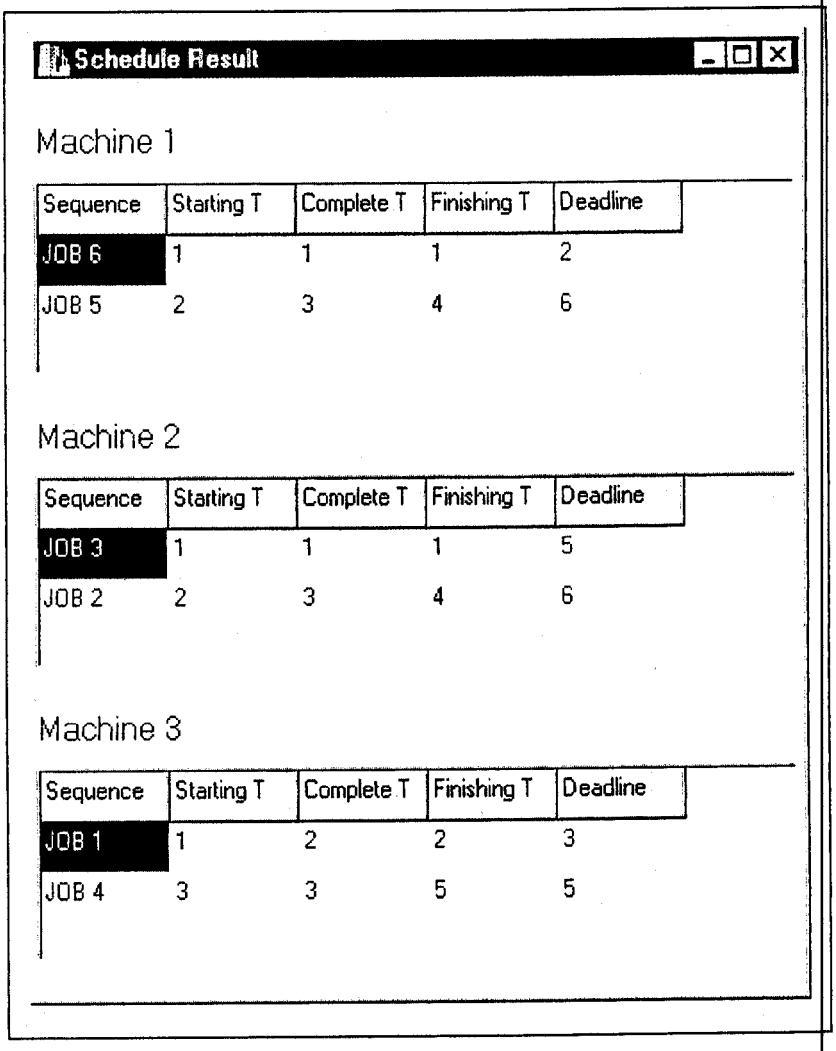
Berikut ini contoh hasil yang dikeluarkan oleh perangkat lunak jika persoalan tersebut diujikan. Gambar 5.3 menampilkan status pelatihan beserta nilai akhir dari masing-masing *Lagrange multiplier*. Gambar 5.4 menampilkan jadual yang dihasilkan oleh perangkat lunak. Jadual ditampilkan dalam bentuk urut-urutan pekerjaan pada masing-masing mesin. Gambar 5.5 menampilkan

penugasan pada masing-masing mesin. Hasil penugasan ini identik dengan output sel saraf pada akhir pelatihan. Gambar 5.6 menampilkan nilai aktivasi sel saraf pada akhir pelatihan.



Gambar 5.3 Nilai Akhir untuk Masing-masing *Lagrange multiplier*

Hasil yang ditunjukkan dalam gambar-gambar tersebut diperoleh dengan memberikan nilai awal, $\lambda_\alpha(0) = 1$, untuk setiap *Lagrange multiplier*. Nilai akhir ini ditentukan oleh nilai aktivasi awal sel saraf, $u_{ij}(0)$. Seperti dijelaskan sebelumnya, nilai $u_{ij}(0)$ diperoleh secara acak berdasarkan suatu nilai rata-rata. Sehingga nilai acak berada dalam kisaran nilai rata-rata tersebut. Dengan demikian hasil yang diperoleh tidak mesti sama jika pada persoalan tersebut dilakukan pelatihan kembali.



Gambar 5.4 Jadual untuk
Persoalan 6 Pekerjaan dan 3 Mesin dengan 6 Alokasi Waktu

Jadual yang dihasilkan dalam Gambar 5.4 dijelaskan berikut ini. Informasi yang ditampilkan berupa *Sequence*, *Starting T*, *Complete T*, *Finishing T*, dan *Deadline*. *Sequence* menampilkan informasi urut-urutan pekerjaan dalam mesin. *Starting T* menampilkan informasi alokasi waktu dari suatu pekerjaan pada saat pekerjaan tersebut mulai diproses. *Complete T* menampilkan informasi jumlah alokasi waktu yang diperlukan mesin untuk menyelesaikan pekerjaan. *Finishing T* menampilkan informasi alokasi waktu dari suatu pekerjaan pada saat pekerjaan

tersebut diselesaikan. *Deadline* menampilkan informasi waktu jatuh tempo dari pekerjaan yang ditangani. Dalam hal ini perlu diperhatikan penggunaan istilah 'alokasi waktu' dan 'waktu'. Alokasi waktu ke-t adalah waktu antara (t-1) dan t. Jadi alokasi waktu ke-1 adalah waktu antara 0 dan 1, alokasi waktu ke-2 adalah waktu antara 1 dan 2, dan seterusnya.

Dalam Gambar 5.4 bisa dilihat jadual yang dihasilkan oleh mesin 1. Urutan pekerjaan yang diproses adalah pekerjaan 6 dan pekerjaan 5. Pekerjaan 6 mempunyai *Starting T* sama dengan 1, *Complete T* sama dengan 1, *Finishing T* sama dengan 1, dan *Deadline* sama dengan 2. Artinya, pekerjaan 6 diproses pada alokasi waktu ke-1, jumlah alokasi waktu yang dibutuhkan mesin 1 untuk menyelesaikan pekerjaan 6 sama dengan 1, sehingga pada alokasi waktu ke-1 pekerjaan 6 selesai diproses, dan memenuhi waktu jatuh tempo, yaitu 2. Seperti dijelaskan dalam bab sebelumnya, jumlah alokasi waktu yang dibutuhkan suatu mesin j untuk menyelesaikan pekerjaan i sama dengan panjang proses pekerjaan i dibagi dengan kapasitas mesin j dan hasilnya dibulatkan ke atas. Karena pekerjaan 6 diselesaikan pada alokasi waktu ke-1 maka pada alokasi waktu ke-2 mesin 1 memroses pekerjaan 5. Hal ini bisa dilihat dari informasi yang diperoleh untuk pekerjaan 5, yaitu *Starting T* sama dengan 2, *Complete T* sama dengan 3, *Finishing T* sama dengan 4, dan *Deadline* sama dengan 6. Artinya, pekerjaan 5 mulai diproses pada alokasi waktu ke-2, jumlah alokasi waktu yang dibutuhkan mesin 1 untuk menyelesaikan pekerjaan 5 sama dengan 3, sehingga alokasi waktu ke-2, alokasi waktu ke-3, dan alokasi waktu ke-4 digunakan mesin 1 untuk menyelesaikan pekerjaan 5. Dengan demikian pada alokasi waktu ke-4, pekerjaan

5 diselesaikan oleh mesin 1 sehingga hasil ini memenuhi jatuh tempo untuk pekerjaan 5, yaitu 6.

Dari beberapa uji coba yang dilakukan pada perangkat lunak, pada prinsipnya perangkat lunak mampu menyelesaikan persoalan yang diberikan. Keterbatasan perangkat lunak terletak pada kemampuan komputer dimana perangkat lunak ini diujikan. Waktu komputasi ditentukan oleh berapa banyak sel saraf yang terlibat dan seberapa 'baik' nilai $\lambda(0)$. Konvergensi ditentukan oleh konstanta waktu τ pada persamaan karakteristik sel saraf yang berupa persamaan differensial.

Hasil penugasan n pekerjaan pada m mesin seperti ditunjukkan dalam Gambar 5.5 identik dengan output masing-masing sel saraf. Dalam perangkat lunak ini untuk menjamin validitas solusi, dalam pengecekan feasibilitas, sel saraf dikatakan aktif jika nilai outputnya lebih dari 0,5 dan sebaliknya jika kurang dari atau sama dengan 0,5 sel saraf dikatakan tidak aktif. Nilai ini muncul karena fungsi yang digunakan bersifat kontinyu sehingga nilainya tidak diskrit, 0 atau 1, tetapi berada dalam kisaran antara 0 dan 1. Dengan demikian hasil yang diperoleh lebih halus. Nilai output tersebut dihitung berdasarkan nilai aktivasi sel saraf. Nilai aktivasi sel saraf yang dihasilkan dalam simulasi persoalan ini ditunjukkan dalam Gambar 5.6.

Machine 1

	1	2	3	4	5	6
JOB 1	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 3	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 4	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 5	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000
JOB 6	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Machine 2

	1	2	3	4	5	6
JOB 1	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 2	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000
JOB 3	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 4	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 6	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Machine 3

	1	2	3	4	5	6
JOB 1	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 3	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 4	0.00000	0.00000	0.96475	0.00000	0.00000	0.00000
JOB 5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
JOB 6	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Gambar 5.5 Hasil Penugasan untuk Masing-masing Mesin

Machine 1

	1	2	3	4	5	6
JOB 1	-297.18008	-143.79157	-151.20839	-80.84383	-91.37061	-150.43836
JOB 2	-436.83257	-305.59641	-266.02513	-47.91156	-25.28588	-51.13745
JOB 3	-169.92840	-47.75027	-143.86696	-51.27705	-16.75443	-25.32523
JOB 4	-422.05526	-290.82443	-190.43739	-41.55948	-36.61031	-78.88226
JOB 5	-338.39763	15.87108	-45.43048	-46.07091	-55.53099	-79.83204
JOB 6	14.56586	-91.17238	-184.45418	-114.09673	-125.21043	-182.88900

Machine 2

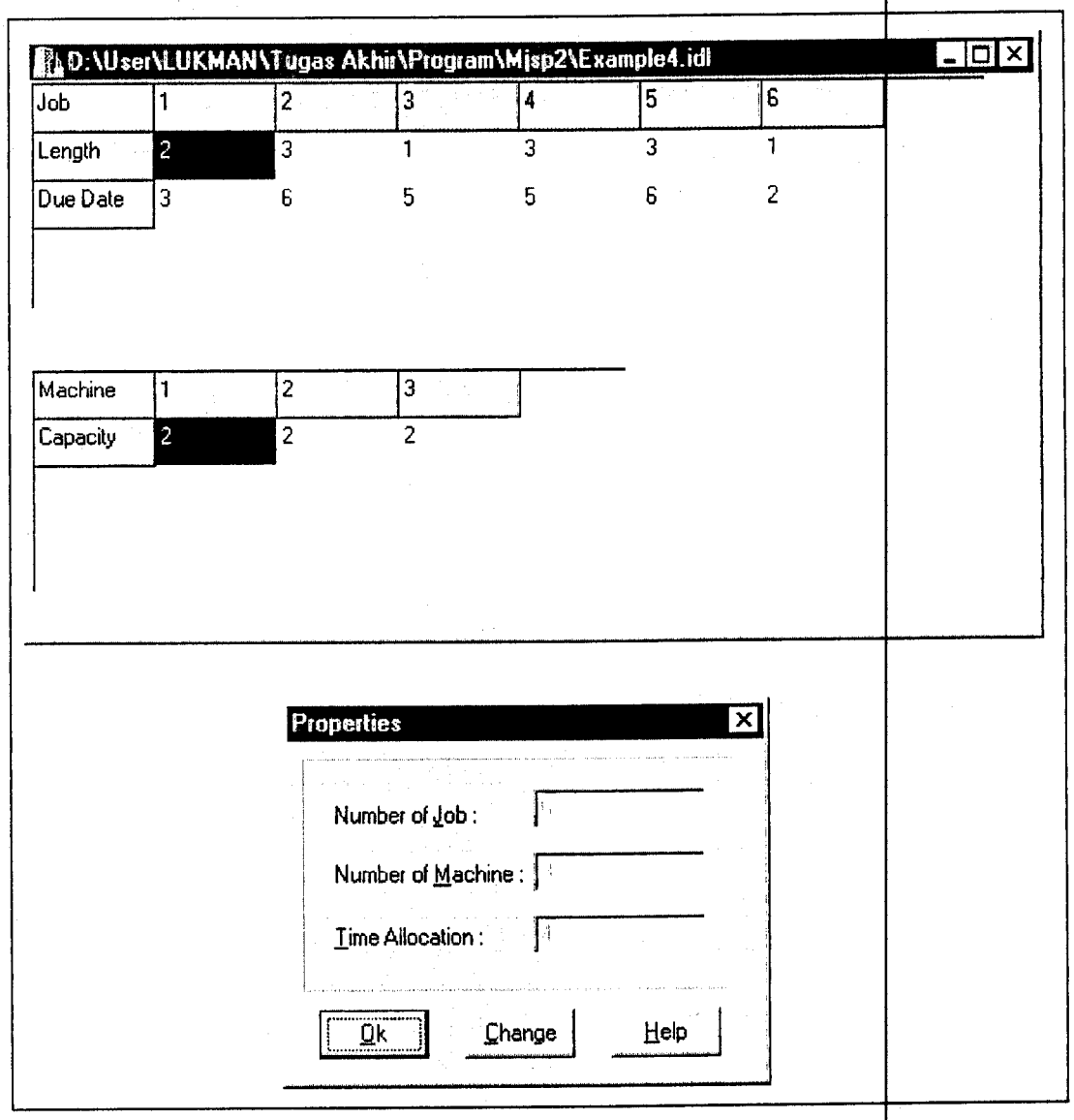
	1	2	3	4	5	6
JOB 1	-273.84832	-68.66098	-153.32621	-178.04006	-123.30116	-150.76351
JOB 2	-359.98645	15.27732	-85.09957	-46.29403	-55.07430	-80.14955
JOB 3	14.71626	-54.15485	-79.12583	-179.58223	-79.78514	-57.10389
JOB 4	-346.35912	-171.12674	-354.33248	-139.32892	-69.36175	-78.08230
JOB 5	-361.33294	-307.38862	-307.63207	-145.85354	-57.45586	-53.07001
JOB 6	-174.31847	-15.49397	-61.27642	-183.32907	-127.78212	-156.41870

Machine 3

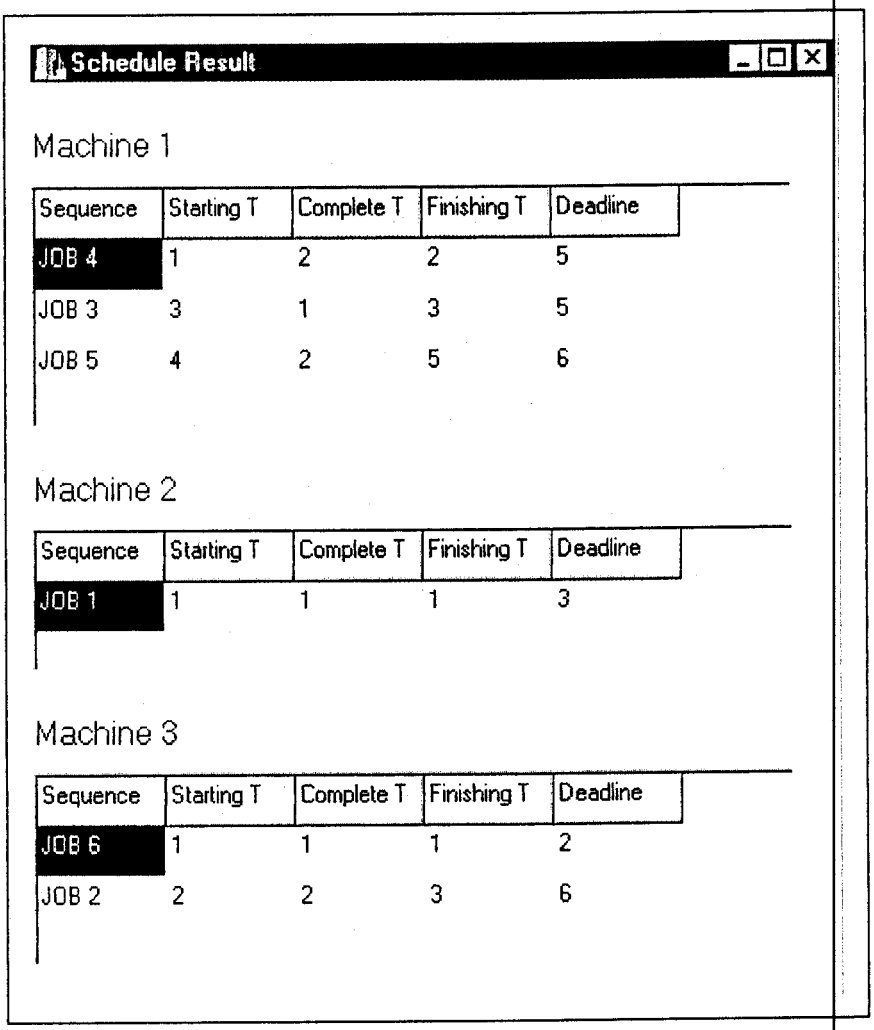
	1	2	3	4	5	6
JOB 1	11.70244	-73.87296	-137.81411	-154.37243	-145.75932	-179.15073
JOB 2	-332.21005	-135.61805	-314.20943	-104.80461	-55.58747	-49.62574
JOB 3	-171.10988	-50.47183	-39.62606	-71.69541	-38.92874	-25.93605
JOB 4	-309.94045	-60.33823	0.03309	-48.23465	-49.50569	-93.27822
JOB 5	-332.29787	-257.03638	-191.60781	-104.38482	-57.09416	-49.74780
JOB 6	-172.77935	-63.03921	-54.17760	-117.76790	-115.52759	-152.54321

Gambar 5.6 Nilai Aktivasi Sel Saraf pada Akhir Pelatihan

Gambar 5.7 menampilkan persoalan yang sama, 6 pekerjaan dan 3 mesin tetapi dengan 4 alokasi waktu dan kapasitas masing-masing mesin dinaikkan 2 kali lipat. Panjang proses dan jatuh tempo masing-masing pekerjaan juga sama. Gambar 5.8 menampilkan pemecahan dari persoalan tersebut. Bisa dilihat dalam gambar tersebut, setiap pekerjaan sudah ditangani oleh sebuah mesin dalam waktu kurang dari sama dengan 4. Hasil ini sesuai dengan alokasi waktu yang disediakan. Pemecahan persoalan ini bisa dibandingkan dengan pemecahan pada persoalan sebelumnya.

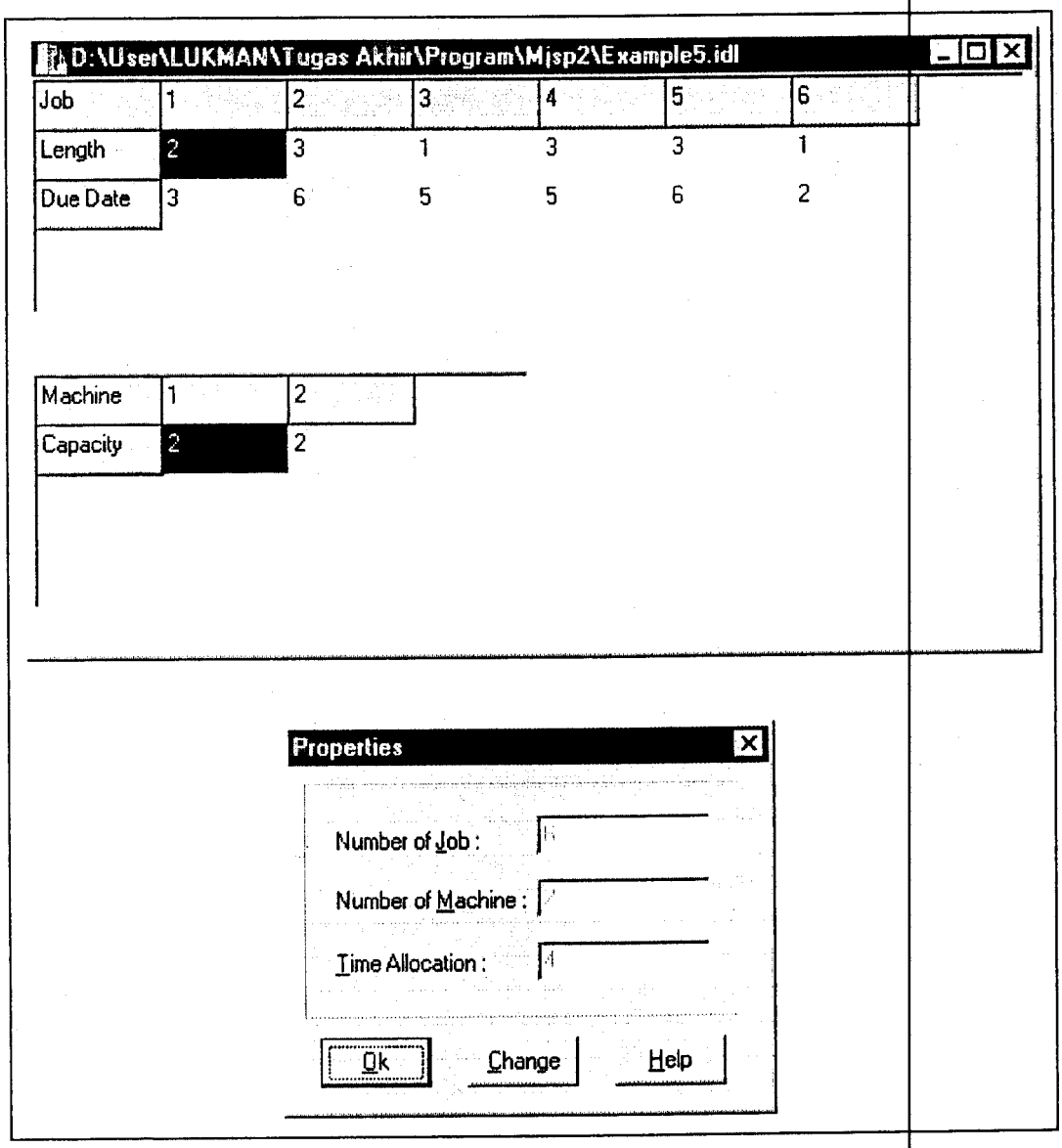


Gambar 5.7 Data Uji Coba untuk
Persoalan 6 Pekerjaan dan 3 Mesin dengan 4 Alokasi Waktu

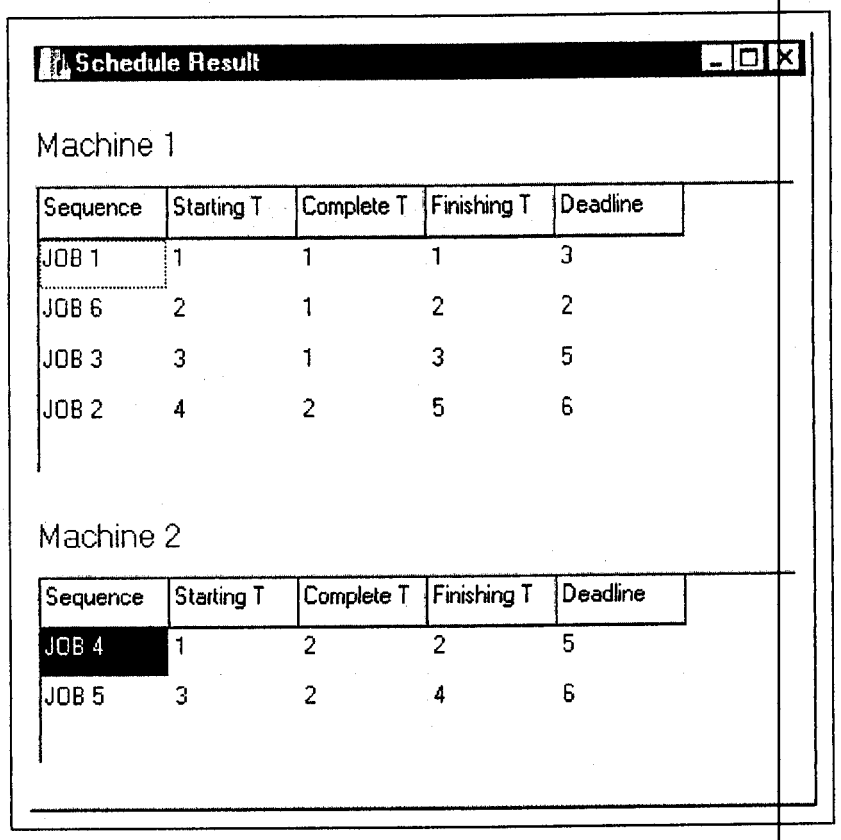


Gambar 5.8 Jadual untuk
Persoalan 6 Pekerjaan dan 3 Mesin dengan 4 Alokasi Waktu

Gambar 5.9 menampilkan data uji coba untuk persoalan yang sama, panjang proses dan jatuh tempo masing-masing pekerjaan tetap, tetapi jumlah mesin dikurangi menjadi 2 dengan 4 alokasi waktu. Pemecahan persoalan ini ditunjukkan dalam Gambar 5.10. Pemecahan yang diperoleh ini bisa dibandingkan dengan pemecahan dua persoalan sebelumnya.



Gambar 5.9 Data Uji Coba untuk
Persoalan 6 Pekerjaan dan 2 Mesin dengan 4 Alokasi Waktu



Gambar 5.10 Jadual untuk
Persoalan 6 Pekerjaan dan 2 Mesin dengan 4 Alokasi Waktu

BAB VI

KESIMPULAN DAN SARAN

Bab ini menjelaskan kesimpulan dan saran dari Tugas Akhir Perancangan dan Pembuatan Perangkat Lunak Penjadualan Banyak Pekerjaan dengan Jaringan saraf Tiruan. Kesimpulan dan saran diambil berdasarkan studi literatur, perancangan, dan hasil uji coba perangkat lunak yang telah diuraikan dalam bab-bab sebelumnya.

6.1 Kesimpulan

Berikut ini dijelaskan beberapa kesimpulan yang dapat diambil dari Tugas Akhir. Studi literatur, perancangan, dan hasi uji coba beserta evaluasinya berperan penting dalam perumusan kesimpulan ini.

Pertama, persoalan penjadualan banyak pekerjaan, yang merupakan persoalan NP-*Complete*, dapat diselesaikan dengan pendekatan jaringan saraf tiruan yang dirumuskan oleh Zhen-Ping Lo dan Behnam Bavarian[22]. Pendekatan ini menggunakan jaringan Hopfield yang didesain khusus untuk menyelesaikan persoalan penjadualan. Persoalan penjadualan dirumuskan dalam bentuk penugasan *n* buah pekerjaan pada *m* buah mesin menggunakan jaringan saraf tiga dimensi dengan model Hopfield-*like*, yaitu *Neuro-Box Network* (NBN).

[22] Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling with Artificial Neural Network", Univerisity of California, Irvine, 1991.

Kedua, pendekatan NBN dapat menggabungkan beberapa mesin dengan kapasitas yang berbeda dan beberapa pekerjaan dengan panjang proses dan jatuh tempo yang berbeda pula. Pendekatan NBN mengijinkan ekspansi ke ketiga arah dimensi, dengan kata lain NBN mampu mengadopsi persoalan dengan sejumlah pekerjaan, sejumlah mesin, dan sejumlah waktu alokasi.

Ketiga, konvergensi menuju suatu pemecahan yang *feasible* dapat dicapai tergantung pada persoalan yang ditangani. Hasil uji coba menunjukkan bahwa model ini tidak bisa mendapatkan pemecahan yang *feasible* untuk persoalan dengan pembatas (*constraint*) yang ketat. Hasil pengamatan menunjukkan ternyata benar bahwa konvergensi ditentukan oleh aproksimasi terhadap konstanta waktu τ dari persamaan differensial yang merupakan persamaan karakteristik sel saraf dan tidak ditentukan oleh dimensi NBN. Dimensi NBN hanya berpengaruh terhadap waktu komputasi.

Keempat, nilai awal untuk *Lagrange multiplier*, $\lambda_d(0)$, sangat berpengaruh pada konvergensi menuju solusi yang *feasibel*. Pemilihan nilai awal yang 'baik' akan mempercepat waktu komputasi karena mengurangi jumlah pelatihan terhadap jaringan. Sejumlah nilai awal mungkin baik untuk beberapa persoalan, tetapi tidak untuk beberapa persoalan yang lain. Cara yang paling mudah untuk mendapatkannya hanyalah dari eksperimen.

6.2 Saran

Saran yang bisa diajukan untuk Tugas Akhir ini dapat diuraikan seperti berikut. Saran yang diajukan berkaitan dengan perumusan konstruksi jaringan saraf tiruan yang digunakan dan untuk pengembangan perangkat lunak.

Pertama, perlu dilakukan studi lebih lanjut untuk mendapatkan nilai awal, $\lambda_\alpha(0)$, yang 'baik' untuk *Lagrange multiplier*. Sehingga nilai awal yang 'baik' tersebut tidak lagi diperoleh dari serangkaian eksperimen, melainkan melalui suatu perumusan yang mengurangi faktor ketidakpastian.

Kedua, pengembangan perangkat lunak dapat dilakukan dengan memperbaiki metode komputasi yang digunakan. Sehingga dapat memanfaatkan sumber daya komputer secara optimal. Dengan demikian kinerja perangkat lunak dapat ditingkatkan.

DAFTAR PUSTAKA

1. James A. Freeman, David M. Skapura, "Neural Networks Algorithms, Application, and Programming Techniques", Addison-Wesley Publishing Company, 1991.
2. James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, "Object-Oriented Modelling and Design", Prentice Hall Englewood Cliffs, New Jersey 07632, 1991.
3. Michael Pinedo, "Scheduling: Theory, Algorithms, and Systems", Prentice Hall Englewood Cliffs, New Jersey 07632, 1995.
4. Nico Artanto, "Aplikasi Hopfield Memory untuk Traveling Salesman Problem", Tugas Akhir, Jurusan Teknik Informatika, ITS, 1995.
5. Ramachandran Bharat, James Drosen, "Neural Network Computing", McGraw-Hill Tab Asian Student Edition, 1994.
6. Valluru B. Rao, Hayagriva V. Rao, "C++ Neural Networks and Fuzzy Logic", Management Information Source, Inc. A subsidiary of Henry Holt and Company, Inc. 115 West 18th Street New York, New York 10011, 1993.
7. Zhen-Ping Lo and Behnam Bavarian, "Multiple Job Scheduling with Artificial Neural Networks", Department of Electrical and Computer Engineering, University of California, Irvine, CA 92717, U.S.A., 1991.

LAMPIRAN A

FUNGSI OBYEKTIF DAN FUNGSI ENERGI

$$E_t = \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}\left(\frac{l + T_{ij} - 1}{C_k}\right)$$

$$\frac{\partial E_t}{\partial v_{ijl}} = \frac{1}{2}\left(\frac{l + T_{ij} - 1}{C_k}\right)$$

$$E_1 = \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{k}\sum_{j=1}^{m}\sum_{\substack{l_1=1\\ l_1\neq l}}^{k}\sum_{\substack{j_1=1\\ j_1\neq j}}^{m} v_{ijl}\left(v_{ij_1l} + v_{ijl_1} + v_{ij_1l_1}\right)$$

$$\frac{\partial E_1}{\partial v_{ijl}} = \frac{1}{2}\sum_{\substack{l_1=1\\ l_1\neq l}}^{k}\sum_{\substack{j_1=1\\ j_1\neq j}}^{m}\left(v_{ij_1l} + v_{ijl_1} + v_{ij_1l_1}\right)$$

$$E_2 = \frac{1}{2}\sum_{j=1}^{m}\sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{\substack{i_1=1\\ i_1\neq i}}^{n} v_{ijl}\, v_{i_1jl}$$

$$\frac{\partial E_2}{\partial v_{ijl}} = \frac{1}{2}\sum_{\substack{i_1=1\\ i_1\neq i}}^{n} v_{i_1jl}$$

$$E_3 = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k} v_{ijl} - n\right)^2$$

$$u = v_{ijl} - n$$

$$u' = 1$$

$$\frac{\partial E_3}{\partial v_{ijl}} = \frac{1}{2}2u'u = \sum_{i_1=1}^{n}\sum_{j_1=1}^{m}\sum_{l_1=1}^{k} v_{i_1j_1l_1} - n$$

$$E_4 = \frac{1}{2}\sum_{j=1}^{m} f_j^2 H(f_j)$$

$$f_j = 1 - \sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}$$

$$H(f) = \frac{1}{1+e^{-\gamma f}} = \frac{1}{1+e^{-\gamma\left(1-\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}\right)}}$$

$$u = 1 + e^{-\gamma} e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}}$$

$$u' = e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}}$$

$$v = v_{ijl}$$

$$\frac{\partial u^{-1}}{\partial v} = -1u^{-2}u' = -\frac{1}{\left(1+e^{-\gamma}e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}}\right)^2} e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}}$$

$$\frac{\partial E_4}{\partial v_{ijl}} = \frac{\partial\left(\frac{1}{2}\sum_{j=1}^{m} f_j^2 H(f_j)\right)}{\partial v_{ijl}}$$

$$= \frac{\partial\left(\frac{1}{2}\sum_{j=1}^{m}\left(1-\sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl}\right)H(f_j)\right)}{\partial v_{ijl}}$$

$$= \frac{\partial\left(\frac{1}{2}H(f_j) - \sum\sum\sum v_{ijl}H(f_j) + \frac{1}{2}\sum\sum\sum v_{ijl}^2 H(f_j)\right)}{\partial v_{ijl}}$$

$$= \begin{bmatrix} -\frac{1}{2}e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}}H^2(f_j) - H(f_j) + v_{ijl}e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}}H^2(f_j) + v_{ijl}H(f_j) \\ -\frac{1}{2}v_{ijl}^2 e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}}H^2(f_j) \end{bmatrix}$$

$$= \begin{bmatrix} \left[\frac{1}{2}H^2(f_j)\left[-e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}} + 2v_{ijl}e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}} - v_{ijl}^2 e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}}\right]\right] \\ + H(f_j)\left[-1+v_{ijl}\right] \end{bmatrix}$$

$$= \left[-H(f_j)f_j + \frac{1}{2}H^2(f_j)(-1+2v_{ijl}-v_{ijl}^2)\left(e^{-\gamma}\gamma e^{\gamma\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}}\right)\right]$$

$$= \left[-H(f_j)f_j + \frac{1}{2}H^2(f_j)(-1+2v_{ijl}-v_{ijl}^2)\left(e^{-\gamma}\gamma e^{\gamma\left(1-\sum_{i=1}^{n}\sum_{l=1}^{k}v_{ijl}\right)}\right)\right]$$

$$= -H(f_j) - \frac{1}{2}H^2(f_j)f_j^2 Ae^{-Af_j}$$

$$E_5 = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{m}v_{ij1} - m\right)^2 = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k}(v_{ijl}-m)\delta(l-1)\right)^2$$

$$u = (v_{ijl}-m)\delta(l-1)$$

$$u' = 1$$

$$\frac{\partial E_5}{\partial v_{ijl}} = \frac{1}{2}2u'u = \left(\sum_{i_1=1}^{n}\sum_{j_1=1}^{m}v_{i_1j_1l}\right)\delta(l-1)$$

$$E_6 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k} v_{ijl} g_{ijl}^2 H(g_{ijl})$$

$$\frac{\partial E_6}{\partial v_{ijl}} = g_{ijl}^2 H(g_{ijl})$$

$$E_7 = \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{k}\sum_{\substack{i_1=1\\ i_1\neq i}}^{n} v_{ijl}\left[h_1^2 H(h_1)\sum_{l_1=l}^{k} v_{ijl_1} + h_2^2 H(h_2)\sum_{\substack{l_2=1\\ l_2\neq l}}^{l-1} v_{ijl_2} \right]$$

$$\frac{\partial E_7}{\partial v_{ijl}} = \sum_{\substack{i_1=1\\ i_1\neq i}}^{n} v_{ijl}\left[h_1^2 H(h_1)\sum_{l_1=l}^{k} v_{ijl_1} + h_2^2 H(h_2)\sum_{\substack{l_2=1\\ l_2\neq l}}^{l-1} v_{ijl_2} \right]$$

$$f_j = 1 - \sum_{i=1}^{n}\sum_{l=1}^{k} v_{ijl},$$

$$g_{ijl} = l - 1 + T_{ij} - d_i,$$

$$T_{ij} = \frac{L_i}{m_j},$$

$$h_1 = (T_{ij} + l) - \sum_{l_1=l}^{k} l_1 v_{i_1 j l_1},$$

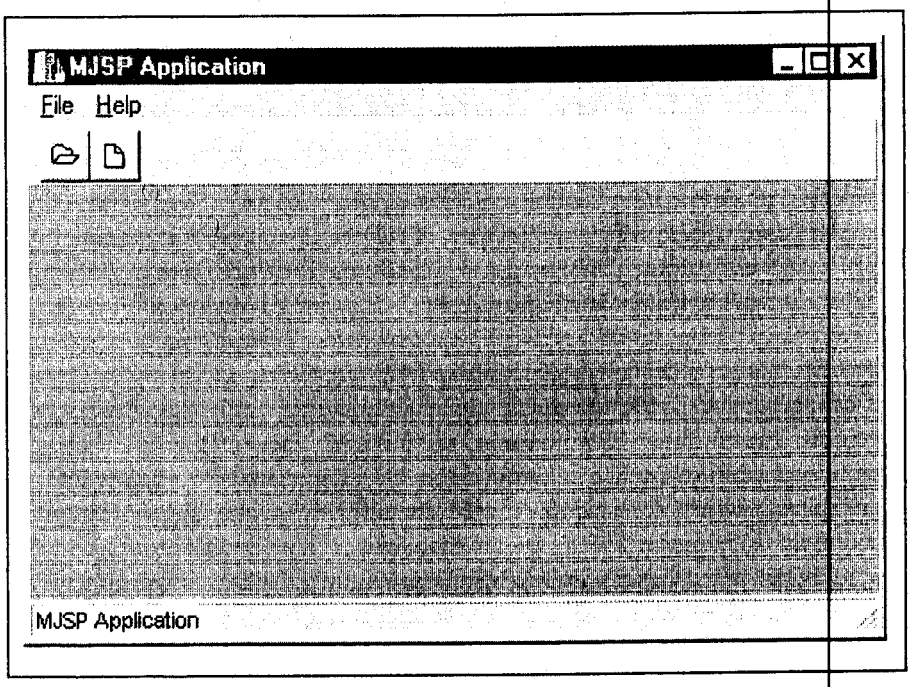
$$h_2 = \sum_{\substack{l_2=1\\ l_2\neq l}}^{l-1} (T_{i_1 j} + l_2) v_{i_1 j l_2} - l,$$

$$H(f) = \frac{1}{1+e^{-\gamma f}}$$

LAMPIRAN B

RANGKUMAN PETUNJUK PENGGUNAAN

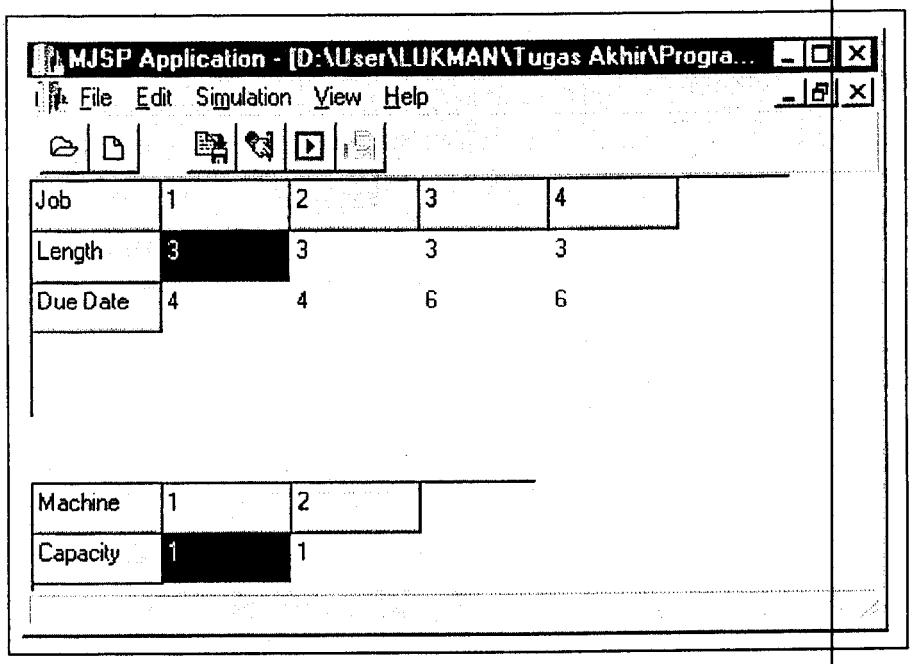
Gambar B.1 menampilkan perangkat lunak MJSP Application dengan halaman utamanya. Fasilitas yang aktif hanya untuk membuka sebuah berkas, membuat berkas baru, dan keluar dari program. Seperti dijelaskan dalam bab sebelumnya, perangkat lunak dibuat dengan Borland C++ Builder Professional[22] di atas sistem operasi Windows 95. Selanjutnya, manajemen memori dan penanganan piranti diserahkan pada sistem operasi.



Gambar B.1 Halaman Utama Perangkat Lunak *MJSP Application*

[22] http://www.inprise.com

Gambar B.2 menampilkan perangkat lunak MJSP Application lengkap dengan seluruh menu yang disediakan pada saat ada sebuah berkas terbuka. Berkas tersebut bisa berupa berkas yang sudah ada atau merupakan berkas yang baru dibuat. Semua fasilitas aktif kecuali fasilitas yang berkaitan dengan laporan. Fasilitas ini baru aktif jika sudah dilakukan pelatihan pada berkas.



Gambar B.2 Perangkat Lunak *MJSP Application* dengan Sebuah Berkas

Struktur menu dan operasi yang dilakukan ditunjukkan dalam tabel B.1 berikut ini.

Tabel B.1 Struktur Menu

Group Item	Menu Item	Operasi
File	New	Membuat berkas baru
	Open	Membuka berkas
	Save	Menyimpan berkas
	Save As	Menyimpan berkas dengan nama lain
	Printer Setup	Mengatur printer yang digunakan
	Print	Mencetak berkas
	Exit	Keluar dari program
Edit	Properties	Mengubah properti
	Modify	Memodifikasi berkas
	Parameters	Mengubah parameter
Simulation	Start	Menjalankan simulasi
View	Data Simulation Result	Menampilkan fungsi obyektif, fungsi biaya, dan fungsi energi
	Neurons Activation	Menampilkan nilai aktivasi sel saraf
	Job Assignment	Menampilkan hasil penugasan pada masing-masing mesin
	Schedule Result	Menampilkan jadual yang dihasilkan
Help	Content	Menampilkan informasi bantuan penggunaan perangkat lunak
	About	Menampilkan informasi perangkat lunak