APLIKASI ALGORITMA *ARTIFICIAL IMMUNE SYSTEM* (AIS) PADA PENJADWALAN *JOB SHOP* DALAM PEMBUATAN *SPRING BED* (STUDI KASUS PT. CAHAYA KAWI ULTRA POLYINTRACO)

SKRIPSI

Oleh
Shandiputra Budhi Perdana
NIM 071810101068

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2011

APLIKASI ALGORITMA *ARTIFICIAL IMMUNE SYSTEM* (AIS) PADA PENJADWALAN *JOB SHOP* DALAM PEMBUATAN *SPRING BED* (STUDI KASUS PT. CAHAYA KAWI ULTRA POLYINTRACO)

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk
menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

oleh
Shandiputra Budhi Perdana
NIM 071810101068

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2011

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Ibunda Dewi Siti Sukowati dan Ayahanda Budhi Nuryono tercinta, atas untaian dzikir dan do'a yang mengiringi langkahku selama menuntut ilmu, dukungan dan curahan kasih sayang yang telah diberikan sejak aku kecil, serta pengorbanan selama ini;
2. adik Sandraputra D.P., atas do'a dan dukungan yang telah diberikan kepada kakak selama ini;
3. guru-guru sejak taman kanak-kanak sampai dengan perguruan tinggi yang telah mendidik dengan penuh kesabaran;
4. almamater Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

MOTO

Dan, berkata orang-orang yang diberi ilmu pengetahuan dan keimanan (kepada orang-orang kafir): Sesungguhnya, kamu telah berdiam (dalam kubur) menurut ketetapan Allah, sampai hari berbangkit. *)

*) QS. *Ar-Rum*:56. *La Tahzan, Jangan Bersedih*. Terjemahan oleh Samson Rahman. 2003. Jakarta: Qisthi Press.

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

nama : Shandiputra Budhi Perdana

NIM : 071810101068

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul ”Aplikasi Algoritma *Artificial Immune System* (AIS) pada Penjadwalan *Job Shop* dalam Pembuatan *Spring Bed* (Studi Kasus PT. Cahaya Kawi Polyintraco)” adalah benar-benar hasil karya sendiri, kecuali kutipan yang telah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak mana pun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 24 Agustus 2011

Yang menyatakan,

Shandiputra Budhi Perdana
NIM 071810101078

SKRIPSI

APLIKASI ALGORITMA *ARTIFICIAL IMMUNE SYSTEM* (AIS) PADA PENJADWALAN *JOB SHOP* DALAM PEMBUATAN *SPRING BED* (STUDI KASUS PT. CAHAYA KAWI ULTRA POLYINTRACO)

oleh
Shandiputra Budhi Perdana
NIM 071810101068

Pembimbing

Dosen Pembimbing Utama : Agustina Pradjaningsih, S.Si., M.Si.

Dosen Pembimbing Anggota : Drs. Rusli Hidayat, M.Sc.

PENGESAHAN

Skripsi berjudul ”Aplikasi Algoritma *Artificial Immune System* (AIS) pada Penjadwalan *Job Shop* dalam Pembuatan *Spring Bed* (Studi Kasus PT. Cahaya Kawi Polyintraco)” telah diuji dan disahkan pada:

hari, tanggal : Kamis, 8 September 2011

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Jember

Tim Penguji:

Ketua,	Sekretaris,
Agustina Pradjaningsih, S.Si, M.Si. NIP 19710802 200003 2 009	Drs. Rusli Hidayat, M.Sc. NIP 19661012 199303 1 001
Penguji I,	Penguji II,
Drs. Moh. Hasan, MSc., PhD. NIP 19640404 198802 1 001	Kosala Dwidja P., S.Si., M.Si. NIP 19690828 199802 1 001

Mengesahkan

Dekan,

Prof. Drs. Kusno, DEA., Ph.D.
NIP 19610108 198602 1 001

Aplikasi Algoritma *Artificial Immune System* (AIS) pada Penjadwalan *Job Shop* dalam Pembuatan *Spring Bed* (Studi Kasus PT. Cahaya Kawi Polyintraco); Shandiputra Budhi Perdana, 071810101068; 2011: 44 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Permasalahan yang terjadi pada industri manufaktur secara umum mengacu pada permasalahan *job shop*, yaitu suatu proses sejumlah pekerjaan pada sejumlah mesin yang terjadi pada suatu sistem yang mempunyai *routing* berbeda dan sifatnya acak. Permasalahan seputar penjadwalan akan timbul apabila beberapa pekerjaan (*jobs*) dikerjakan bersamaan, sedangkan sumber daya seperti mesin atau peralatan yang dimiliki jumlahnya terbatas. Untuk mencapai hasil yang optimal dengan keterbatasan sumber daya yang dimiliki, maka diperlukan adanya penjadwalan sumber-sumber tersebut secara efektif, yaitu dengan membangun jadwal yang memiliki waktu penyelesaian semua tugas atau pekerjaan (*makespan*) optimal. Oleh karena itu, dalam skripsi ini dibahas penyelesaian *job shop* dengan algoritma AIS untuk membangun jadwal dengan *makespan* yang optimal. Tujuan penelitian adalah untuk mengaplikasikan algoritma *Artificial Immune System* (AIS) pada *job shop* dan membuat program algoritma AIS untuk menyelesaikan *job shop* dengan PHP & *MySQL*

Penelitian dilakukan melalui beberapa langkah, yaitu mengolah data yang diperoleh menjadi data urutan mesin dan waktu proses kemudian menentukan urutan setiap pekerjaan pada mesin secara acak. Selanjutnya membuat algoritma pemrograman dari masalah penjadwalan *job shop* menggunakan algoritma AIS, membuat program berdasarkan algoritma menggunakan bahasa pemrograman PHP, dan terakhir adalah menampilkan jadwal dengan *makespan* paling optimal dengan menggunakan program yang telah dibuat.

Program *Job Shop Application* yang dibuat dengan menggunakan algoritma AIS ternyata berhasil untuk menyelesaikan masalah penjadwalan *job shop*, algoritma

ini juga dapat menemukan jadwal dengan *makespan* optimal lainnya (tidak merupakan solusi tunggal) pada pembuatan *spring bed* di PT. Cahaya Kawi Polyintraco. Salah satu jadwal optimal tersebut adalah 3-1-3-4-4-1-4-1-2-2-1-2-2-3-4-3+1-2-3-4 dengan total *makespan* 433 menit. Hasil tersebut diperoleh dari program yang telah dibuat menggunakan algoritma AIS dengan bantuan PHP dimana *input* berupa banyaknya pekerjaan, mesin yang akan digunakan, data waktu proses pada setiap mesin, banyaknya perpustakaan antibodi, komponen, dan gen, sedangkan *output* berupa *makespan* jadwal, jumlah iterasi, waktu komputasi, dan sekumpulan jadwal optimal. Program tersebut dapat mempermudah menemukan jadwal dengan *makespan* optimal terutama untuk jumlah pekerjaan dan mesin yang relatif besar dalam hal ini ukuran masalah adalah maksimal 9 pekerjaan pada 9 mesin.

PRAKATA

Puji syukur ke hadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini yang berjudul "Aplikasi Algoritma *Artificial Immune System* (AIS) pada Penjadwalan *Job Shop* dalam Pembuatan *Spring Bed* (Studi Kasus PT. Cahaya Kawi Polyintraco)". Skripsi ini disusun untuk memenuhi salah satu syarat untuk menyelesaikan Program Studi Matematika (S1) dan mencapai gelar Sarjana Sains.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Agustina Pradjaningsih, S.Si., M.Si., selaku Dosen Pembimbing Utama dan Drs. Rusli Hidayat, M.Sc., selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
2. Drs. Moh. Hasan, M.Sc., Ph.D., selaku Dosen Penguji I dan Kosala Dwidja P., S.Si., M.Si., selaku Dosen Penguji II yang telah memberikan kritik dan saran demi kesempurnaan skripsi ini;
3. keluarga di rumah yang telah memberikan dukungan moril;
4. Titi Hayatina, Medhi Amalia, Novika Hertianti, dan teman-teman angkatan lainnya;
5. semua pihak yang tidak dapat disebutkan satu per satu.

Penulis menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap semoga skripsi ini dapat bermanfaat.

Jember, Agustus 2011 Penulis

DAFTAR ISI

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Masalah penjadwalan merupakan tantangan besar bagi suatu industri manufaktur. Hal ini menjadi penting karena penjadwalan berkaitan dengan ketepatan waktu penyelesaian *order* atau pesanan dan pendistribusian kepada pemesan. Keterlambatan yang dikarenakan jadwal yang buruk mengakibatkan kerugian karena akan menambah biaya produksi pembuatan barang. Serta hilangnya kepercayaan pemesan untuk *order* barang akan berakibat menurunnya pesanan, sehingga mengurangi keuntungan perusahaan di masa mendatang.

Penjadwalan adalah proses mengoptimalkan penugasan dari serangkaian tugas kepada sekelompok sumber daya yang terbatas. Tujuan utama dari penjadwalan umumnya untuk mengoptimalkan dimensi tertentu, yaitu *makespan* (waktu penyelesaian semua tugas atau pekerjaan), penundaan, keterlambatan, keuntungan bagi perusahaan, dan lain-lain. Penjadwalan kegiatan produksi bergantung pada sumber daya yaitu mesin, mesin memproses pekerjaan tetapi juga dapat mencakup kegiatan tenaga kerja untuk mengoperasikan mesin tersebut.

Permasalahan yang terjadi pada industri manufaktur secara umum mengacu pada permasalahan *job shop*, yaitu pemrosesan sejumlah pekerjaan pada sejumlah mesin yang terjadi pada suatu sistem yang mempunyai urutan proses berbeda dan sifatnya acak. Permasalahan seputar penjadwalan akan timbul apabila beberapa pekerjaan (*jobs*) dikerjakan bersamaan, sedangkan sumber daya seperti mesin atau peralatan yang dimiliki jumlahnya terbatas. Untuk mencapai hasil yang optimal dengan keterbatasan sumber daya yang dimiliki, maka diperlukan adanya penjadwalan sumber-sumber tersebut secara efektif. Dalam hal ini, tujuan penjadwalan adalah membangun suatu jadwal yang memiliki *makespan* optimal untuk permasalahan *m* pekerjaan pada *n* mesin sehingga proses pengerjaan terjadwal secara efektif. Tingkat kesulitan untuk membangun sebuah jadwal *job shop* relatif

tinggi, hal ini dikarenakan semua pekerjaan akan masuk ke sebagian besar mesin sehingga ada kemungkinan pekerjaan akan menunggu pada mesin yang sedang berproses atau sedang sibuk. Jika proses itu terjadi, dapat mengakibatkan produktifitas di dalam sistem menjadi rendah (Conway, 1967).

Berdasarkan penelitian terdahulu, masalah penjadwalan *job shop* dapat diselesaikan dengan 2 macam teknik, yaitu teknik eksak dan teknik pendekatan heuristik. Teknik eksak meliputi algoritma *branch and bound* yang dikembangkan oleh Land dan Doig pada tahun 1960, tetapi algoritma tersebut tidak menjamin solusi atau jadwal yang terbentuk adalah optimal dan juga memerlukan waktu lama untuk mencapai solusi yang optimal (Bondal, 2008). Teknik lain yang dapat dipergunakan adalah teknik pendekatan heuristik, antara lain *priority dispatch rules*, *bottleneck based heuristik*, *oportunistic schedulling*, dan *artificial intelligence*. Menurut penulis, teknik pendekatan heuristik menarik untuk dibahas, salah satu diantaranya adalah *artificial intelligence* atau kecerdasan buatan. Di bidang penjadwalan, terutama masalah penjadwalan *job shop*, *artificial intelligence* adalah algoritma yang evolusioner. Algoritma evolusioner adalah algoritma yang terinspirasi oleh evolusi biologi, yaitu reproduksi, mutasi, rekombinasi, dan seleksi. Algoritma *artificial intelligence* atau kecerdasan buatan meliputi algoritma genetika (GA), *artificial immune system* (AIS), jaringan syaraf tiruan, tabu *search, simulated annealing*, dan lain-lain.

Pada skripsi ini, penulis tertarik untuk mengaplikasikan salah satu algoritma pada *artificial intelligence*, yaitu algoritma AIS. Algoritma ini terinspirasi oleh prinsip kerja sistem imun manusia, menurut penelitian sebelumnya, Coello *et al* (tanpa tahun) telah menggunakan algoritma AIS untuk masalah *job shop* dan membandingkannya dengan algoritma GRASP (*Greedy Randomized Adaptive Search Procedure*) atau algoritma Greedy. Pada penelitian tersebut, Coello *et al* hanya menerapkan prinsip seleksi klonal, sedangkan pada skripsi ini penulis mencoba menerapkan prinsip seleksi klonal sekaligus prinsip seleksi positif. Penerapan prinsip seleksi positif bertujuan mempercepat algoritma untuk mencapai hasil optimal karena

prinsip tersebut akan mempersempit ruang cari solusi yang layak pakai. Menurut Coello *et al*, secara umum AIS lebih sering mampu mencapai solusi optimal sebesar 38,7% dibandingkan GRASP yang hanya sebesar 35,4%. Untuk rata-rata persentase kesalahan AIS lebih kecil daripada GRASP yaitu 2,2% dan 2,3%.

Pada skripsi ini, AIS akan diaplikasikan pada masalah penjadwalan *job shop* dengan diberikan rincian tentang pekerjaan, urutan mesin, dan waktu proses mesin. Tujuannya adalah menemukan jadwal dengan *makespan* optimal untuk masalah *n* mesin dan *m* pekerjaan. Berdasarkan data yang diperoleh untuk kasus ini, jumlah pekerjaan, dan mesin ditentukan, yaitu 4 pekerjaan pada 9 jenis mesin. Berdasarkan proses yang terjadi di pabrik *spring bed*, pembuatan *spring bed* adalah proses pembuatan dan perakitan komponen *spring bed* yang terdiri dari matras, sandaran, dan *divan spring bed.* Sembilan jenis mesin yang digunakan untuk membuat dan merakit komponen *spring bed* secara bersamaan memiliki urutan proses tersendiri, dimana proses pembuatannya tidak dapat diubah urutannya. Jika urutannya diubah, maka proses pembuatan dan perakitan *spring bed* akan kacau karena ada pekerjaan yang saling mendahului (*overlap*) dengan pekerjaan lainnya yang merupakan prioritas. Untuk menghindari *overlap*, mesin-mesin yang berkerja harus diatur sedemikian rupa sehingga pembuatan komponen *spring bed* berjalan secara *independent* (tidak saling bergantung), kecuali proses perakitan yang dilakukan setelah pembuatan komponen selesai. Ringkasnya, sejumlah mesin akan dikelompokkan dari 9 jenis mesin menjadi 5 mesin, yaitu mesin yang memproses pembuatan matras, mesin yang memproses sandaran, mesin yang memproses *divan*, mesin yang memproses kaki *spring bed*, dan mesin yang memproses perakitan *spring bed* dengan cara memisalkan beberapa mesin dan menyimpannya kedalam notasi mesin-mesin yang baru, dengan demikian ukuran masalah pada skripsi ini menjadi 4 pekerjaan pada 5 mesin (4 mesin untuk pembuatan komponen dan 1 mesin perakitan).

Algoritma AIS ini menggunakan bahasa pemrograman PHP untuk membuat program karena memiliki banyak keunggulan dibandingkan bahasa pemrograman yang lain. Selain *open source*, jika program telah terkoneksi dengan internet, program

ini tidak memperdulikan sistem operasi yang sedang digunakan oleh pengguna saat menjalankannya.

1.2 Perumusan Masalah

Berdasarkan data yang diperoleh untuk kasus ini, permasalahan yang akan dibahas adalah menemukan jadwal pada permasalahan *job shop* untuk 4 pekerjaan pada 5 mesin dengan waktu proses (*makespan*) yang optimal menggunakan algoritma AIS.

1.3 Tujuan

Tujuan yang ingin dicapai dalam penulisan skripsi ini adalah:

1. Mengaplikasikan algoritma AIS pada permasalahan penjadwalan *job shop*.
2. Menemukan jadwal pembuatan 4 tipe *spring bed* yang memiliki *makespan* optimal.
3. Membuat program algoritma AIS dengan bantuan bahasa pemrograman PHP.

1.4 Manfaat

Adapun manfaat yang diharapkan dari penulisan skripsi ini adalah sebagai berikut.

a. Memberikan alternatif metode penyelesaian *job shop* menggunakan algoritma AIS.
b. Dengan meminimasi *makespan*, maka diharapkan pengguna akan dapat mengoptimalkan biaya produksi.
c. Dengan adanya program, permasalahan *job shop* lebih mudah diselesaikan.

BAB 2. TINJAUAN PUSTAKA

2.1 Penjadwalan Manufaktur

Penjadwalan adalah proses pengurutan pembuatan produk secara menyeluruh pada beberapa mesin dengan menggunakan sumber daya yang terbatas (Conway *et al*, 1967). Sumber daya ini tidak hanya berupa mesin yang menjalankan pekerjaan tetapi juga dapat mencakup tenaga kerja yang diperlukan untuk mengoperasikan mesin tersebut. Penjadwalan juga dapat didefinisikan sebagai pengambilan keputusan tentang penyesuaian aktivitas dan sumber daya dalam rangka menyelesaikan sekumpulan *jobs*/proyek agar tepat pada waktunya dan memiliki kualitas seperti yang diinginkan (Morton,1993). Keputusan yang dibuat dalam penjadwalan meliputi:

a. pengurutan pekerjaan (*sequencing*);
b. waktu mulai dan selesai pekerjaan (*timing*);
c. urutan operasi untuk suatu pekerjaan (*routing*).

Menurut Baker (1974), secara umum tujuan penjadwalan adalah sebagai berikut:

a. mengurangi persediaan barang setengah jadi dengan cara mengurangi jumlah pekerjaan yang menunggu dalam antrian di mesin yang sedang sibuk;
b. meningkatkan produktivitas mesin dengan mengurangi waktu menganggur;
c. mengurangi keterlambatan karena telah melampaui batas waktu;
d. pemenuhan waktu dimana suatu produk harus selesai diproses atau diproduksi (*due date*).

Penjadwalan mempunyai beberapa elemen penting yang harus diperhatikan antara lain sebagai berikut.

a. Pekerjaan (*job*)

   Job dapat didefinisikan sebagai sebuah pekerjaan yang harus diselesaikan untuk mendapatkan suatu produk. *Job* biasanya terdiri dari beberapa operasi yang harus dikerjakan (minimal 1 operasi). Perencanaan yang telah dibuat atau berdasarkan

pesanan dari pelanggan, memberikan *job* kepada bagian *shop floor* untuk dikerjakan. Informasi yang dimiliki oleh *job* ketika datang ke bagian *shop floor* biasanya adalah operasi-operasi yang harus dilakukan didalamnya (dari bagian *engineering*), yaitu saat *job* harus diselesaikan dan saat *job* mulai dapat dikerjakan.

b. Operasi

Operasi adalah himpunan bagian dari *job* untuk menyelesaikan *job*. Operasi-operasi dalam *job* diurutkan dalam suatu urutan pengerjaan tertentu. Urutan tersebut ditentukan pada saat perencanaan proses. Suatu operasi baru dapat dikerjakan apabila operasi atau proses yang mendahuluinya sudah dikerjakan terlebih dahulu.

c. Mesin

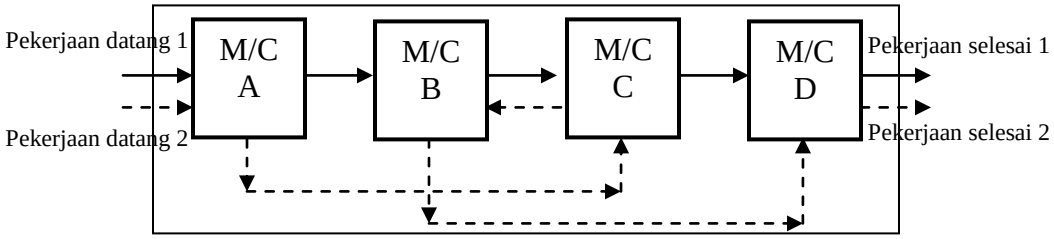
Mesin adalah sumber daya yang diperlukan untuk mengerjakan proses penyelesaian suatu *job*. Setiap mesin hanya dapat memproses satu *job* pada satu kali proses.

2.1.1 Penjadwalan *Job Shop*

Menurut Conway (1967), *job shop* adalah suatu masalah dimana setiap pekerjaan mempunyai pola yang berbeda dan sifatnya acak. Dalam hal ini, pola acak adalah urutan dimana suatu pekerjaan akan diproses tidak selalu dimulai pada mesin pertama, tetapi pekerjaan dapat dimulai pada mesin yang lain. Hal ini membuat proses penjadwalan lebih sulit karena semua pekerjaan akan masuk ke sebagian besar mesin sehingga ada kemungkinan pekerjaan tersebut akan menunggu pada mesin yang sedang berproses (sedang sibuk). Jika proses itu terjadi, maka mengakibatkan produktifitas dalam sistem akan rendah.

Secara umum, *job shop* merupakan suatu pemrosesan sejumlah pekerjaan pada sejumlah mesin yang terjadi pada suatu sistem. Pekerjaan-pekerjaan tersebut hanya diproses tepat satu kali pada setiap mesin, tetapi dimungkinkan suatu pekerjaan tidak diproses menggunakan semua mesin tergantung operasi yang diperlukan untuk menyelesaikan produk tersebut. Namun, jenis masalah *job shop* yang dibahas dalam

skripsi ini merupakan tipe khusus yang disesuaikan dengan barang yang akan diproduksi dan memiliki bentuk unik, yaitu setiap pekerjaan diproses pada setiap mesin tepat satu kali dan bersifat saling asing, yaitu setiap operasi pada mesin tidak bergantung terhadap operasi pada mesin lainnya. Gambar 2.1 berikut menunjukkan gambaran tentang *job shop*.



Gambar 2.1 Proses penjadwalan *job shop*

Terdapat 2 pekerjaan yang akan diproses pada 4 mesin, yaitu mesin *A*, *B*, *C*, dan *D* dimana urutan (*routing*) pekerjaan 1 dan 2 berbeda. Seperti pada proses di atas, pekerjaan 1 memiliki *routing* sebagai berikut, pada operasi pertama pekerjaan masuk ke mesin *A*, lalu operasi kedua pekerjaan masuk ke mesin *B*, operasi ketiga pekerjaan masuk ke mesin *C*, dan kemudian operasi keempat pekerjaan masuk ke mesin *D*. Sedangkan pekerjaan 2 memiliki *routing* sebagai berikut, pada operasi pertama pekerjaan masuk ke mesin *A*, lalu operasi kedua pekerjaan masuk ke mesin *C*, operasi ketiga pekerjaan masuk ke mesin B, dan kemudian operasi keempat pekerjaan masuk ke mesin *D*. Untuk *routing* pekerjaan yang bersamaan, akan diprioritaskan pada pekerjaan yang pertama kali masuk kedalam mesin.

Berikut diberikan definisi formal dari *job shop*, menurut Coello *et al* (tanpa tahun), terdapat satu set *n* pekerjaan (*J*), $\{J_x\}_{1\leq x\leq n}$ yang harus diproses dalam *m* mesin (*M*), $\{M_y\}_{1\leq y\leq m}$. Setiap pekerjaan memiliki urutan yang bergantung pada kendala prioritas yang ada. Pengolahan pekerjaan dalam mesin disebut operasi (*O*). Operasi O_{xy} memerlukan penggunaan *M* untuk jangka waktu proses (*P*) tak terputus P_{xy}. Sebuah jadwal (*c*) merupakan satu kali durasi untuk setiap operasi, $\{c_{xy}\}_{1\leq x\leq n,1\leq y\leq m}$ yang memenuhi indikasi kondisi sebelumnya. Waktu total durasi

yang dibutuhkan untuk menyelesaikan semua pekerjaan (*makespan*) disebut *L*, sehingga tujuan akhirnya adalah meminimalkan *L* dengan menemukan jadwal dengan *makespan* optimal. *Makespan* jadwal yang optimal memiliki ciri, yaitu jadwal pekerjaan yang meminimalkan waktu menganggur atau pekerjaan di dalam mesin-mesin produksi dibuat serapat mungkin tetapi di dalamnya tidak ada pekerjaan yang saling *overlap* atau mendahului.

Masalah penjadwalan *job shop* merupakan salah satu masalah optimasi kombinatorika karena jadwal diperoleh dengan cara permutasi antara banyak perkerjaan dengan banyak mesin. Menurut Tirta (2004), apabila ada persyaratan bahwa lokasi yang telah dipilih (ditempati) suatu objek (pekerjaan) tidak bisa dipilih (ditempati) objek lain lagi, atau suatu objek hanya bisa menempati satu tempat, maka persoalaan tersebut disebut permutasi. Permutasi tersebut terjadi di seluruh pekerjaan yang dikerjakan di sembarang mesin secara bersama-sama, sehingga kasus tersebut masuk dalam dalam jenis permutasi dengan unsur yang identik. Terkadang tidak semua unsur dalam permutasi dapat dibedakan, unsur-unsur ini adalah unsur-unsur yang identik atau sama secara kualitas.

Misalkan, suatu permasalahan *job shop* dengan ukuran 2×2 (2 pekerjaan pada 2 mesin) akan terdiri dari 4 macam unsur, yaitu $1, 1, 2, 2$. Unsur 1 muncul sebanyak dua kali dan unsur 2 muncul sebanyak dua kali, keduanya memiliki unsur identik sebanyak 2 buah, maka permutasi dari jadwal $1, 1, 2, 2$ adalah berjumlah $(2 \times 2)! = 4! = 24$ permutasi.

Tabel 2.1 Hasil permutasi unsur $1_a, 1_b, 2_a, 2_b$

No.	**Hasil Permutasi**			
1.	$1_a\,1_b\,2_a\,2_b$	$1_b\,1_a\,2_a\,2_b$	$1_b\,1_a\,2_b\,2_a$	$1_a\,1_b\,2_b\,2_a$
2.	$2_a\,2_b\,1_a\,1_b$	$2_b\,2_a\,1_a\,1_b$	$2_b\,2_a\,1_b\,1_a$	$2_a\,2_b\,1_b\,1_a$
3.	$1_a\,2_b\,1_a\,2_b$	$1_b\,2_a\,1_a\,2_b$	$1_b\,2_a\,1_b\,2_a$	$1_a\,2_b1_b\,2_a$
4.	$2_a\,1_b\,2_a\,1_b$	$2_b\,1_a\,2_a\,1_b$	$2_b\,1_a\,2_b\,1_a$	$2_a\,1_b\,2_b\,1_a$
5.	$1_a\,2_b\,2_a\,1_b$	$1_b\,2_a\,2_a\,1_b$	$1_b\,2_a\,2_b\,1_a$	$1_a\,2_b\,2_b\,1_a$
6.	$2_a\,1_b\,1_a\,2_b$	$2_b\,1_a\,1_a\,2_b$	$2_b\,1_a\,1_b\,2_a$	$2_a\,1_b\,1_a\,2_b$

Total permutasi ini juga mencakup posisi 1_a dan 1_b yang saling bertukar dan berjumlah 4 (karena 1 terdiri dari 2 unsur, yaitu 1_a dan 1_b). Dengan demikian berdasarkan Tabel 2.1, jika dianggap $1_a = 1_b$ dan $2_a = 2_b$, maka akan menjadi:

- $1_a\,1_b\,2_a\,2_b = 1_b\,1_a\,2_a\,2_b = 1_b\,1_a\,2_b\,2_a = 1_a\,1_b\,2_b\,2_a = 1\ 1\ 2\ 2$
- $2_a\,2_b\,1_a\,1_b = 2_b\,2_a\,1_a\,1_b = 2_b\,2_a\,1_b\,1_a = 2_a\,2_b\,1_b\,1_a = 2\ 2\ 1\ 1$
- $1_a\,2_b\,1_a\,2_b = 1_b\,2_a\,1_a\,2_b = 1_b\,2_a\,1_b\,2_a = 1_a\,2_b\,1_b\,2_a = 1\ 2\ 1\ 2$
- $2_a\,1_b\,2_a\,1_b = 2_b\,1_a\,2_a\,1_b = 2_b\,1_a\,2_b\,1_a = 2_a\,1_b\,2_b\,1_a = 2\ 1\ 2\ 1$
- $1_a\,2_b\,2_a\,1_b = 1_b\,2_a\,2_a\,1_b = 1_b\,2_a\,2_b\,1_a = 1_a\,2_b\,2_b\,1_a = 1\ 2\ 2\ 1$
- $2_a\,1_b\,1_a\,2_b = 2_b\,1_a\,1_a\,2_b = 2_b\,1_a\,1_b\,2_a = 2_a\,1_b\,1_a\,2_b = 2\ 1\ 1\ 2$

Sehingga hasil permutasinya menjadi:

$$\frac{(2 \times 2)!}{2! \times 2!} = \frac{4!}{2 \times 2} = \frac{24}{4} = 6$$

Perhitungan ini dapat digeneralisasikan untuk ukuran $r = n \times m$, dimana n adalah jumlah pekerjaan dan m adalah jumlah mesin. Jadwal *job shop* dengan ukuran r memiliki jadwal sebanyak $P(r,r) = \frac{r!}{r_1!r_2!\cdots r_k!}$ (Tirta, 2004), jika disesuaikan dengan ukuran masalah secara umum pada *job shop* dengan $r = n \times m$, maka secara teoritis *job shop* akan menghasilkan $\frac{(n\times m)!}{m_1!m_2!\cdots m_n!}$ alternatif jadwal layak pakai (*feasible*).

Menurut Agam *et al* (tanpa tahun), suatu jadwal *job shop* dikatakan layak apabila memenuhi kriteria berikut:

a. urutan pengerjaan operasi pada setiap *job* cocok dengan *routing* yang telah ditetapkan;
b. tidak ada *overlap* waktu pengerjaan pada operasi yang berpadanan pada mesin yang sama.

2.1.2 Diagram *Gantt*

Diagram *gantt* pertama kali diperkenalkan oleh Henry Laurence Gantt pada tahun 1916. Menurut Gantt (1916), diagram *gantt* adalah diagram yang menampilkan

durasi dari tugas terhadap perubahan waktu. Tujuan dibuatnya diagram *gantt* antara lain.

a. Menentukan durasi pekerjaan terhadap perkembangan waktu.
b. Perencanaan dan penjadwalan proyek pekerjaan.
c. Pemantauan kemajuan proyek pekerjaan.

Setiap jadwal yang dibangun memiliki *makespan* total yang bebeda-beda, sedangkan tujuan algoritma AIS adalah untuk mendapatkan jadwal yang memiliki *makespan* yang paling optimal. Pada penjadwalan *job shop*, *makespan* total suatu jadwal pekerjaan dapat dihitung menggunakan diagram *gantt*. Misal, terdapat sebuah masalah *job shop* untuk 3 pekerjaan pada 3 mesin dengan panjang jadwal lengkap 3×3 berisi total 9 digit jadwal.

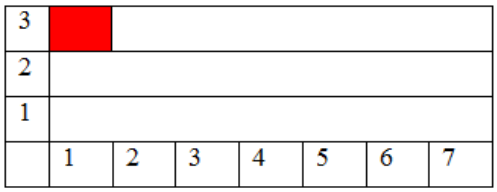
Tabel 2.2 Nomor urut mesin (*M*) dan waktu proses (*T*) ukuran 3×3

Jobs	**Operasi**		
	***Opn.1* (*M,T*)**	***Opn.2* (*M,T*)**	***Opn.3* (*M,T*)**
1	(2,3)	(1,3)	(3,2)
2	(3,1)	(2,2)	(1,3)
3	(1,4)	(3,2)	(2,1)

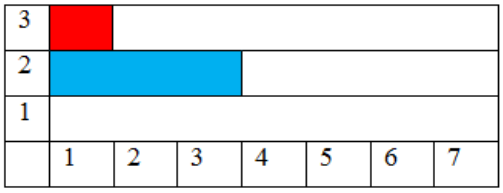
Tabel di atas adalah data awal yang berisi pasangan mesin dan waktu yang diperlukan pada permasalahan *job shop* dengan ukuran 3×3. Pekerjaan 1 operasi pertama, pekerjaan diproses di mesin 2 dengan waktu sebesar 3 menit. Selanjutnya operasi kedua diproses pada mesin 1 dengan waktu sebesar 3 menit, dan terakhir diproses pada mesin 3 dengan waktu sebesar 2 menit. Analogi dengan sebelumnya, pekerjaan 2 secara berturut-turut diproses pada mesin 3, mesin 2, lalu ke mesin 1 dengan waktu sebesar 1 menit, 2 menit, dan 3 menit. Sedangkan pekerjaan 3 secara berturut-turut diproses pada mesin 1, mesin 3, lalu ke mesin 2 dengan waktu sebesar 4 menit, 2 menit, dan 1 menit.

Untuk lebih memahami proses menghitung *makespan*, berikut diberikan jadwal parsial (misalnya, hanya 5 digit pertama dari sebuah jadwal lengkap 9 digit

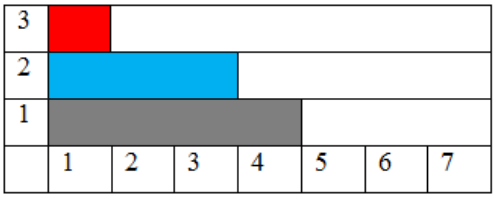
yang dihasilkan oleh jadwal acak) yaitu 2-1-3-3-1, maka untuk menghitung *makespan* jadwal parsial tersebut dibuatlah sebuah diagram *gantt* seperti di bawah ini.



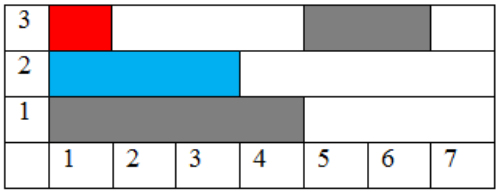
Gambar 2.2a



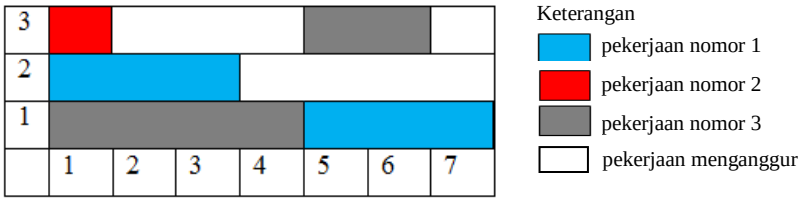
Gambar 2.2b



Gambar 2.2c



Gambar 2.2d



Gambar 2.2e

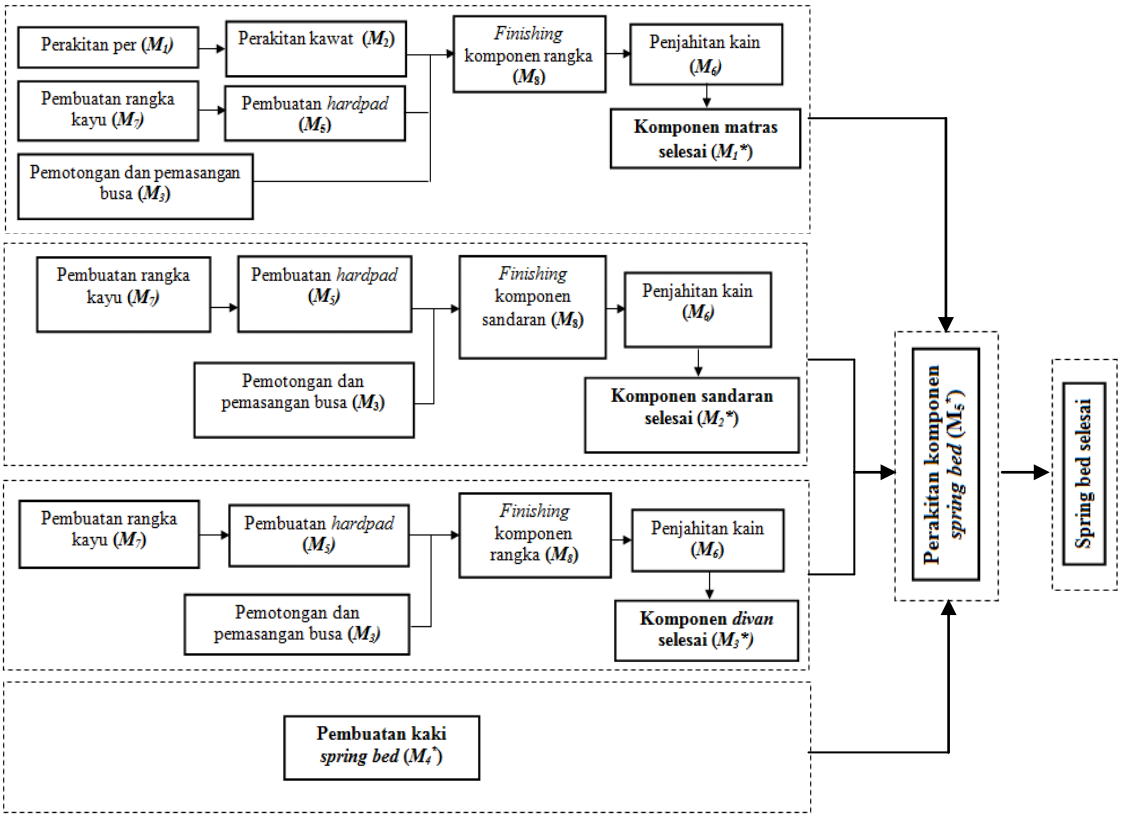
Gambar 2.2 Proses perhitungan *makespan* jadwal dengan diagram *gantt*

Diagram *gantt* di atas digunakan untuk menghitung *makespan* jadwal parsial acak 2-1-3-3-1. Urutan pertama yaitu mengacu pada pekerjaan 2, pada operasi pertama pekerjaan tersebut diproses pada mesin 2 dengan waktu proses 3 unit waktu. Berikutnya pekerjaan 1, pada operasi pertama diproses pada mesin 2 dengan waktu proses 3 unit waktu. Kemudian dilanjutkan pekerjaan 3 karena urutan ini berulang 2 kali, akan diproses bergantian. Pekerjaan 3 yang pertama, untuk operasi pertama diproses pada mesin 1 dengan waktu proses 4 unit waktu. Lalu pada pekerjaan 3 yang kedua akan melanjutkan pekerjaan sebelumnya yaitu diproses pada mesin 3 dengan waktu proses 2 unit waktu. Terakhir, pekerjaan 1 akan diproses pada mesin 1 dengan waktu proses 3 unit waktu. Terlihat pada diagram *gantt* terdapat waktu menganggur yang ditandai oleh arsiran putih dengan waktu menganggur pada mesin 1 sebesar 0

unit waktu, pada mesin 2 sebesar 4 unit waktu, dan pada mesin 3 sebesar 4 unit waktu, sedangkan *makespan* yang dihasilkan jadwal adalah 7 unit waktu.

2.1.3 Proses Produksi Industri *Spring Bed*

Permasalahan penjadwalan *job shop* akan diaplikasikan untuk menentukan jadwal pada pembuatan 4 tipe *spring bed*. Menurut Junida (2009)**,** secara umum proses pembuatan *spring bed* di PT. Cahaya Kawi Ultra Polyintraco diklasifikasikan menjadi 3 tahapan utama, yaitu pembuatan matras *spring bed*, pembuatan sandaran *spring bed*, dan pembuatan *divan spring bed* dimana setiap tahapan terdiri dari berbagai macam proses pembuatan komponen kemudian dirakit pada akhir proses pembuatan seperti pada gambar di bawah ini.



Gambar 2.3 Proses pembuatan *spring bed* PT. Cahaya Kawi Ultra Polyintraco

Mesin dalam skripsi ini dapat berupa suatu proses pekerjaan pembuatan komponen atau perakitan pada pembuatan *spring bed* yang dapat dilakukan oleh mesin ataupun tenaga manusia yang disimbolkan sebagai $M_1^*, M_2^*, M_3^*,$ dan M_4^*. Sedangkan M_5^* akan dianggap sebagai sebuah mesin yang akan merakit komponen (perakitan *spring bed*) pada pekerjaan 1, 2, 3, dan 4 yang dilakukan oleh sekumpulan tenaga manusia. Berikut mesin-mesin yang digunakan untuk memproduksi *spring bed*.

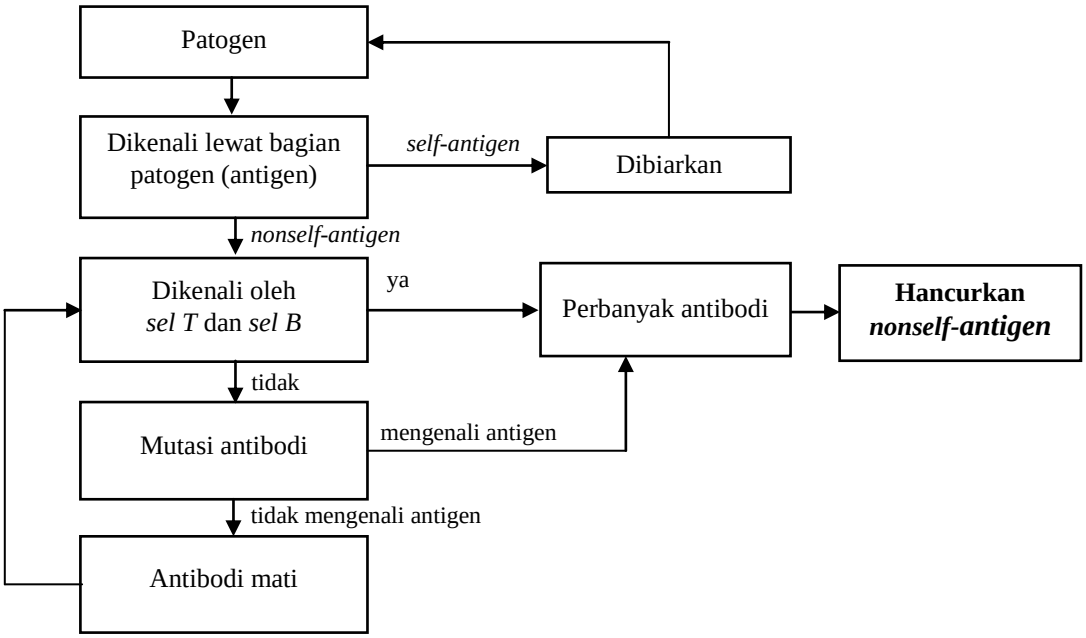
a. Mesin perakitan pegas (M_1)
b. Mesin perakitan kawat (M_2)
c. Mesin pemotongan dan pemasangan busa (M_3)
d. Mesin pembuatan kaki *spring bed* (M_4)
e. Mesin pembuatan *hardpad* (M_5)
f. Mesin penjahitan kain (M_6)
g. Mesin pembuatan rangka kayu (M_7)
h. Mesin *finishing* komponen rangka (M_8)
i. Mesin perakitan *spring bed* (M_9)

2.2 *Artificial Immune System* (AIS)

Artificial Immune System (AIS) dikembangkan pada tahun 1986 oleh Farmer *et al.* yang terinspirasi oleh sistem kekebalan tubuh manusia. Algoritma AIS meniru perilaku dan sifat sel-sel kekebalan, khususnya sel B, sel T,dan antigen. Menurut Aickelin dan Dasgupta (2005), prinsip sistem kekebalan tubuh manusia adalah mengenali benda asing yang masuk ke dalam tubuh manusia atau biasa disebut patogen yang dapat mengganggu kestabilan tubuh. Patogen sendiri tidak dapat langsung dikenali tetapi dapat dideteksi melalui bagian patogen yang disebut antigen. Jika terdeteksi adanya patogen, sistem kekebalan tubuh bertugas untuk mengeleminasinya dari tubuh. Agar proses eliminasi patogen berlangsung dengan baik dan benar, sistem kekebalan tubuh harus mampu untuk membedakan antara

antigen pada patogen yang selanjutnya disebut dengan *nonself-antigen* dengan antigen yang dimiliki sel-sel tubuh yang disebut dengan *self-antigen*. Jika terdeteksi sebagai *self-antigen*, maka antigen tersebut akan dibiarkan, tetapi jika terdeteksi sebagai *nonself-antigen*, maka sel T dan sel B akan bermutasi sampai menemukan kombinasi antibodi yang cocok untuk mengeliminasi antigen tersebut. Prinsip algoritma AIS bertujuan menemukan sebuah jadwal pekerjaan yang diproses pada mesin tertentu sedemikian, sehingga *makespan* dari sistem secara keseluruhan dapat diminimalkan. Algoritma AIS dalam skripsi ini menggunakan dua teori imunologi, yaitu algoritma seleksi positif dan algoritma seleksi klonal.

Berikut proses terjadinya kekebalan tubuh manusia.



Gambar 2.4 Proses terjadinya kekebalan tubuh mausia

Menurut Bondal (2008), untuk memudahkan memahami algoritma ini berikut diberikan analogi pada sistem kekebalan tubuh dan AIS.

a. Antigen

Antigen (*nonself-antigen*) adalah benda asing yang harus dieliminasi karena dapat mengganggu kestabilan tubuh. Sel tubuh yang dapat mengenali dan kemudian

menghancurkan antigen disebut antibodi. Dalam AIS antigen merupakan representasi masalah yang akan dicari solusi optimalnya.

b. Antibodi

Dalam teori sistem kekebalan tubuh, antibodi mengacu kepada sel-sel dalam tubuh manusia (sel T dan sel B) yang melawan infeksi yang menyerang. Dalam terminologi AIS, antibodi mengacu pada sekumpulan solusi potensial yang dihasilkan algoritma. Solusi yang baik adalah antibodi yang dapat mengenali dan mengeliminasi antigen. Untuk menghasilkan antibodi diperlukan bantuan perpustakaan antibodi.

c. Perpustakaan antibodi

Perpustakaan antibodi berisi sekumpulan kombinasi antibodi yang mampu mengeliminasi antigen. Sel T dan sel B dikombinasikan untuk membentuk struktur yang lebih kompleks, sehingga dapat mengenali antigen. Ketika infeksi menyerang, sistem kekebalan tubuh akan mencoba untuk menemukan kombinasi yang cocok untuk mengatasi serangan tersebut. Dalam AIS, perpustakaan antibodi berisi sekumpulan jadwal parsial yang siap disatukan, sehingga menjadi jadwal yang utuh. Beberapa hal yang mendukung terbentuknya perpustakaan antibodi dan membantu membangun antibodi, antara lain.

1) Komponen

Setiap perpustakaan antibodi terdapat kelompok komponen, komponen ini terdiri dari sejumlah gen tertentu. Suatu antibodi (solusi) terdiri dari kombinasi komponen pada setiap perpustakaan yang berbeda. Pada gambar 2.4 komponen horizontal adalah *string* angka untuk setiap perpustakaan dan dinotasikan dengan label *K1, K2, K3* dan *K4*.

2) Gen

Gen adalah bagian terkecil pada sebuah antibodi. Sekelompok gen menyusun sebuah komponen dan sekelompok komponen menyusun sebuah antibodi. Gen secara acak diberi nomor sebagai representasi nomor pekerjaan yang akan dilakukan.

d. Mutasi

Mutasi pada ilmu biologi mengacu kepada perubahan struktur DNA. Sedangkan pada sistem kekebalan tubuh manusia, mutasi dilakukan ketika tubuh mendapatkan serangan virus yang menyebabkan sel T dan sel B tidak dapat secara langsung menghancurkan virus tersebut, sehingga sel harus bermutasi untuk melakukan serangan balasan. Pada AIS, mutasi berarti menukarkan nilai 2 gen dalam rangka memperbaiki solusi atau disebut permutasi.

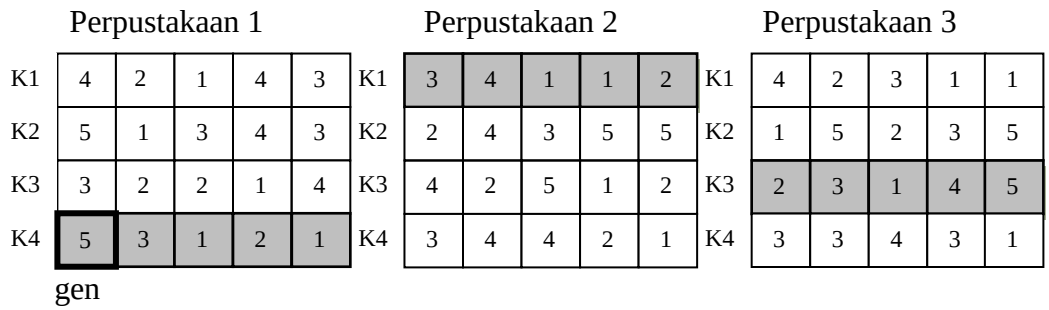
e. Prinsip seleksi positif

Prinsip seleksi positif dalam sistem imun adalah di saat antibodi mengenali *self-antigen* atau *nonself-antigen.* Saat antibodi mengenalinya sebagai *nonself-antigen*, maka antigen tersebut akan dieliminasi dari tubuh. Sedangkan pada AIS, proses seleksi diterapkan untuk menyaring solusi. Solusi yang disimpan dan akan digunakan di masa mendatang adalah solusi yang memenuhi kendala atau kriteria tertentu. Dalam hal ini, *makespan* dari *running* pertama program menjadi kriteria AIS, jika *makespan* selanjutnya lebih buruk, maka *makespan* tersebut tidak akan dipakai.

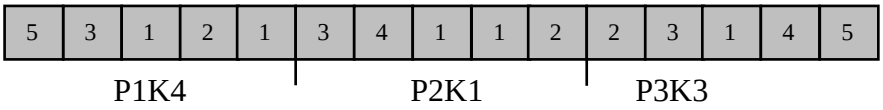
f. Prinsip seleksi klonal

Seleksi klonal pada sistem kekebalan tubuh adalah proses dimana sel B dan sel T bersama-sama melawan berbagai macam infeksi (antigen) yang menyerang tubuh manusia. Sel-sel tersebut dikombinasikan dan memperbanyak diri untuk dapat mengatasi infeksi atau virus. Prinsip seleksi klonal pada AIS adalah jika terdapat solusi yang mampu mendekati optimal, maka solusi tersebut akan dikloning dan diperbanyak dalam rangka untuk memperbaikinya.

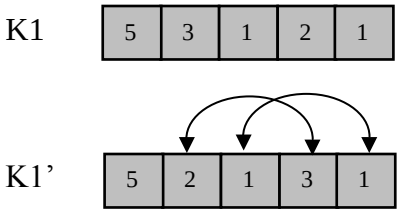
Berikut ilustrasi dari beberapa komponen penting dari AIS, yaitu perpustakaan antibodi dan gen pada Gambar 2.5, antibodi pada Gambar 2.6, dan proses mutasi pada Gambar 2.7.



Gambar 2.5 Perpustakaan antibodi untuk 3 pekerjaan pada 5 mesin



Gambar 2.6 Proses membangun antibodi dari perpustakaan antibodi



Gambar 2.7 Mutasi gen pada sebuah komponen

Pada Gambar 2.5, menurut Bondal (2008) jadwal acak dibangun dengan menggabungkan bagian-bagian dari perpustakaan antibodi (komponen) ke dalam satu antibodi lengkap seperti pada Gambar 2.6. Sejumlah antibodi pertama dievaluasi untuk menentukan *makespan* dari sistem dan kemudian bermutasi dalam rangka mencoba untuk memperbaiki agar didapatkan hasil yang paling optimal seperti pada Gambar 2.7.

Cara kerja algoritma AIS dapat dijelaskan sebagai berikut.

a. Menentukan parameter awal antibodi.

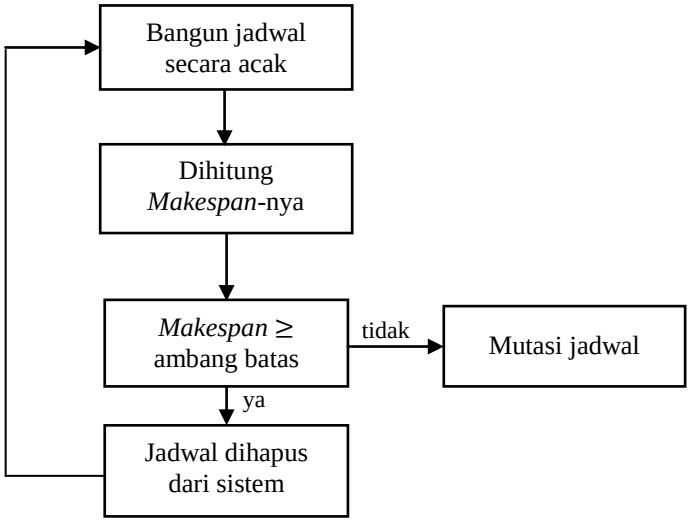
1. Jumlah perpustakaan antibodi, panjang antibodi dihitung dari hasil kali mesin dan pekerjaan.
2. Jumlah komponen perpustakaan, komponen adalah banyak baris pada setiap perpustakaan antibodi.

3. Jumlah gen pada komponen, banyak kolom pada perpustakaan antibodi dimana jumlahnya merupakan faktor dari perkalian antara pekerjaan dan mesin.

b. Membangun perpustakaan antibodi, perpustakaan antibodi ini berisi jadwal parsial. Terutama terdiri dari *string* (atau sebagian) dari solusi, ketika sejumlah tertentu dari komponen bergabung bersama-sama, mereka akan membentuk sebuah solusi lengkap. Masing-masing perpustakaan antibodi telah menetapkan sejumlah komponen seperti pada gambar 2.5.
c. Membangun antibodi seperti pada Gambar 2.6 dengan memilih satu komponen secara acak dari setiap perpustakaan. Prosedur untuk membangun antibodi adalah sebagai berikut.
   1. Memilih perpustakaan secara acak.
   2. Memilih secara acak komponen pada perpustakaan.
   3. Menyalin gen dari komponen ke antibodi.
   4. Memilih secara acak perpustakaan lain (perpustakaan yang telah dipilih sekali tidak dapat dipilih lagi).
   5. Memilih secara acak komponen dan menambahkan gen dari komponen ke dalam jadwal, dimulai dari posisi setelah akhir urutan gen sebelumnya.
   6. Kembali memilih komponen pada perpustakaan dan melanjutkan proses sampai semua antibodi terisi.
d. Mutasi antibodi, mutasi dilakukan pada solusi-solusi yang telah dibangun sebelumnya, yaitu solusi yang telah memenuhi kriteria awal. Hal ini bertujuan untuk mendapatkan solusi yang baik bahkan lebih baik dengan *makespan* paling minimal seperti pada Gambar 2.7.
e. Menguraikan antibodi yang telah terbentuk dan membangun jadwal dengan menugaskan pekerjaan ke mesin dengan urutan prioritas tertentu, kemudian dihitung *makespan*-nya dengan menggunakan diagram *gantt*.
f. Menampilkan antibodi dengan *makespan* paling optimal, hal ini tidak mutlak hanya menghasilkan jadwal tunggal, dimungkinkan akan terbentuk jadwal lebih dari satu yang juga memiliki *makespan* paling optimal.

2.2.1 Prinsip Seleksi Positif

Menurut Seiden & Celada (2002), ciri utama dari prinsip seleksi positif adalah membedakan antara solusi baik dan solusi buruk dimana solusi baik akan dipertahankan sedangkan solusi buruk akan dibuang dari sistem. Sedangkan pada sistem kekebalan tubuh manusia, seleksi positif berguna saat antibodi mengenali antigen terutama *nonself-antigen* karena antigen tersebut akan dieliminasi dari tubuh. Untuk mengenali antigen, sebuah patogen membutuhkan tingkat *affinitty* yang tinggi. Menurut Sasongko (2007), kuat atau tidaknya sebuah antibodi dalam mengenali antigen, dinamakan *affinitty* (afinitas). Semakin besar nilai *affinitty* menunjukkan semakin kuatnya ikatan antara antigen dan antibodi, demikian pula sebaliknya jika nilai *affinity* kecil menunjukkan ikatan antibodi dengan antigen yang lemah. Sel yang memiliki kemampuan untuk mengenali antigen dengan baik dipindahkan untuk dipertahankan, sehingga dapat digunakan lebih lanjut untuk menghancurkan antigen. Di sisi lain, antibodi yang gagal untuk mengenali antigen tidak diperlukan lagi dan tidak akan digunakan untuk melawan infeksi oleh sistem kekebalan tubuh.



Gambar 2.8 Prinsip seleksi positif pada AIS

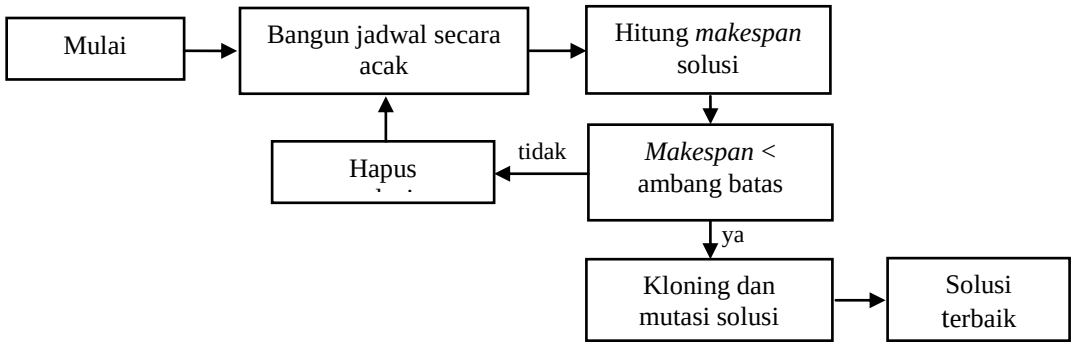
Gambar 2.8 mengilustrasikan bekerjanya prinsip seleksi positif dalam AIS. Misalkan sebuah jadwal acak terbentuk, maka pada langkah selanjutnya jadwal

tersebut akan dihitung *makespan*-nya. Jika *makespan* tersebut lebih dari ambang batas (dalam hal ini ambang batas adalah *makespan* optimal pada *running* program pertama) maka jadwal akan dihapus dari sistem, jika tidak maka jadwal akan dimutasi dalam rangka untuk memperbaikinya sehingga akan didapatkan *makespan* yang diharapkan lebih optimal dari jadwal sebelumnya.

Algoritma seleksi positif diterapkan untuk menyaring tujuan dimana hanya solusi yang memenuhi kendala tertentu atau kriteria yang perlu disimpan dan akan digunakan di masa mendatang. Tujuan diterapkannya algoritma seleksi positif adalah untuk mempersempit ruang cari solusi layak pakai untuk masalah penjadwalan *job shop*, sehingga diharapkan mengurangi waktu komputasi.

2.2.2 Prinsip Seleksi Klonal

Prinsip seleksi klonal didasarkan pada pengamatan yang dilakukan pada sel B dalam sistem kekebalan tubuh manusia. Sel B dan sel T bersama-sama melawan berbagai macam infeksi (antigen) yang menyerang tubuh manusia. Sel-sel itu dikombinasikan untuk dapat mengatasi infeksi atau virus. Sel-sel akan berkembang biak dan diproduksi secara massal untuk menghancurkan semua jejak infeksi di seluruh tubuh. Pada dasarnya, sel B mengkloning diri menjadi sebanyak yang diperlukan untuk melawan infeksi dan hanya kloning sel yang cukup baik yang akan dipertahankan. Algoritma ini bertujuan untuk mengenali solusi yang baik dan hanya mengkopi solusi yang layak sebanyak yang diperlukan untuk menyelesaikan masalah.



Gambar 2.9 Prinsip seleksi klonal pada AIS

Prinsip seleksi klonal pada AIS di atas dapat dijelaskan sebagai berikut, prinsip seleksi klonal pada AIS bekerja jika terdapat solusi yang mampu mendekati optimal, maka solusi tersebut akan dikloning. Prinsip ini berlaku sama untuk masalah penjadwalan *job shop*. Setelah solusi awal secara acak dihasilkan kemudian disaring (kurang dari *makespan* awal), solusi layak yang didapatkan akan dikloning dengan jumlah tertentu dan bermutasi lebih lanjut dalam rangka untuk memperbaikinya dan akhirnya akan didapatkan solusi terbaik dengan *makespan* paling minimum.

BAB 3. METODE PENELITIAN

3.1 Data

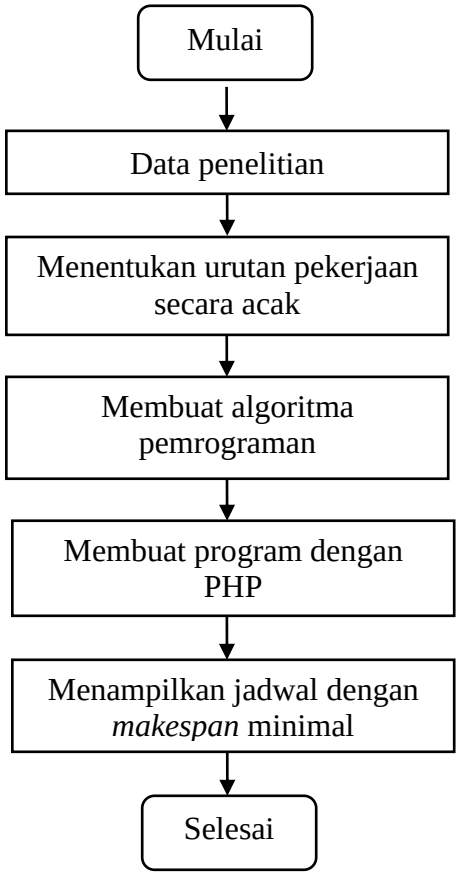
Data yang digunakan dalam skripsi ini berupa data sekunder hasil dari pengamatan oleh Junida pada skripsinya pada tahun 2009 pada pabrik *spring bed*. Pabrik ini memproduksi 4 tipe *spring bed* yaitu tipe *Platinum*, *Silver*, *Golden*, dan *Bedline* (Junida, 2009). Perbedaan empat jenis *spring bed* didasarkan pada kain *quilting*, rakitan per bulat dan busa yang digunakan. Program yang akan dibuat menggunakan bahasa pemrograman PHP.

3.2 Langkah-Langkah Penyelesaian

Langkah-langkah penyelesaian masalah penjadwalan *job shop* dengan kasus perakitan pada industri *spring bed* adalah sebagai berikut.

1. Mengolah data yang diperoleh menjadi data urutan mesin dan waktu proses.
2. Menentukan urutan setiap pekerjaan pada mesin secara acak.
3. Membuat algoritma pemrograman dari masalah penjadwalan *job shop* menggunakan algoritma AIS.
4. Membuat program berdasarkan algoritma pada langkah ke-3 menggunakan bahasa pemrograman PHP.
5. Menampilkan jadwal dengan *makespan* paling optimal dengan menggunakan program yang dibuat pada langkah-4.

Berikut skema untuk menyelesaikan permasalahan penjadwalan *job shop* dengan AIS.



Gambar 3.1 Skema langkah-langkah penyelesaian

BAB 4. HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas penyelesaian masalah penjadwalan *job shop* pada pabrik *spring bed* PT. Cahaya Kawi Ultra Polyintraco dengan algoritma *artificial immune system* (AIS) menggunakan program yang telah dibuat dengan bantuan bahasa pemrograman PHP.

4.1 Hasil

4.1.1 Identifikasi Mesin Produksi

Data dalam penelitian ini berupa data pengamatan mesin pada pembuatan 4 tipe *spring bed*, yaitu tipe *Platinum, Golden, Silver,* dan *Bigline*. Data tersebut berupa data nomor mesin dan waktu proses pembuatan tiap komponen di masing-masing mesin yang dihitung dengan melakukan pengukuran (berdasarkan *stopwatch*). Terdapat 9 jenis mesin yang digunakan pada proses pembuatan *spring bed*. Mesin-mesin tersebut direpresentasikan dengan nomor urut, sedangkan waktu proses mesin direpresentasikan dengan angka (dalam menit).

Spring bed akan diproses pada 9 jenis mesin yang kemudian mengelompokkan kesembilan jenis mesin tersebut berdasarkan mesin yang dibutuhkan untuk membuat komponen-komponen *spring bed*. Misal, untuk pembuatan matras diperlukan 7 jenis mesin, ketujuh mesin tersebut dinotasikan dengan simbol pekerjaan baru. Sama halnya dengan komponen yang lain, pembuatan sandaran dan *divan spring bed* yang masing-masing terdiri dari 5 jenis mesin juga akan dinotasikan dengan simbol pekerjaan baru, ditambah proses pembuatan kaki *spring bed*. Jadi, proses keseluruhan yang terdiri dari 4 tipe *spring bed* yang akan dibuat, dimana setiap *spring bed* terdiri atas 4 komponen penting, yaitu matras, sandaran, divan, dan kaki *spring bed*. Keempat komponen tersebut dapat dikerjakan tanpa memperhatikan urutan prosesnya.

Proses detailnya adalah sebagai berikut, pembuatan matras terdiri dari proses perakitan pegas (M_1), perakitan kawat (M_2), pembuatan rangka kayu (M_7), pembuatan *hardpad* (M_5), pemotongan dan pemasangan busa (M_3), *finishing* komponen rangka (M_8), dan penjahitan kain (M_6) dimana proses pembuatannya tidak dapat diubah urutannya, maka M_1, M_2, M_3, M_5, M_6, M_7 dan M_8 pada proses pembuatan matras akan disimpan pada memori M_1^* (penotasian baru) yang dinamakan proses pembuatan matras *spring bed*. Hal yang sama untuk pembuatan sandaran, untuk pembuatan sandaran terdiri dari proses pembuatan rangka kayu (M_7), pemotongan dan pemasangan busa (M_3), pembuatan *hardpad* (M_5), *finishing* komponen rangka (M_8), dan penjahitan kain (M_6) disimpan pada memori M_2 atau notasi M_7, M_3, M_5, M_6, dan M_8 menjadi notasi baru M_2^* yang dinamakan proses pembuatan sandaran *spring bed*. Proses pembuatan *divan* yang terdiri atas pembuatan rangka kayu (M_7), pembuatan *hardpad* (M_5), pemotongan dan pemasangan busa (M_3), *finishing* komponen rangka (M_8), dan penjahitan kain (M_6) disimpan pada memori M_3^* atau notasi M_7, M_3, M_5, M_6, dan M_8 menjadi notasi M_3^*. Tabel pereduksian mesin pada proses pembuatan matras, sandaran, dan *divan spring bed* adalah sebagai berikut.

Tabel 4.1 Reduksi 9 mesin menjadi 5 mesin dengan penotasian baru

No.	Komponen *Spring bed*	Notasi Mesin Lama	Tugas Mesin	Notasi Mesin Baru	Tugas Mesin
1.	Matras	M_1	Perakitan pegas	M_1^*	Pembuatan matras *spring bed*
		M_2	Perakitan kawat		
		M_7	Pembuatan rangka kayu		
		M_5	Pembuatan *hardpad*		
		M_3	Pemotongan dan pemasangan busa		
		M_8	*Finishing* komponen rangka		
		M_6	Penjahitan kain		
2.	Sandaran	M_7	Pembuatan rangka kayu	M_2^*	Pembuatan sandaran *spring bed*
		M_5	Pembuatan *hardpad*		
		M_3	Pemotongan dan pemasangan busa		
		M_8	*Finishing* komponen rangka		
		M_6	Penjahitan kain		

No.	Komponen *Spring bed*	Notasi Mesin Lama	Tugas Mesin	Notasi Mesin Baru	Tugas Mesin
3.	*Divan*	M_7	Pembuatan rangka kayu	M_3*	Pembuatan *divan spring bed*
		M_5	Pembuatan *hardpad*		
		M_3	Pemotongan dan pemasangan busa		
		M_8	*Finishing* komponen rangka		
		M_6	Penjahitan kain		
4.	Kaki	M_4	Pembuatan kaki *spring bed*	M_4*	Pembuatan kaki *spring bed*
5.	Perakitan	M_9	Perakitan *spring bed*	M_5*	Perakitan *spring bed*

Dengan penotasian baru seperti pada Tabel 4.1, maka proses pada M_1^*, M_2^*, M_3^*, M_4^*, dan M_5^* adalah proses yang saling asing karena proses pengerjaannya tidak saling bergantung satu sama lain, sehingga proses selanjutnya dapat diselesaikan dengan program yang telah dibuat. Berikut akan diberikan tabel waktu proses tiap mesin (sebelum direduksi) dan mesin dengan penotasian baru.

Tabel 4.2 Waktu proses mesin pada pembuatan 4 jenis *spring bed*

No.	Komponen *Spring bed*	Notasi Mesin Lama	Waktu (Menit) Tipe *Spring bed*				Notasi Mesin Baru	Waktu (Menit) Tipe *Spring bed*			
			1	2	3	4		1	2	3	4
1.	Matras	M_1	14	14	13	13	M_1^*	91	91	86	85
		M_2	15	15	15	14					
		M_7	18	18	16	16					
		M_5	3	3	2	2					
		M_3	6	6	5	5					
		M_8	15	15	15	15					
		M_6	20	20	20	18					
2.	Sandaran	M_7	10	10	10	10	M_2^*	48	48	45	44
		M_5	3	3	2	2					
		M_3	8	8	8	7					
		M_8	12	12	11	11					
		M_6	15	15	14	14					
3.	*Divan*	M_7	16	16	15	15					
		M_5	10	10	9	9					

No.	Komponen *Spring bed*	Notasi Mesin Lama	Waktu (Menit) Tipe *Spring bed*				Notasi Mesin Baru	Waktu (Menit) Tipe *Spring bed*			
			1	2	3	4		1	2	3	4
		M_3	8	8	8	7	M_3^*	66	66	63	61
		M_8	15	15	14	14					
		M_6	17	17	17	16					
4.	Kaki	M_4	8	8	8	8	M_4^*	8	8	8	8
5.	Perakitan	M_9	20	18	14	12	M_5^*	20	18	14	12

Tabel di atas menunjukkan bahwa proses pembuatan matras untuk 4 tipe *spring bed* berturut-turut yang dilakukan oleh 7 jenis mesin yaitu sebesar 91, 91, 86, dan 85 menit, waktu yang dibutuhkan untuk membuat sandaran yang dilakukan oleh 5 jenis mesin berturut-turut yaitu sebesar 48, 48, 45, dan 44 menit, dan waktu untuk membuat *divan* yang dilakukan oleh 5 jenis mesin berturut-turut yaitu sebesar 66, 66, 63, dan 61 menit. Pembuatan kaki *spring bed* berturut-turut membutuhkan waktu sebesar 8,8,8, dan 8 menit, serta perakitan komponen *spring bed* berturut-turut sebesar 20, 18, 14, dan 12 menit. Selengkapnya dapat dilihat pada Tabel 4.3.

Tabel 4.3 Data waktu proses mesin di setiap pekerjaan (dalam menit)

No. of jobs	*Job*	Nomor Mesin				
		M_1^*	M_2^*	M_3^*	M_4^*	M_5^*
1	***Platinum***	91	48	66	8	20
2	***Golden***	91	48	66	8	18
3	***Silver***	86	45	63	8	14
4	***Bigline***	85	44	61	8	12

4.1.2 Penyelesaian *Job Shop* dengan Program

a. *Input* Ukuran Masalah

Dalam skripsi ini, ukuran masalah penjadwalan *job shop* adalah 4×5 atau 4 pekerjaan akan diproses pada 5 mesin, sehingga jadwal yang terbentuk memiliki panjang 20 digit jadwal.

b. *Input* Data ke dalam *Form* atau *Import* dari *Excel*

Pada *form* ini, data urutan proses (*Opn.*), nomor mesin (*M*), dan waktu proses (*T*) dapat di-*input* langsung ke dalam *form* atau dapat juga di-*import* dari *excel*. *File* tersebut kemudian disimpan dalam satu *folder* beserta masalah lain yang memiliki ukuran berbeda-beda. Data lengkap tersebut seperti pada Tabel 4.5 di bawah ini.

Tabel 4.4 Urutan mesin secara acak dan waktu proses

Job	**Opn.1 (M,T)**	**Opn.2 (M,T)**	**Opn.3 (M,T)**	**Opn.4 (M,T)**	**Opn.5 (M,T)**
1	(3,66)	(1,91)	(4,8)	(2,48)	(5,20)
2	(2,48)	(3,66)	(1,91)	(4,8)	(5,18)
3	(4,8)	(1,86)	(3,63)	(2,45)	(5,14)
4	(4,8)	(3,61)	(1,85)	(2,44)	(5,12)

Data seperti pada Tabel 4.5 sebelumnya kemudian dimasukkan kedalam *excel* dan disimpan dengan nama *file* "4x4.xlsx", sehingga tampilan menjadi seperti pada Tabel 4.6.

Tabel 4.5 Urutan mesin acak dan waktu proses dalam *excel*

M1	T1	M2	T2	M3	T3	M4	T4	M5	T5
3	66	1	91	4	8	2	48	5	20
2	48	3	66	1	91	4	8	5	18
4	8	1	86	3	63	2	45	5	14
4	8	3	61	1	85	2	44	5	12

Tabel di atas menunjukkan 4 pekerjaan, yaitu pembuatan *spring bed* tipe *Platinum*, *Golden*, *Silver*, dan *Bigline* yang dilakukan pada mesin-mesin dengan urutan acak. Sehingga dapat dijelaskan sebagai berikut, pekerjaan 1 operasi pertama diproses pada mesin 3 dengan waktu sebesar 66 menit. Selanjutnya diproses berturut-turut pada mesin 1, mesin 4, mesin 2 dan terakhir mesin 5 dengan waktu berturut-turut sebesar 91 menit, 8 menit, 48 menit, dan 20 menit. Pekerjaan 2 operasi pertama diproses pada mesin 2 dengan waktu sebesar 48 menit. Selanjutnya diproses berturut-turut pada mesin 3, mesin 1, mesin 4 dan terakhir mesin 5 dengan waktu berturut-

turut sebesar 66 menit, 91 menit, 8 menit, dan 18 menit. Pekerjaan 3 operasi pertama diproses pada mesin 4 dengan waktu sebesar 8 menit. Selanjutnya diproses berturut-turut pada mesin 1, mesin 3, mesin 2 dan terakhir mesin 5 dengan waktu berturut-turut sebesar 61 menit, 85 menit, 44 menit, dan 14 menit. Pekerjaan 4 operasi pertama diproses pada mesin 4 dengan waktu sebesar 8 menit. Selanjutnya diproses berturut-turut pada mesin 3, mesin 1, mesin 2 dan terakhir mesin 5 dengan waktu berturut-turut sebesar 61 menit, 85 menit, 44 menit, dan 12 menit

c. Analisis Data

Berdasarkan algoritma pemrograman yang dibuat, selanjutnya dibuat suatu program yang digunakan untuk mencari jadwal yang memiliki *makespan* optimal pada permasalahan *job shop.*

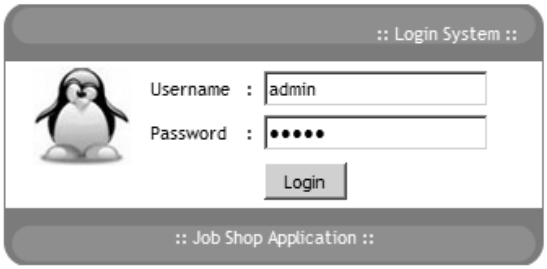
d. *Output* Program

Hasil akhir yang diinginkan adalah membangun jadwal dengan *makespan* minimal. *Output* yang lain berupa waktu komputasi dan jumlah iterasi. Tampilan awal program seperti pada Gambar 4.1 dan program ini dinamakan *Job Shop Application*.

4.1.3 Langkah-Langkah Menjalankan Program

Ada beberapa langkah yang harus dilakukan untuk menjalankan program ini antara lain sistem *login*, *input/import* data, *input* parameter dan *output*. Berikut dijelaskan langkah demi langkah menjalankan program *Job Shop Application*.

Tampilan awal dari program ini adalah sistem *login*, pengguna harus mengisi *form* "*username* dan *password"* .



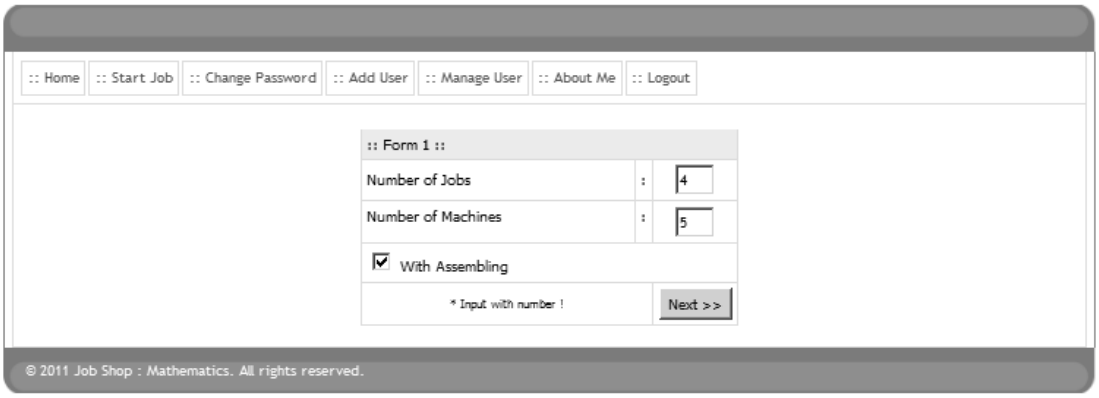
Gambar 4.1 Tampilan sistem *login*

Pada Gambar 4.1, penulis bertindak sebagai admin program *Job Shop Application*. Menu pada *form* dari tampilan di atas adalah:

a. *username*, yaitu *form* ini berisikan nama dari akun *user* pada *Job Shop Application*;
b. *password*, yaitu *form* kata sandi *user* yang digunakan untuk masuk kedalam *Job Shop Application*;
c. tombol *login*, yaitu untuk memulai *Job Shop Application* jika *input username* dan *password* sukses dieksekusi program.

Menu selanjutnya dari program ini adalah sebuah *form* yang digunakan untuk meng-*input* ukuran masalah yang diselesaikan. Program ini dibuat dengan memperhatikan tingkat kesulitan algoritma, penulis membatasi maksimal ukuran masalah yang dapat dieksekusi pada program ini adalah sampai 9×9 atau 9 pekerjaan pada 9 mesin.

Pada saat input pada *form* 1 ukurannya harus sama dengan data yang akan di-*import* nantinya. Untuk ke *form* berikutnya dapat meng-*klik* tombol *next*.



Gambar 4.2 Menu *start jobs* "*form* 1"

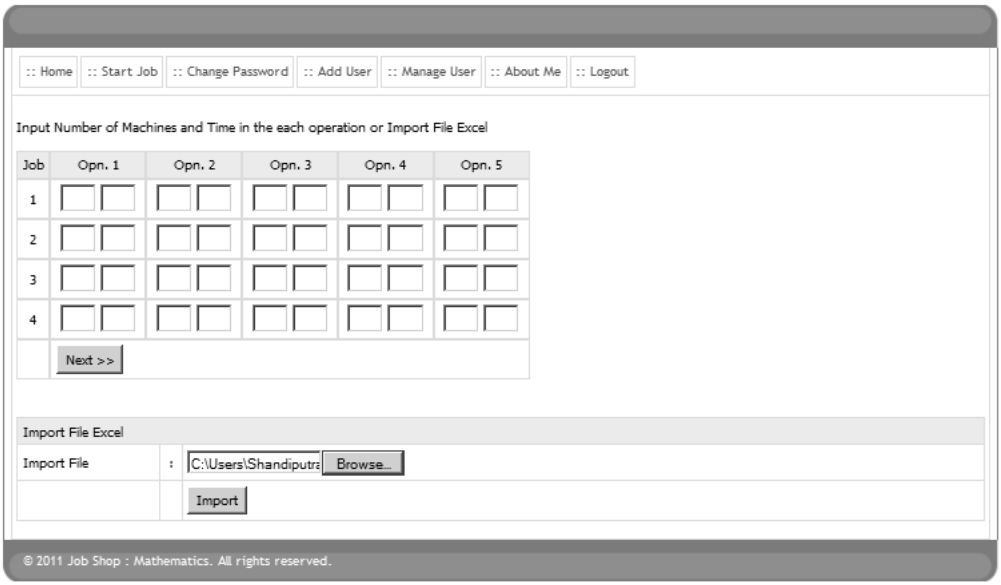
Menu pada *form* dari tampilan di atas adalah:

a. *number of jobs*, yaitu jumlah pekerjaan (*jobs*) yang akan diproses pada kasus ini penulis akan menyelesaikan 4 pekerjaan*;*
b. *number of machines*, yaitu jumlah mesin yang akan memproses pekerjaan pada kasus ini penulis akan menggunakan 5 jenis mesin;

c. tombol *next*, yaitu tombol untuk melanjutkan ke *form* berikutnya.

d. *check box*, yaitu sebuah fitur yang digunakan apabila *user* menggunakan proses perakitan pada proses produksinya.

Tampilan selanjutnya adalah sebuah *form* kosong untuk memasukan data urutan mesin dan waktu proses. Dalam kasus ini adalah 4 pekerjaan yang diproses pada 5 jenis mesin. Kotak pertama pada setiap kolom *opn.* berisi nomor urut mesin dan kotak kedua berisi waktu proses. Hal tersebut dapat dilakukan secara manual dan dapat juga di-*import* dari *excel* seperti pada Tabel 4.5. Fitur ini berkaitan dengan faktor kenyamanan pengguna karena pekerjaan yang membosankan untuk memasukkan data secara manual setiap kali program dimulai. Tahap ini akan selalu dilakukan setiap program dijalankan, program dijalankan beberapa kali untuk mendapatkan hasil yang optimal.



Gambar 4.3 Menu *start jobs* "*input* nomor mesin dan waktu"

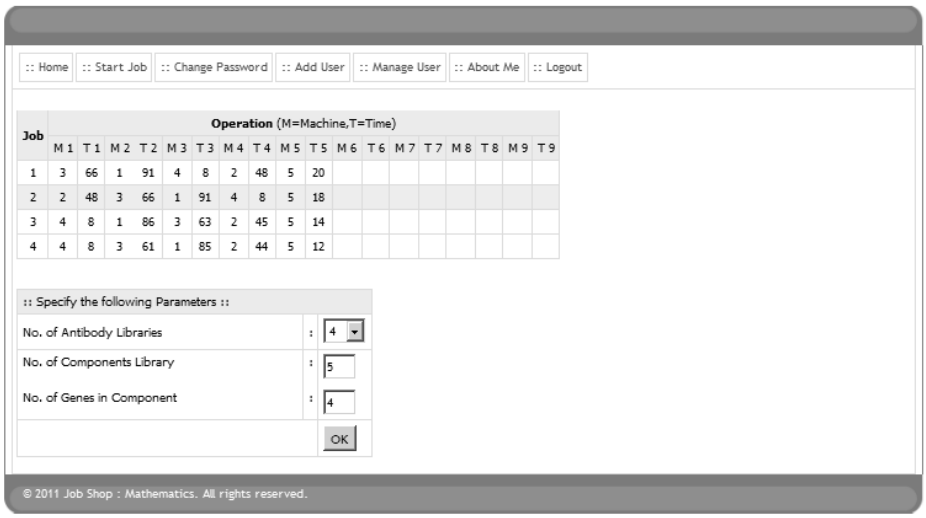
Menu pada *form* dari tampilan di atas adalah:

a. *form* data urutan mesin dan waktu proses, yaitu sebuah *form* yang digunakan untuk memasukkan data urutan mesin dan waktu proses;

b. *browse*, yaitu mencari data yang akan di-*import* ke *database*;

c. *import*, yaitu *import* data dari *excel* ke *database*, sehingga dapat digunakan kembali berkali-kali;
d. tombol *next*, yaitu melanjutkan ke *form* berikutnya.

Tampilan selanjutnya adalah sebuah *form* yang berisikan data hasil *import* pada *form* sebelumnya. *Form* ini digunakan untuk memeriksa apakah data-data yang telah dimasukkan pada program ini sudah benar. Berikutnya adalah kolom *no. of component library*, *no. of component library*, dan *no. of gen*s *in component*. Ketiga kolom tersebut merupakan parameter-parameter yang digunakan untuk membangun perpustakaan antibodi dimana setiap perpustakaan antibodi berisikan jadwal parsial yang siap disatukan, sehingga menjadi jadwal yang utuh. Dalam kasus ini panjang jadwal sama dengan 20 dimana 4 digit jadwal terakhir adalah proses perakitan komponen pada pekerjaan 1,2,3, dan 4. Kolom *no. of antibody library* dan *no. of gen*s *in component* adalah faktor dari 16. Angka 16 tersebut didapatkan dari sebuah jadwal utuh 20 digit jadwal dikurangi 4 digit jadwal (proses perakitan pekerjaan 1,2,3, dan 4). Misalkan dipilih *no. of antibody library* sama dengan 4, maka secara otomatis *no. of gen*s *in component* akan terisi angka 4 karena $4 \times 4 = 16$ dan seterusnya. Untuk *no. of component library* tidak memiliki batasan pengisian, *form* tersebut untuk menetukan jumlah baris pada setiap perpustakaan antibodi. *Form* tersebut seperti Gambar 4.4 berikut.

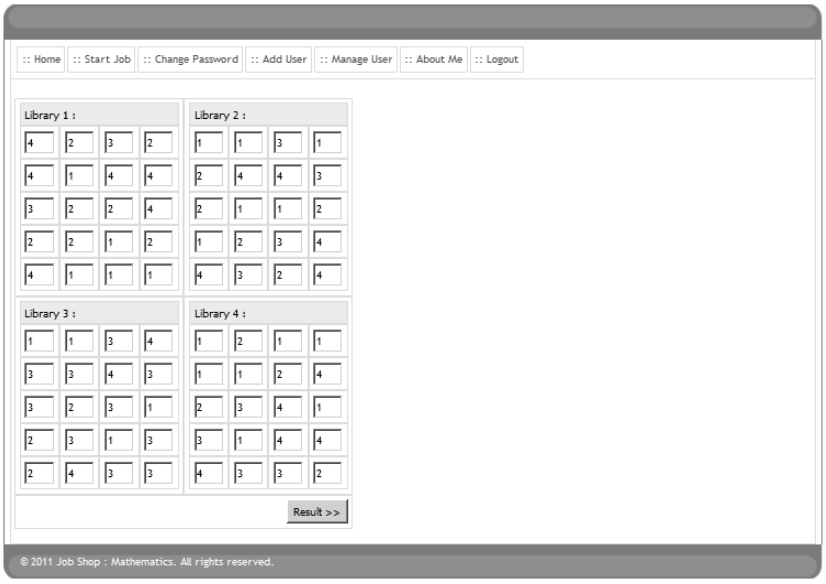


Gambar 4.4 Menu *start jobs* "data hasil *input* secara manual atau

Menu pada *form* dari tampilan di atas adalah:

a. *no. of antibody library*, yaitu menentukan jumlah perpustakaan antibodi dalam kasus ini dipilih adalah 4;
b. *no. of component library*, yaitu menentukan jumlah komponen (baris) dalam kasus ini dipilih 5;
c. *no. of gens in component*, yaitu menentukan jumlah gen (kolom) dalam kasus ini otomatis akan terpilih 4.
e. Tombol *Ok*, yaitu melanjutkan ke *form* berikutnya.

Form berikutnya adalah perpustakaan antibodi atau sekumpulan jadwal parsial secara acak yang berisi *job*. Tombol "*result*" untuk menampilkan *output* penyelesaian masalah *job shop* ini.



Gambar 4.5 Menu *start jobs* "perpustakaan antibodi"

Dalam rangka membangun solusi, salah satu komponen yang dipilih secara acak dari masing-masing perpustakaan yang berisi jadwal parsial. Sebuah perpustakaan yang telah digunakan dapat digunakan kembali. Ketika jadwal pertama kali dibangun dengan cara ini, solusi mungkin tidak sempurna yang idealnya harus memiliki jumlah yang sama pada tiap pekerjaan, yaitu pekerjaan 1, pekerjaan 2, pekerjaan 3, dan pekerjaan 4 adalah masing-masing sebanyak 4 digit jadwal, tetapi

dengan sifat acak dari proses seleksi akan ada probabilitas tinggi, yaitu terjadinya ketidakmerataan jumlah angka pada solusi. Masalah ini dipecahkan dengan strategi perbaikan, strategi perbaikannya adalah dengan menggantikan angka yang berlebih dengan angka-angka yang tidak memadai pada solusi.

Terakhir adalah *form output*, *output* ini berupa penyelesaian dari program. *Output* dari program adalah:

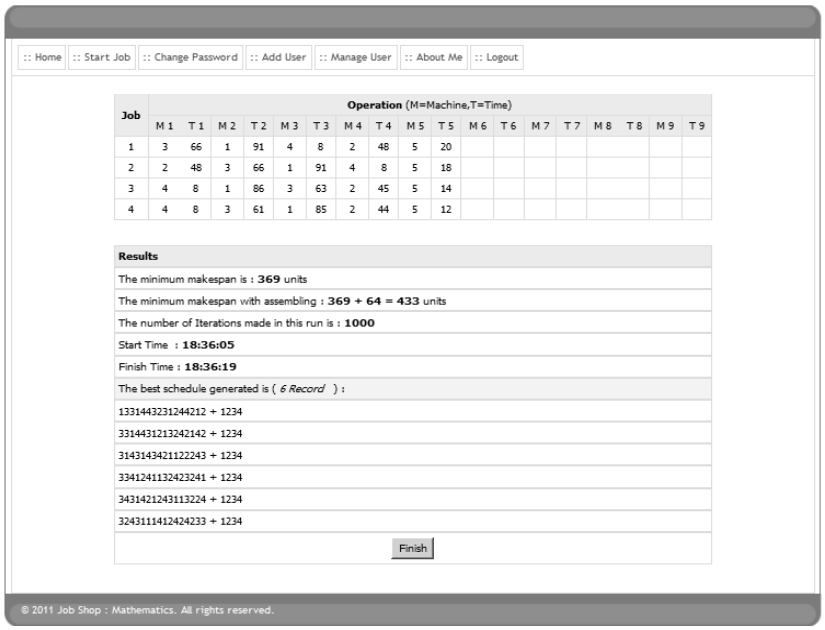
a. minimum makespan, yaitu menunjukkan makespan yang dibutuhkan untuk membuat komponen-komponen *spring bed* yang diproses pada mesin 1 hingga mesin 4;

b. minimum makespan with assembling, yaitu menunjukkan total *makespan* yang dibutuhkan untuk membuat dan merakit komponen pada 4 tipe *spring bed*;

c. number of iterations, yaitu menunjukkan banyak permutasi yang dilakukan program dalam hal ini sebanyak 1000 kali;

d. start time dan *finish time*, yaitu menunjukkan waktu komputasi yang dilakukan program untuk mencapai *makespan* optimal, yaitu 14 detik;

e. best schedule, yaitu menunjukkan semua jadwal yang terbentuk dengan *minimum makespan*.



Gambar 4.6 Salah satu *output* program *job shop application*

Hasil di atas merupakan salah satu dari percobaan yang dilakukan sebanyak 10 kali dengan 1000 permutasi tiap *running* program dan didapatkan *makespan optimal with assembling* sebesar 433 menit. Di dalam *database* program, terdapat *makespan* yang tidak mampu mencapai optimal yaitu jadwal dengan *makespan* lebih dari 369 menit (tanpa perakitan). Berikut adalah tabel yang berisi sebagian jadwal yang mampu dan jadwal yang tidak mampu mencapai *makespan* optimal pada *running* pertama, hasil selengkapnya dapat dilihat pada Lampiran A.

Tabel 4.6 Kumpulan jadwal acak pada *database*

No.	Isi	Hasil
1.	2414313124332421	443
2.	1413133312242244	733
3.	3143242143211423	369
4.	3314214231241423	458
5.	2414321133413422	427
6.	4243113432124231	540

No.	Isi	Hasil
7.	3224433211441321	617
8.	1223332144431412	491
9.	2141343321242341	640
10.	1432144322132134	480
11.	2134313224314241	491
12.	2424213334111342	612

Pada tabel di atas, terdapat jadwal dengan *makespan* yang tidak optimal, yaitu lebih dari 369 menit. Hasil diatas hanya sebagian dari *database* program *Job Shop Application*.

Setelah 10 kali *running* program, jadwal optimal mampu dicapai sebanyak 10 kali dengan rata-rata waktu komputasi 21,7 detik. Di bawah ini hasil dari 10 *running* program *Job Shop Application*

Tabel 4.7 Hasil percobaan 10 *running* program

No.	Jadwal yang Terbentuk	*Makespan* (menit)	*Makespan* + perakitan (menit)	Waktu (detik)	Jml. Iterasi	Ket.
1.	a) 3134414122122343 + 1234 b) 1233442111424332 + 1234 c) 2344344112321312 + 1234 d) 3312441314122342 + 1234 e) 2344134234121321 + 1234	369	433	30	1000	Optimal
2.	a) 3131241421423234 + 1234 b) 3312441314122342 + 1234	369	433	26	1000	Optimal

No.	Jadwal yang Terbentuk	*Makespan* (menit)	*Makespan* + perakitan (menit)	Waktu (detik)	Jml. Iterasi	Ket.
	c) 2344134234121321 + 1234 d) 3441342112324321 + 1234					
3.	a) 3134411324132242 + 1234 b) 3311442412212433 + 1234 c) 3341421343214122 + 1234 d) 2314341142313242 + 1234 e) 3412314142322143 + 1234 f) 3341142334214122 + 1234	369	433	26	1000	Optimal
4.	a) 2343114121434322 + 1234 b) 2134314314341222 + 1234 c) 3314143234212421 + 1234	369	433	29	1000	Optimal
5.	a) 3134412412212334 + 1234 b) 3423441124231231 + 1234 c) 1344231433212142 + 1234 d) 3241314413212324 + 1234 e) 2134431432213214 + 1234	369	433	29	1000	Optimal
6.	a) 3443411223413122 + 1234 b) 3314411213244223 + 1234 c) 1233443311242241 + 1234 d) 2331441114332224 + 1234	369	433	14	1000	Optimal
7.	a) 1233411434231242 + 1234 b) 3134412412212334 + 1234 c) 3243114241143322 + 1234 d) 3143241124212433 + 1234 e) 1332441124142332 + 1234 f) 3413141224312234 + 1234	369	433	13	1000	Optimal
8.	a) 2314432141143322 + 1234 b) 1334423132421142 + 1234 c) 1323144412132342 + 1234 d) 1343421331412242 + 1234 e) 2343141131242423 + 1234	369	433	14	1000	Optimal
9.	a) 1331443231244212 + 1234 b) 1233411434231242 + 1234 c) 3431243112143422 + 1234 d) 3342143321421421 + 1234 e) 1343124324212341 + 1234 f) 3142314411342232 + 1234	369	433	24	1000	Optimal
10.	a.) 3314414322311422 + 1234 b) 1331441234132422 + 1234 c) 3134413221412423 + 1234	369	433	14	1000	Optimal

4.2 Pembahasan

Algoritma AIS adalah salah satu algoritma yang dapat digunakan untuk menyelesaikan permasalahan optimasi. Dalam penelitian ini algoritma AIS diaplikasikan dengan bahasa pemrograman PHP untuk menyelesaikan masalah penjadwalan *job shop*. Alasan penulis menggunakan teknologi *web* (PHP), agar pengguna dapat mengakses program tanpa memperdulikan sistem operasi yang digunakannya setelah program ini terpasang di internet .

Semua jadwal optimal yang dapat dicapai program ini adalah sebagian kecil dari total jadwal yang mungkin terbentuk yaitu sebanyak $\frac{(n\times m)!}{m_1!m_2!\cdots m_n!}$ atau sebanyak $\frac{(4\times 5)!}{5!5!5!5!} = \frac{20!}{480}$ dimana *n* adalah pekerjaan dan *m* adalah jumlah mesin. Pada *running* pertama, program mampu menemukan 5 jadwal optimal dengan waktu komputasi sebesar 30 detik dari 1000 jadwal yang diperiksa *makespan*nya dan pada *running* berikutnya program juga mampu menemukan jadwal optimal dengan *makespan optimal with assembling* sebesar 433 menit.

Penulis hanya membatasi iterasi sebanyak 1000 kali permutasi karena setelah dilakukan beberapa percobaan ternyata 1000 permutasi adalah jumlah maksimal iterasi yang mampu dilakukan *hardware*. Artinya, dari seluruh jadwal yang terbentuk hanya 1000 jadwal yang akan dihitung nilai *makespan*nya pada tiap *running* program, oleh karena itu dibutuhkan beberapa kali *running* untuk memutuskan hasil akhir jadwal dengan *makespan* optimal. Hal ini diakui sebagai kelemahan dalam pembuatan program *Job Shop Application* karena jadwal valid yang mungkin terbentuk dengan jumlah iterasi yang mampu dilakukan program relatif sangat rendah sehingga keakuratan jadwal yang mampu dicapai program masih diragukan kevaliditasannya.

Dalam hal ini *hardware* yang digunakan memiliki spesifikasi sebagai berikut.

- *Prosessor* : Intel® Core™ i3 CPU @2.27GHz
- HDD : 350 GB

- *Installed Memory (RAM)* : 2,00 GB (1,86 GB *usable*)
- *Operating System* : Windows 7 Utimate.

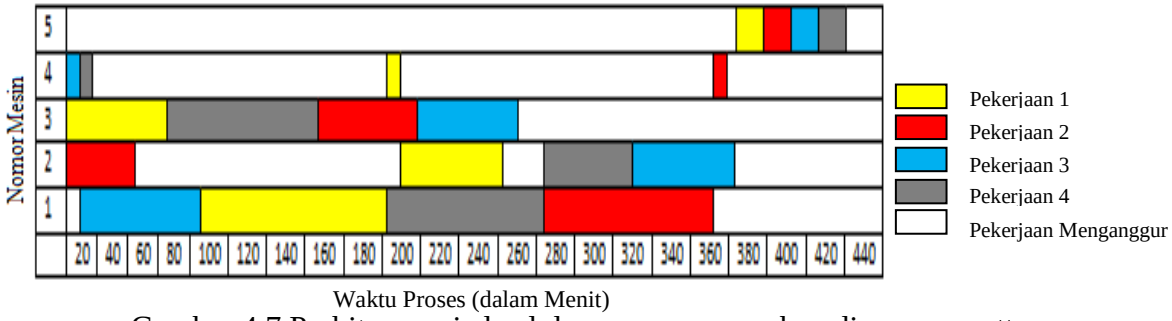
Untuk masalah yang lebih kompleks, diharapkan menggunakan spesifikasi *hardware* yang lebih tinggi dengan meng-*upgrade* RAM dan *prossesor*, hal ini dilakukan dalam rangka meningkatkan iterasi program agar ruang cari solusi lebih luas sehingga hasil optimal dapat diperoleh.

Secara ringkas, aspek-aspek yang dapat digunakan sebagai dasar untuk mengevaluasi algoritma AIS yaitu optimalitas. Hasil akhir program ini akan diperiksa apakah mampu mencapai optimal untuk masalah yang diberikan. Semua masalah yang diuji diharapkan menunjukkan solusi optimal. Jika solusi yang optimal tidak tercapai, maka yang terdekat dengan solusi optimal ditetapkan sebagai solusi optimal yang mampu dicapai oleh algoritma ini.

Pada program ini, prinsip seleksi positif diterapkan ketika *makespan* optimal ditemukan saat *running* pertama program *Job Shop Application,* yaitu sebesar 369 (tanpa perakitan) menit. *Makespan* tersebut menjadi kriteria awal untuk menyaring solusi-solusi selanjutnya, yaitu *makespan* yang kurang dari atau sama dengan kriteria awal nantinya akan dipertahankan. Pada *running* berikutnya, jika tidak ada jadwal dengan *makespan* yang lebih kecil dari kriteria awal, maka *makespan* terkecil pada *running* tersebut akan diambil sebagai solusi optimal. Setelah itu, prinsip seleksi klonal mulai diterapkan, yaitu dengan mutasi hasil yang didapatkan oleh prinsip seleksi positif dalam rangka memperbaikinya dan berharap akan mendapatkan *makespan* yang lebih baik. Program ini ternyata mampu mencapai solusi yang optimal untuk 10 kali *running* dan menghasilkan hasil akhir sama dengan 369 menit (tanpa perakitan) atau 433 menit (dengan perakitan). Hasil tersebut cukup untuk mewakili dalam mengambil kesimpulan akhir dari skripsi ini, yaitu 433 menit untuk *makespan* akhir proses perakitan 4 tipe *spring bed* menggunakan 5 mesin.

Berikut akan dijelaskan cara menghitung *makespan* jadwal dengan menggunakan diagram *gantt* yaitu diagram yang dapat membantu untuk menguraikan

jadwal, sehingga mudah untuk dipahami. Misal salah satu jadwal yang terbentuk oleh program *Job Shop Application* yaitu, "3-1-3-4-4-1-4-1-2-2-1-2-2-3-4-3 + 1-2-3-4". Pada jadwal tersebut didapatkan *makespan* sebesar 433 menit, iterasi sebanyak 1000 permutasi dengan waktu komputasi 30 detik. Berikut ditampilkan diagram *gantt* untuk jadwal tersebut.



Gambar 4.7 Perhitungan jadwal dengan menggunakan diagram *gantt*

Diagram *gantt* di atas digunakan untuk menghitung jadwal "3-1-3-4-4-1-4-1-2-2-1-2-2-3-4-3 + 1-2-3-4" dengan bantuan Tabel 4.4, sehingga dapat dijelaskan sebagai berikut. Angka pertama pada jadwal mengacu pada pekerjaan 3, pada operasi pertama pekerjaan 3 dilakukan di mesin 4 dengan waktu proses 8 menit. Pekerjaan selanjutnya adalah pekerjaan 1, pada operasi pertama dilakukan di mesin 3 dengan waktu proses 66 menit, lalu pekerjaan 3 pada operasi kedua dilakukan pada mesin 1 dengan waktu proses 91 menit. Selanjutnya pekerjaan 4, pada operasi pertama dilakukan di mesin 4 dengan waktu proses 8 menit, pekerjaan 4 yang kedua pada operasi yang kedua dilakukan pada mesin 3 dengan waktu proses 61 menit. Selanjutnya, proses perhitungan analog dengan perhitungan sebelumnya yaitu sampai pekerjaan 1, 2, 3, dan 4 pada operasi kelima (perakitan pekerjaan 1,2,3, dan 4) dan dilakukan sampai selesai 20 digit jadwal. Diagram *gantt* di atas digunakan untuk menghitung secara manual *makespan* suatu jadwal yang nantinya akan dicocokan dengan hasil yang diperoleh program yaitu *makespan* total sebesar 433 menit.

Ada pula cara lain untuk membaca jadwal yang telah terbentuk yaitu dengan mengisi Tabel 4.8 dengan urutan angka seperti pada jadwal. Berikut hasil dari urutan jadwal saat disajikan dengan tabel.

Tabel 4.8 Representasi jadwal dalam bentuk tabel

Job	***Opn.1 (M,T)***	***Opn.2 (M,T)***	***Opn.3 (M,T)***	***Opn.4 (M,T)***	***Opn.5 (M,T)***
1	2 (3,66)	6 (1,91)	8 (4,8)	11 (2,48)	17 (5,20)
2	9 (2,48)	10 (3,66)	12 (1,91)	13 (4,8)	18 (5,18)
3	1 (4,8)	3 (1,86)	14 (3,63)	16 (2,45)	19 (5,14)
4	4 (4,8)	5 (3,61)	7 (1,85)	15 (2,44)	20 (5,12)

Dengan menggunakan tabel di atas dapat terlihat urutan pekerjaan yang akan dilakukan selama 1 kali proses pembuatan 4 tipe *spring bed* dari nomor 1 (satu) hingga 20 (dua puluh).

Berdasarkan Tabel 4.8 dapat dibuat rincian urutan proses produksi sebagai berikut.

1). Pekerjaan 3 (operasi ke-1/ M_4^* [Pembuatan kaki *spring bed*]).
2). Pekerjaan 1 (operasi ke-1/ M_3^* [Pembuatan *divan spring bed*]).
3). Pekerjaan 3 (operasi ke-2/ M_1^* [Pembuatan matras *spring bed*]).
4). Pekerjaan 4 (operasi ke-1/ M_4^* [Pembuatan kaki *spring bed*]).
5). Pekerjaan 4 (operasi ke-2/ M_3^* [Pembuatan *divan spring bed*]).
6). Pekerjaan 1 (operasi ke-2/ M_1^* [Pembuatan matras *spring bed*]).
7). Pekerjaan 4 (operasi ke-3/ M_1^* [Pembuatan matras *spring bed*]).
8). Pekerjaan 1 (operasi ke-3/ M_4^* [Pembuatan kaki *spring bed*]).
9). Pekerjaan 2 (operasi ke-1/ M_2^* [Pembuatan sandaran *spring bed*]).
10). Pekerjaan 2 (operasi ke-2/ M_3^* [Pembuatan *divan spring bed*]).
11). Pekerjaan 1 (operasi ke-4/ M_2^* [Pembuatan sandaran *spring bed*]).
12). Pekerjaan 2 (operasi ke-3/ M_1^* [Pembuatan matras *spring bed*]).
13). Pekerjaan 2 (operasi ke-4/ M_4^* [Pembuatan kaki *spring bed*]).
14). Pekerjaan 3 (operasi ke-3/ M_3^* [Pembuatan *divan spring bed*]).
15). Pekerjaan 4 (operasi ke-4/ M_2^* [Pembuatan sandaran *spring bed*]).

16). Pekerjaan 3 (operasi ke-4/ M_2^* [Pembuatan sandaran *spring bed*]).

17). Pekerjaan 1 (operasi ke-5/ M_5^* [Perakitan *spring bed*]).

18). Pekerjaan 2 (operasi ke-5/ M_5^* [Perakitan *spring bed*]).

19). Pekerjaan 3 (operasi ke-5/ M_5^* [Perakitan *spring bed*]).

20). Pekerjaan 4 (operasi ke-5/ M_5^* [Perakitan *spring bed*]).

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan yang diuraikan pada bab 4, dapat diambil kesimpulan bahwa:

1. untuk memproduksi 4 jenis *spring bed*, yaitu *Platinum*, *Golden*, *Silver*, dan *Bigline* yang diproses melalui 5 mesin (dengan penotasian baru) dibutuhkan waktu 433 menit atau 7 jam 13 menit;
2. jadwal optimal yang terbentuk setelah 10 kali *running* program *Job Shop Application* tidak tunggal sehingga dapat memberikan alternatif jadwal lainnya bagi perusahaan.

5.2 Saran

Algoritma AIS dapat digunakan untuk menyelesaikan permasalahan optimasi dan masih terbuka bagi peneliti lain untuk mengaplikasikan algoritma AIS pada permasalahan optimasi lainnya misalnya pada masalah *flow shop* ataupun *travelling saleman problem* (TSP). Program *Job Shop Application* dengan menggunakan bahasa pemrograman PHP mampu menyelesaikan permasalahan penjadwalan *job shop* dan memiliki banyak keunggulan dibanding bahasa pemrograman yang lain. Serta, terbuka bagi peneliti lain untuk mengembangkan program ini karena masih terdapat kelemahan pada proses iterasi, yaitu iterasi yang mampu dilakukan program masih relatif kecil antara 800-1000 iterasi setiap *running program* yang mengakibatkan solusi yang mampu dicapai diragukan kevaliditasannya karena solusi optimal yang mampu dicapai program memiliki rasio yang rendah terhadap semua kemungkinan jadwal yang dapat terbentuk.

DAFTAR PUSTAKA

Aicklein, U. & Dasgupta, D. 2005. *Artificial Immune System*. http:/eprints.nottingham.ac.uk/621/1/03intros_ais_tutorial.pdf [3 april 2011].

Agam, H., Fariza, & A., Prasetyaningrum, I. Tanpa Tahun. *Penjadwalan Metode Simulated Annealing Untuk Penjadwalan Job Shop pada Mesin Pabrik*. http:/*repo.eepis-its.edu/586/1/1238.pdf* [24 Agustus 2011].

Baker, K. 1974. *Introduction to Sequencing and Scheduling*. Jhon Wiley and Sons, Inc.

Bondal, A. A. 2008. *Artificial Immune Systems Applied to Job Shop Scheduling*. Fakultas Mesin dan Teknologi College Russ, Universitas Ohio:Ohio.

Coello, Carlos A. C., Rivera, D. C., & Cortes, N. C. Tanpa Tahun. *Use of an Artificial Immune System for Job Shop Scheduling*. http:/arnetminer.org/viewpub.do?pid=273336 [4 april 2011].

Conway, R. W., Maxwell, W. L., & Miller, L. W. 1967. *Theory of Scheduling*. Addison-Wesley, Reading, MA.

Haibin, T. 2009. *Comparison of Synchronized Flow with Classical Flow in Multi-Stage Production Systems*. Universitas Ohio:Ohio.

Gantt, H. L. 1916. *Work, Wages, and Profit*. Second Edition, Engineering Megazine Co., New York.

Junida, W. 2009. *Penentuan Jumlah Produksi Optimal untuk Memaksimumkan Laba dengan Menggunakan Metode Integer Programing di PT. Cahaya Kawi Ultra Polyintraco*. Departemen Teknik Industri, Universitas Sumatra Utara:Medan.

Morton, T. E. & Pentico, D. 1993. *Heuristic Scheduling Systems with Applications to Production and Project Management* . Wiley:New York.

Timmis, J., Knight, T., de Castro, L.N., & Hart, E. 2002. *An Overview of Artificial Immune Systems.* http://kar.kent.ac.uk/14054/1. haspreview Thumbnail Version/ An_Overview_of_Artificial_Immune_Systems.pdf [4 maret 2011].

Tirta, I Made. 2004. *Pengantar Statistika Matematika*. Fakutas Matematika dan Ilmu Pengetahuan Alam, Universitas Jember:Jember.

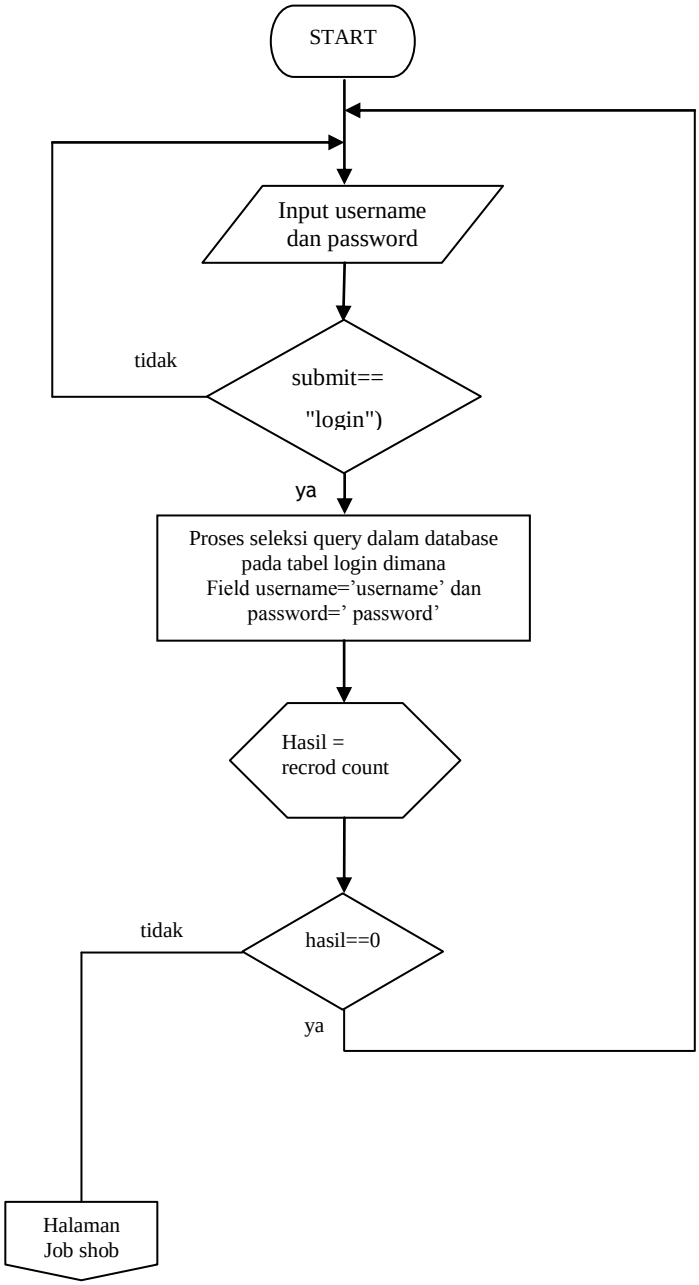
Sasongko, P. B. Tanpa Tahun. *Cloning-Based Algorithm dan Aplikasinya dalam Travelling Salesperson Problem*. http://www.scribd.com/doc/40692839/ MakalahIF2153-0708-096 [8 februari 2011].

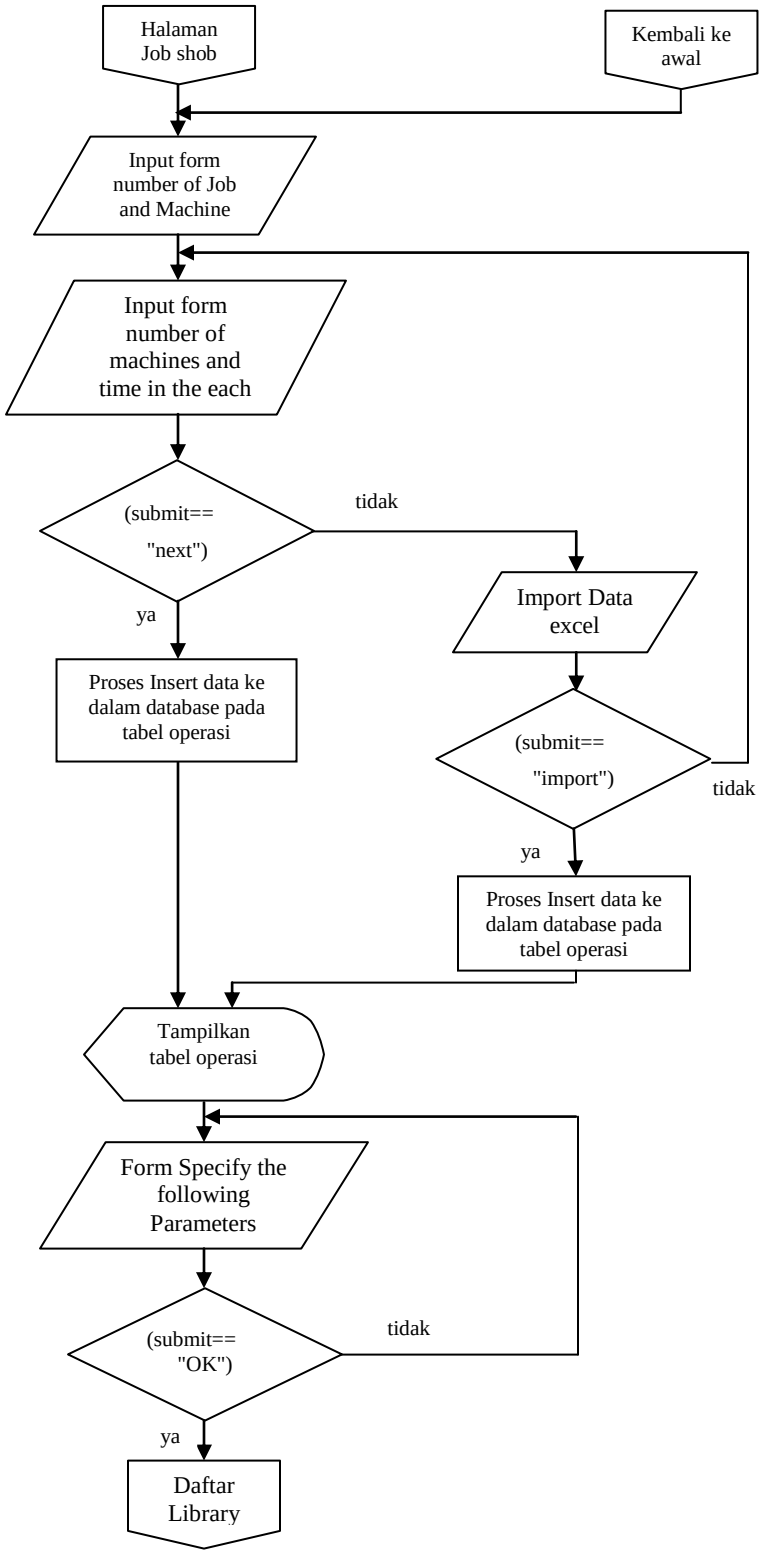
Seiden, P. E. & Celada, F. 1992. *A Model for Simulating Cognate Recognition and Response in the Immune System*. Jurnal Teori Biologi 158, pp. 329 – 357.

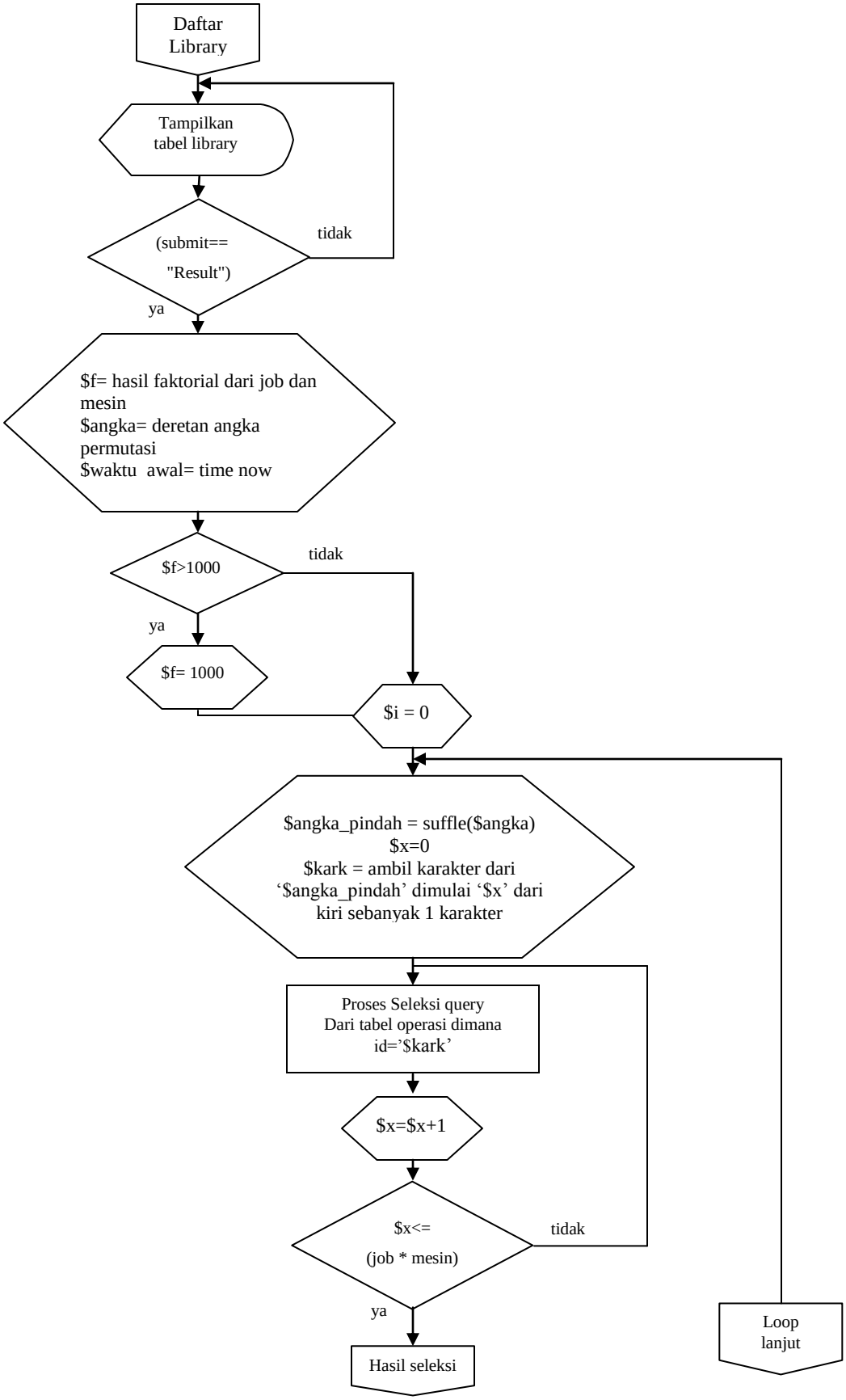
A. Sebagian Database Jadwal pada Program dari 1000 Permutasi

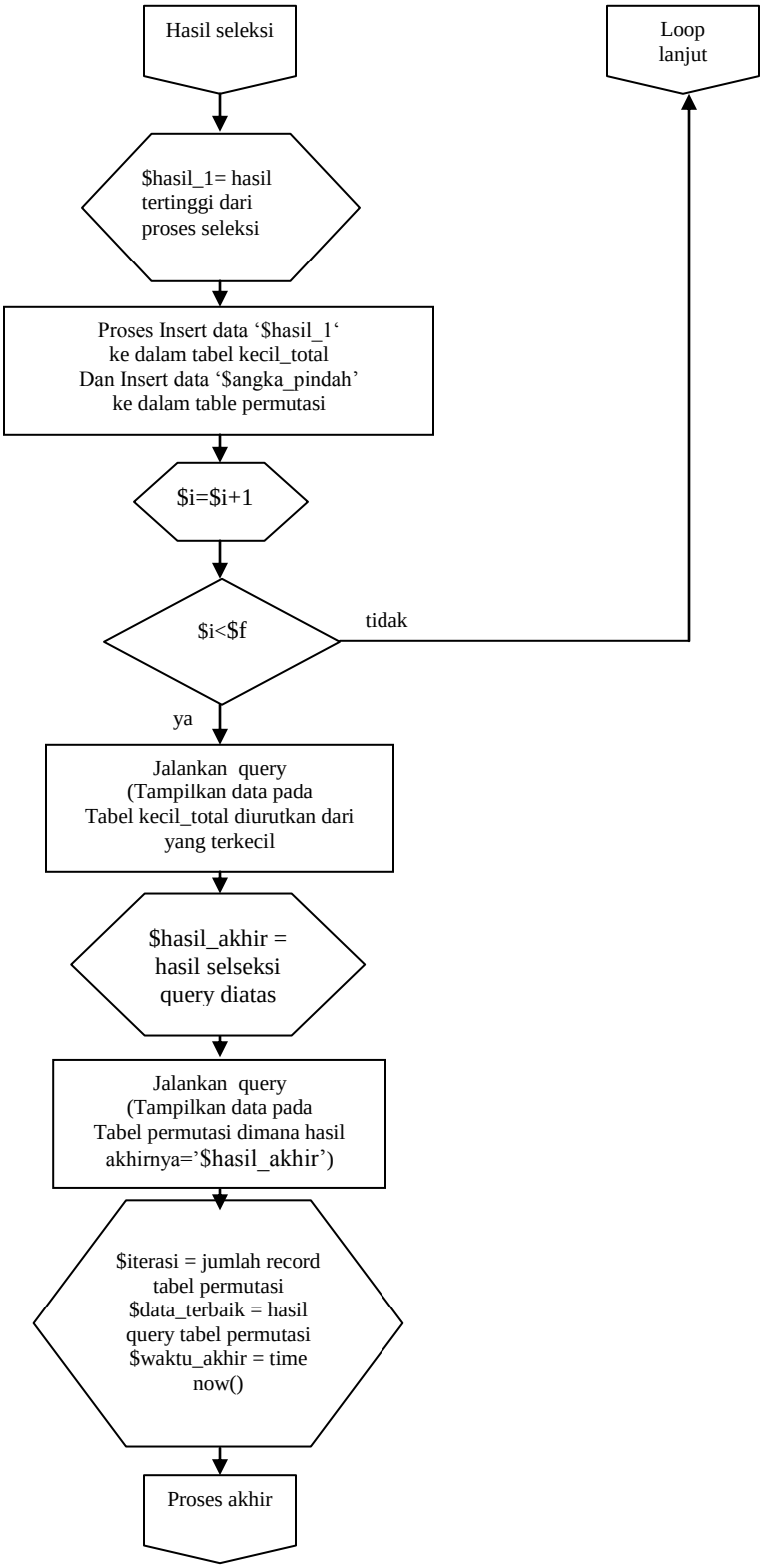
id	isi	hasil	id	isi	hasil	id	isi	hasil	id	isi	hasil	id	isi	hasil
0	2414313124332421	443	30	4423412411321332	439	60	3231344214221413	503	90	1442143133314222	556	120	2411233323442411	544
1	1413133312242244	733	31	1421442213312433	527	61	4113231422432431	556	91	2214311343241234	577	121	1244332112343412	461
2	3143242143211423	369	32	2323143244413211	536	62	2422231433141431	620	92	2234334211314214	524	122	1214331342422134	551
3	3314214231241423	458	33	3444323242111312	615	63	3214321432141324	405	93	4143124124233321	575	123	1234323441413221	531
4	2414321133413422	427	34	3312424311431224	393	64	4121334341241223	471	94	3322442441132131	437	124	4314121331432224	471
5	4243113432124231	540	35	3213324221441341	582	65	3124123142334142	463	95	2334131342442112	402	125	4214213231213434	522
6	4342331222114341	656	36	1124421344323321	484	66	2144132314321243	427	96	3142341123413224	369	126	4413114232122334	549
7	3224433211441321	617	37	4331144222132134	377	67	4142113443332212	556	97	3341412243213124	393	127	1112213244234433	508
8	1223332144431412	491	38	2442331442121313	470	68	2324341411124332	496	98	3421233341124124	405	128	4231334141423212	448
9	2141343321242341	640	39	3232341314242114	554	69	3142444333221112	665	99	1212141432343234	486	129	3312322443114214	534
10	1432144322132134	480	40	2143112413232344	463	70	4212324143211343	593	100	1432232414413321	536	130	4434412213212133	439
11	2134313224314241	491	41	1141223431423432	443	71	3221234134431124	544	101	4114221433334122	528	131	4224131234231431	731
12	2424213334111342	612	42	2244431313142123	538	72	2233434141321214	514	102	4134231232423411	461	132	3321421334142142	440
13	3222341431143421	559	43	2311341321442243	617	73	2412433314121342	377	103	3242343111443212	617	133	1211243142423334	571
14	2412133423324411	511	44	2321411134344322	541	74	1312241334144223	551	104	4122441133231342	623	134	3342233211144241	511
15	4132314132412342	377	45	3231313211224444	520	75	4323411231214423	491	105	2341412311432234	427	135	1414314133242322	556
16	2443121133123442	532	46	3141342123242341	511	76	1134434231243221	471	106	2433431141341222	506	136	2332431431414122	518
17	2214424233314131	568	47	2124214134313243	593	77	4221132414333241	733	107	2141242334143321	556	137	3341113241224243	549
18	2332142411231344	441	48	1133134432412242	564	78	2411434231324132	436	108	3442133231441221	418	138	1231114342432423	427
19	3131114423423224	490	49	1341312242313442	463	79	1124412423341233	527	109	3344131344122221	576	139	2231422311434431	604
20	3341121321344242	468	50	2322411413433421	582	80	1442121314243233	527	110	3342124414311232	470	140	3343324141212124	494
21	1112232442144333	527	51	1144322213334142	570	81	3433422344111221	540	111	1433143414222231	535	141	4333131124422412	506
22	1224421431213433	593	52	4341213211332424	532	82	4433324412111223	423	112	4144423311133222	526	142	4123444123112323	497
23	2142443112433231	702	53	4214333211342412	377	83	2441322413431231	710	113	3311411434222423	454	143	1133344432224211	620
24	3134413222123414	476	54	1341214413232324	480	84	1111432323224434	580	114	2413324341212413	470	144	2413224134131234	552
25	4232444311331122	536	55	3421213443142312	465	85	3413414312231422	369	115	3444223121321143	575	145	3424234411312312	399
26	4123322234131414	443	56	4132221233144431	746	86	3441214423312213	513	116	4311214324214323	427	146	2323243434112141	470

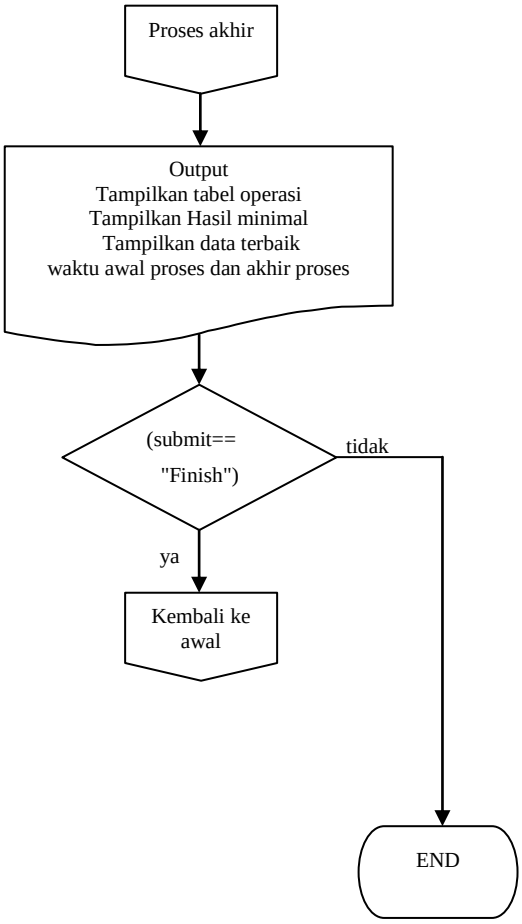
B. *Flowchart* Algoritma AIS




Halaman
Job shob
Kembali ke
awal
Input form
number of Job
and Machine
Input form
number of
machines and
time in the each
(submit==
"next")
tidak
ya
Import Data
excel
Proses Insert data ke
dalam database pada
tabel operasi
(submit==
"import")
tidak
ya
Proses Insert data ke
dalam database pada
tabel operasi
Tampilkan
tabel operasi
Form Specify the
following
Parameters
(submit==
"OK")
tidak
ya
Daftar
Library


Daftar Library
Tampilkan tabel library
(submit== "Result")
tidak
ya
$f= hasil faktorial dari job dan mesin
$angka= deretan angka permutasi
$waktu awal= time now
$f>1000
tidak
ya
$f= 1000
$i = 0
$angka_pindah = suffle($angka)
$x=0
$kark = ambil karakter dari '$angka_pindah' dimulai '$x' dari kiri sebanyak 1 karakter
Proses Seleksi query
Dari tabel operasi dimana id='$kark'
$x=$x+1
$x<= (job * mesin)
tidak
ya
Hasil seleksi
Loop lanjut


Hasil seleksi
Loop lanjut
$hasil_1= hasil tertinggi dari proses seleksi
Proses Insert data '$hasil_1' ke dalam tabel kecil_total Dan Insert data '$angka_pindah' ke dalam table permutasi
$i=$i+1
$i<$f
tidak
ya
Jalankan query (Tampilkan data pada Tabel kecil_total diurutkan dari yang terkecil
$hasil_akhir = hasil selseksi query diatas
Jalankan query (Tampilkan data pada Tabel permutasi dimana hasil akhirnya='$hasil_akhir')
$iterasi = jumlah record tabel permutasi
$data_terbaik = hasil query tabel permutasi
$waktu_akhir = time now()
Proses akhir


Proses akhir
Output
Tampilkan tabel operasi
Tampilkan Hasil minimal
Tampilkan data terbaik
waktu awal proses dan akhir proses
(submit==
"Finish")
tidak
ya
Kembali ke
awal
END

C. Skrip Program *Job Shop* dengan Algoritma AIS

```
<html>
<body>
<?
include "../konfigurasi/connect.php";
include "../konfigurasi/opendb.php";

if (!$proses){
mysql_query("delete from t_operasi");
mysql_query("delete from t_digit");
mysql_query("delete from t_permutasi");
mysql_query("delete from t_kecil");
mysql_query("delete from t_kecil_total");
?>
 <form name=oscar method=post action='?menu=job&proses=lanjut'
enctype=multipart/form-data>
                              <table border=0 cellspacing=0 cellpadding=0
align=center class=datatable>
                                <tr>
                                 <td colspan=3 bgcolor=E5FB69>:: Form 1
::</td>
                                </tr>
                                   <tr>
                                          <td width=200>Number of Jobs</td>
                                          <td>:</td>
                                          <td align=center>

                                          <input type=text name=job size=3
maxlength=1 class=hrf-1 onkeyup=this.value=this.value.replace(/[^1-
9,'(',')']/g,'')></td>
                                   </tr>
                                   <tr>
                                          <td valign=top>Number of
Machines</td>
                                          <td valign=top>:</td>
                                          <td align=center><input type=text
name=mesin size=3 maxlength=1 class=hrf-1
onkeyup=this.value=this.value.replace(/[^1-9,'(',')']/g,'')></td>

                                   </tr>
                                   <tr>
                                          <td colspan=3><input type=checkbox
name=asembli value='1'> With Assembling</td>
                                   </tr>
                                   <tr>
                                          <td colspan=2 align=center><font
class=hrf-2>* Input with number !</font></td>
                                          <td><input type=submit value='Next
>>' class=button></td>
                                   </tr>
                              </table>
                    </form>
<? } if ($proses=='lanjut'){
$job=$_POST['job'];
$mesin=$_POST['mesin'];
$asembli=$_POST['asembli'];
```

```
if($asembli==1){ $asembli_nilai=1; }else{ $asembli_nilai=0;}

echo "Input Number of Machines and Time in the each operation or Import File
Excel<br><br>";
$banyak=$mesin*$job;

echo" <form name=oscar2 method=post action='?menu=job&proses=lanjut2'
enctype=multipart/form-data>
<input type=hidden name='job' value='$job'><input type=hidden name='mesin'
value='$mesin'><input type=hidden name='banyak' value='$banyak'>
<input type=hidden name='asembli' value='$asembli'>
<table cellspacing=0 cellpadding=0><tr><td bgcolor=E5FB69>Job</td>";

if($asembli==1){
$mesin_kurang=$mesin-1;
//echo $mesin_kurang;
echo"<input type=hidden name=asli value='";
for($d=1;$d<=$mesin_kurang;$d++){  for($dx=1;$dx<=$job;$dx++){  echo $dx; }
}
echo"' size=60>";
}else{
echo"<input type=hidden name=asli value='";
for($d=1;$d<=$mesin;$d++){  for($dx=1;$dx<=$job;$dx++){  echo $dx; } }
echo"' size=60>";
}

for ($ix=1; $ix<=$mesin; $ix++)
{ echo"<td width=70 align=center bgcolor=E5FB69>Opn. $ix</td>"; }

echo"</tr><tr><td align=center>1</td>";
//looping txtbox value='$number'
$nomer=1;
$baris=1;
while($baris<=$job) {

$numbers = range(1, $mesin);
shuffle($numbers);
$no_opn=1;
foreach ($numbers as $number) {

echo"<td align=center>
<input type=text name=mesinx[] size=3 maxlength=1 class=hrf-1
onkeyup=this.value=this.value.replace(/[^1-9,'(',')']/g,'')>
<input type=text name=waktux[] size=3 maxlength=2 class=hrf-1
onkeyup=this.value=this.value.replace(/[^0-9,'(',')']/g,'')></td>";
$fil="m".$no_opn."_".$nomer;
$datanya=$no_opn."mesin".$nomer;

$no_opn++;
}
$nomer++;
if($baris<=$job-1){echo "</tr><tr><td align=center>$nomer</td>";}
$baris++;
}

echo"</tr>
<tr><td> </td><td colspan=$mesin><input type=submit value='Next >>'
class=button></td></tr>
```

```
</table></form><table height=20><tr><td></tr></td></table>";

echo" <form name=oscar2 method=post action='?menu=job&proses=lanjut2x'
enctype=multipart/form-data>
<input type=hidden name='job' value='$job'><input type=hidden name='mesin'
value='$mesin'><input type=hidden name='banyak' value='$banyak'>
<input type=hidden name='asembli' value='$asembli'>";

if($asembli==1){
$mesin_kurang=$mesin-1;
//echo $mesin_kurang;
echo"<input type=hidden name=asli value='";
for($d=1;$d<=$mesin_kurang;$d++){  for($dx=1;$dx<=$job;$dx++){  echo $dx; }
}
echo"' size=60>";
}else{
echo"<input type=hidden name=asli value='";
for($d=1;$d<=$mesin;$d++){  for($dx=1;$dx<=$job;$dx++){  echo $dx; } }
echo"' size=60>";
}

echo"<table border=0 width=810 cellspacing=0 cellpadding=0 class=datatable>
       <tr>
              <td colspan=3 bgcolor=#E5FB69>Import File Excel</td>
       </tr>
       <tr>
              <td width=110>Import File</td>
              <td width=10 align=center>:</td>
              <td><input name='userfile' type='file' class=button></td>
       </tr>
       <tr>
              <td> </td>
              <td> </td>
              <td><input name='upload' type='submit' value='Import'
class=button> </td>
       </tr>
</table>
</form>";
}
if ($proses==lanjut2x){
//-----awal code import
// menggunakan class phpExcelReader
include "../job/excel_reader2.php";

mysql_query("insert into t_digit (id,isi)values('1','$asli')");
// membaca file excel yang diupload
$data = new Spreadsheet_Excel_Reader($_FILES['userfile']['tmp_name']);

// membaca jumlah baris dari data excel
$baris = $data->rowcount($sheet_index=0);

// nilai awal counter untuk jumlah data yang sukses dan yang gagal diimport
$sukses = 0;
$gagal = 0;
// $i=2 =import data excel mulai baris ke-2 (karena baris pertama adalah
nama kolom)
$ii=1;
for ($i=2; $i<$baris+1; $i++)
```

```
{
  // membaca data ci (kolom ke-1)
$m1 = $data->val($i, 1);
$o1 = $data->val($i, 2);
$m2 = $data->val($i, 3);
$o2 = $data->val($i, 4);
$m3 = $data->val($i, 5);
$o3 = $data->val($i, 6);
$m4 = $data->val($i, 7);
$o4 = $data->val($i, 8);
$m5 = $data->val($i, 9);
$o5 = $data->val($i, 10);
$m6 = $data->val($i, 11);
$o6 = $data->val($i, 12);
$m7 = $data->val($i, 13);
$o7 = $data->val($i, 14);
$m8 = $data->val($i, 15);
$o8 = $data->val($i, 16);
$m9 = $data->val($i, 17);
$o9 = $data->val($i, 18);
  // setelah data dibaca, sisipkan ke dalam tabel
  $query = "INSERT INTO
t_operasi(id,m1,o1,m2,o2,m3,o3,m4,o4,m5,o5,m6,o6,m7,o7,m8,o8,m9,o9)VALUES('$
ii','$m1','$o1','$m2','$o2','$m3','$o3','$m4','$o4','$m5','$o5','$m6','$o6',
'$m7','$o7','$m8','$o8','$m9','$o9')";
  $hasil=mysql_query($query);
$ii++;
  // jika proses insert data sukses, maka counter $sukses bertambah
  // jika gagal, maka counter $gagal yang bertambah
  if ($hasil){ $sukses++; }
  else{ $gagal++; }
}
//echo $query;
// tampilan status sukses dan gagal
echo "<script language='javascript'> alert('Import is Finished, Succses :
$sukses, Error : $gagal');</script>";
//-----akhir code import

//-- tampil tabel
echo"<table border=0 cellspacing=0 cellpadding=0 class=datatable><tr
bgcolor=E5FB69 align=center><td rowspan=2><b>Job</b></td><td colspan=18
align=center><b>Operation</b> (M=Machine,T=Time)</td></tr>
<tr bgcolor=E5FB69 align=center><td>M 1</td><td>T 1</td><td>M 2</td><td>T
2</td><td>M 3</td><td>T 3</td><td>M 4</td><td>T 4</td><td>M 5</td><td>T
5</td><td>M 6</td><td>T 6</td><td>M 7
</td><td>T 7</td><td>M 8</td><td>T 8</td><td>M 9</td><td>T 9</td></tr>";
$tabel_query=mysql_query("select * from t_operasi order by id asc");
$urut_job=1;
while($rec=mysql_fetch_array($tabel_query)){
echo "<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';
align=center><td>$urut_job</td><td>$rec[m1]</td><td>$rec[o1]</td><td>$rec[m2
]</td><td>$rec[o2]</td>
<td>$rec[m3]</td><td>$rec[o3]</td><td>$rec[m4]</td><td>$rec[o4]</td><td>$rec
[m5]</td><td>$rec[o5]</td><td>$rec[m6]</td><td>$rec[o6]</td><td>$rec[m7]</td
><td>$rec[o7]</td>
<td>$rec[m8]</td><td>$rec[o8]</td><td>$rec[m9]</td><td>$rec[o9]</td>
</tr>";
```

```
$urut_job++; }
echo"</table><br><br>";
//--
echo"
 <form method=post action='?menu=job&proses=library' enctype=multipart/form-
data>
 <input type=hidden name='job' value='$job'><input type=hidden name='mesin'
value='$mesin'><input type=hidden name='banyak' value='$banyak'>
 <input type=hidden name='asembli' value='$asembli'>
 ";  ?>
<table border=0 cellspacing=0 cellpadding=0 class=datatable>
<tr>
<td colspan=7 bgcolor=E5FB69>:: Specify the following Parameters ::</td>
</tr>
<tr>
<td width=250>No. of Antibody Libraries</td>
<td>:</td>
<td>
<script>
function CBtoTB()
{
var a=document.getElementById('a').value;
document.getElementById("TextBox").value=a/(document.getElementById("ComboBo
x").value)
if((document.getElementById("ComboBox").value)==0)document.getElementById("T
extBox").value=''
}
</script>

<select name=aaa id="ComboBox" onchange="CBtoTB()" class=hrf-1>
<option value='' class=hrf-1></option>
<?
if($asembli==1){
$banyak_asembli=($mesin-1)*$job;
}else{
$banyak_asembli=$mesin*$job;
}
for ($i=1; $i<=$banyak_asembli; $i++) {
if($banyak_asembli%$i==0){ echo"<option value='$i' class=hrf-1>$i</option>";
}
}
?>
</select>
</td>
</tr>
<tr>
<td valign=top>No. of Components Library</td>
<td valign=top>:</td>
<td><input type=text name=bbb size=3 maxlength=1 class=hrf-1
onkeyup=this.value=this.value.replace(/[^1-9,'(',')']/g,'')></td>

</tr>
<tr>
<td valign=top>No. of Genes in Component</td>
<td valign=top>:</td>
<td>
<input type=text name=ccc size=3 maxlength=1 class=hrf-1 id="TextBox"
onkeyup=this.value=this.value.replace(/[^1-9,'(',')']/g,'')>
```

```
<input id="a" type=hidden value="<? echo"$banyak_asembli"; ?>">
</td>
</tr>
<tr>
<td colspan=2> </td>
<td><input type=submit value=OK class=button></td>

</tr>
</table>

</form>
<? }
if ($proses==lanjut2){

mysql_query("insert into t_digit (id,isi)values('1','$asli')");
//--------------------------
for ($masuk=1; $masuk<=$job; $masuk++){mysql_query("insert into t_operasi
(id)values('$masuk')");}// di buat insert dlu kosongan ntar biar mudah
updatenya heheheh


$enol=1;
$sampek=$mesin;
$field_masuk=0;
for ($rec=1; $rec<=$job; $rec++){
$masuk2=1;
for ($masuk1=$enol; $masuk1<=$sampek; $masuk1++){
mysql_query("update t_operasi set
m$masuk2='$mesinx[$field_masuk]',o$masuk2='$waktux[$field_masuk]' where
id=$rec"); $field_masuk++; $masuk2++;
       }
$enol++;
$sampek=($enol-1)+$mesin;
}

echo"<input type=hidden name='asli' value='$asli'>
<input type=hidden name='job' value='$job'><input type=hidden name='mesin'
value='$mesin'><input type=hidden name='banyak' value='$banyak'><input
type=hidden name='asembli' value='$asembli'>
<table cellspacing=0 cellpadding=0><tr>";


//-- tampil tabel
$tabel_query=mysql_query("select * from t_operasi order by id asc");
echo"<table border=0 cellspacing=0 cellpadding=0 class=datatable><tr
bgcolor=E5FB69><td rowspan=2 align=center><b>Job</b></td><td colspan=18
align=center><b>Operation</b> (M=Machine,T=Time)</td></tr>
<tr bgcolor=E5FB69 align=center><td>M 1</td><td>T 1</td><td>M 2</td><td>T
2</td><td>M 3</td><td>T 3</td><td>M 4</td><td>T 4</td><td>M 5</td><td>T
5</td><td>M 6</td><td>T 6</td><td>M 7
</td><td>T 7</td><td>M 8</td><td>T 8</td><td>M 9</td><td>T 9</td></tr>";
$urut_job=1;
while($rec=mysql_fetch_array($tabel_query)){
echo "<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';
align=center><td>$urut_job</td><td>$rec[m1]</td><td>$rec[o1]</td><td>$rec[m2
]</td><td>$rec[o2]</td>
```

```
<td>$rec[m3]</td><td>$rec[o3]</td><td>$rec[m4]</td><td>$rec[o4]</td><td>$rec
[m5]</td><td>$rec[o5]</td><td>$rec[m6]</td><td>$rec[o6]</td><td>$rec[m7]</td
><td>$rec[o7]</td>
<td>$rec[m8]</td><td>$rec[o8]</td><td>$rec[m9]</td><td>$rec[o9]</td>
</tr>";
$urut_job++; }
echo"</table><br><br>";
//--
echo"
 <form name=oscar3 method=post action='?menu=job&proses=library'
enctype=multipart/form-data>
 <input type=hidden name='job' value='$job'><input type=hidden name='mesin'
value='$mesin'><input type=hidden name='banyak' value='$banyak'>";  ?>
<table border=0 cellspacing=0 cellpadding=0 class=datatable>
<tr>
<td colspan=7 bgcolor=E5FB69>:: Specify the following Parameters ::</td>
</tr>
<tr>
<td width=250>No. of Antibody Libraries</td>
<td>:</td>
<td>
<script>
function CBtoTB()
{
var a=document.getElementById('a').value;
document.getElementById("TextBox").value=a/(document.getElementById("ComboBo
x").value)
if((document.getElementById("ComboBox").value)==0)document.getElementById("T
extBox").value=''
}
</script>

<select name=aaa id="ComboBox" onchange="CBtoTB()">
<option value='' class=hrf-1></option>
<?
if($asembli==1){
$banyak_asembli=($mesin-1)*$job;
}else{
$banyak_asembli=$mesin*$job;
}
for ($i=1; $i<=$banyak_asembli; $i++) {
if($banyak_asembli%$i==0){ echo"<option value='$i' class=hrf-1>$i</option>";
}
}
?>
</select>
</td>
</tr>
<tr>
<td valign=top>No. of Components Library</td>
<td valign=top>:</td>
<td><input type=text name=bbb size=3 maxlength=1 class=hrf-1
onkeyup=this.value=this.value.replace(/[^1-9,'(',')']/g,'')></td>

</tr>
<tr>
<td valign=top>No. of Genes in Component</td>
<td valign=top>:</td>
```

```
<td>
<input type=text name=ccc size=3 maxlength=1 class=hrf-1 id="TextBox"
onkeyup=this.value=this.value.replace(/[^1-9,'(',')']/g,'')>
<input id="a" type=hidden value="<? echo"$banyak_asembli"; ?>">
</td>
</tr>
<tr>
<td colspan=2> </td>
<td><input type=submit value=OK class=button></td>

</tr>
</table>

 </form>
<? }if ($proses==library){

$aaa=$_POST['aaa'];
$bbb=$_POST['bbb'];
$ccc=$_POST['ccc'];
$ddd=$_POST['ddd'];
$kotak_kecil=$bbb*$ccc;
//mulai menyekrip dg sakti :
echo" <form name=oscar4 method=post action='?menu=job&proses=result'
enctype=multipart/form-data>
<input type=hidden name='job' value='$job'><input type=hidden name='mesin'
value='$mesin'><input type=hidden name='banyak' value='$banyak'>
<input type=hidden name='asembli' value='$asembli'>";

echo"<table cellspacing=0 cellpadding=0><tr>";

if($aaa%4==0){$ganti=4;}
elseif($aaa%3==0){$ganti=3;}
else{$ganti=5;}

if($aaa==4){$ganti=2;}
if($aaa==2){$ganti=1;}

for($no_library=1; $no_library<=$aaa; $no_library++) {
echo"<td><table cellspacing=0 cellpadding=0><tr><td colspan=$ccc
bgcolor=E5FB69>Library $no_library :</td></tr><tr>";

//looping txtbox
$nomer=1;
$baris=1;
while($baris<=$bbb) {

$numbersz = range(1, $ccc);
shuffle($numbersz);
$no_opn=1;
foreach ($numbersz as $numberz) {
$acak=rand(1,$job);
echo"<td align=center><input type=text name=lib[] value='$acak' size=3
maxlength=1 class=hrf-1 onkeyup=this.value=this.value.replace(/[^1-
9,'(',')']/g,'')></td>";

$no_opn++;
}
$nomer++;
```

```
if($baris<=$bbb-1){echo "</tr><tr>";}
$baris++;
}
echo"</tr></table></td>";
if($no_library%$ganti==0){echo "</tr><tr>";}
}
echo"</tr>
<tr><td colspan=$ccc align=right><input type=submit value='Result >>'
class=button></td></tr>
</table></form>";

}if ($proses==result){
$awal_waktunya1=gmdate("H:i:s", time()+60*60*7);
echo"<table border=0 cellspacing=0 cellpadding=0 class=datatable width=610
align=center><tr bgcolor=E5FB69 align=center><td
rowspan=2><b>Job</b></td><td colspan=18 align=center><b>Operation</b>
(M=Machine,T=Time)</td></tr><tr bgcolor=E5FB69 align=center><td>M
1</td><td>T 1</td><td>M 2</td><td>T 2</td><td>M 3</td><td>T 3</td><td>M
4</td><td>T 4</td><td>M 5</td><td>T 5</td><td>M 6</td><td>T 6</td><td>M
7</td><td>T 7</td><td>M 8</td><td>T 8</td><td>M 9</td><td>T 9</td></tr>";
$tabel_query=mysql_query("select * from t_operasi order by id asc");
$urut_job=1;
while($rec=mysql_fetch_array($tabel_query)){
echo "<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';
align=center><td>$urut_job</td><td>$rec[m1]</td><td>$rec[o1]</td><td>$rec[m2
]</td><td>$rec[o2]</td><td>$rec[m3]</td><td>$rec[o3]</td><td>$rec[m4]</td><t
d>$rec[o4]</td><td>$rec[m5]</td><td>$rec[o5]</td><td>$rec[m6]</td><td>$rec[o
6]</td><td>$rec[m7]</td><td>$rec[o7]</td><td>$rec[m8]</td><td>$rec[o8]</td><
td>$rec[m9]</td><td>$rec[o9]</td></tr>";
$urut_job++; }
echo"</table><br><br>";
//--
$query="select * from t_operasi order by id asc";
$jalan=mysql_query($query);

$ambil=mysql_query("select * from t_digit");
$ambilkan=mysql_fetch_array($ambil);
$ds=$ambilkan['isi'];

function factorial($x)
{
$hasil = 1;
if($x > 1)
{
for($i=2;$i<=$x;$i++){$hasil = $hasil * $i;}
}
return $hasil;
}
$kali=$job*$mesin;
$h=factorial($kali);
$gigaliun=$ambilkan['isi'];

if($h>1000){$h=1000;}
for($i=0;$i<$h;$i++)
{
$tt=str_shuffle($gigaliun);
```

```
$mgant[1]=0; $mgant[2]=0; $mgant[3]=0; $mgant[4]=0; $mgant[5]=0;
$mgant[6]=0; $mgant[7]=0; $mgant[8]=0; $mgant[9]=0; $mgant[10]=0;
$jgant[1]=0; $jgant[2]=0; $jgant[3]=0; $jgant[4]=0; $jgant[5]=0;
$jgant[6]=0; $jgant[7]=0; $jgant[8]=0; $jgant[9]=0; $jgant[10]=0;

for($x=0;$x<=$kali-1;$x++){

$ttx=substr($tt,$x,1);
$cari_data=mysql_query("select * from t_operasi where id='$ttx'");
$ketemu=mysql_fetch_array($cari_data);
$erai_satu[$x]=$ttx;

$f_nya=1;
if($x==0){$f_nya=1;}
elseif($x==1){  if($erai_satu[0]==$ttx){$f_nya=2;} }
else{ for($x2=0;$x2<=$x-1;$x2++){ if($erai_satu[$x2]==$ttx){
$f_nya=$f_nya+1; } } }

switch ($f_nya)
{
case 1:
$mf=$ketemu['m1'];
$wf=$ketemu['o1'] ;
break;
case 2 :
$mf=$ketemu['m2'];
$wf=$ketemu['o2'] ;
break;
case 3 :
$mf=$ketemu['m3'];
$wf=$ketemu['o3'] ;
break;
case 4 :
$mf=$ketemu['m4'];
$wf=$ketemu['o4'] ;
break;
case 5 :
$mf=$ketemu['m5'];
$wf=$ketemu['o5'] ;
break;
case 6 :
$mf=$ketemu['m6'];
$wf=$ketemu['o6'] ;
break;
case 7 :
$mf=$ketemu['m7'];
$wf=$ketemu['o7'] ;
break;
case 8 :
$mf=$ketemu['m8'];
$wf=$ketemu['o8'] ;
break;
case 9 :
$mf=$ketemu['m9'];
$wf=$ketemu['o9'] ;
break;
}
```

```
if($jgant[$ttx]>$mgant[$mf]){ $jgant[$ttx]=$jgant[$ttx]+$wf;
$mgant[$mf]=$jgant[$ttx]; }
elseif($jgant[$ttx]<$mgant[$mf]){ $mgant[$mf]=$mgant[$mf]+$wf;
$jgant[$ttx]=$mgant[$mf]; }
else{$jgant[$ttx]=$jgant[$ttx]+$wf; $mgant[$mf]=$mgant[$mf]+$wf; }
$masuk_tabel=$jgant[$ttx];

mysql_query("insert into t_kecil (id,isi)values('$x','$masuk_tabel')");
}
$lihat_kecil=mysql_query("select * from t_kecil order by isi desc limit
0,1");
$kecil=mysql_fetch_array($lihat_kecil);
$masuk_total=$kecil['isi'];
mysql_query("insert into t_kecil_total
(id,isi)values('$i','$masuk_total')");
//--
mysql_query("delete from t_kecil");
//--
mysql_query("insert into t_permutasi
(id,isi,hasil)values('$i','$tt','$masuk_total')");
}
$lihat_total=mysql_query("select * from t_kecil_total order by isi asc limit
0,1");
$terkecil=mysql_fetch_array($lihat_total);
$hasil_akhirnya=$terkecil['isi'];

$tampilin=mysql_query("SELECT DISTINCT isi FROM t_permutasi WHERE
hasil='$hasil_akhirnya'");
$akeh=mysql_num_rows($tampilin);
$akhir_waktu= gmdate("H:i:s", time()+60*60*7);

if($asembli==1){
$query_asembli=mysql_query("SELECT sum(o$mesin)as ju FROM t_operasi");
$sum_asembli=mysql_result($query_asembli,0,'ju');
}
$jum_all=$hasil_akhirnya + $sum_asembli;

echo"
<table cellspacing=0 cellpadding=0 border=0 align=center>
<tr><td bgcolor=E5FB69><b>Results</b></td></tr>
<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';><td valign=top width=600>The
minimum makespan is : <b>$hasil_akhirnya</b> units</td></tr>";
if($asembli==1){ echo"<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';><td valign=top width=600>The
minimum makespan with assembling : <b>$hasil_akhirnya + $sum_asembli =
$jum_all</b> units</td></tr>";  }
echo"<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';><td valign=top>The number of
Iterations made in this run is : <b>$i</b></td></tr>
<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';><td valign=top>Start Time  :
<b>$awal_waktunya1</b></td></tr>
<tr onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';><td valign=top>Finish Time :
<b>$akhir_waktu</b></td></tr>
<tr><td bgcolor=F3F4F5>The best schedule generated is ( <i>$akeh Record
</i>  ) :</td></tr>";
```

```
while($liat=mysql_fetch_array($tampilin)){echo "<tr
onMouseOver=style.backgroundColor='C0FAFC';
onMouseOut=style.backgroundColor='white';><td>$liat[isi]";
if($asembli==1){ echo" + "; for($dx=1;$dx<=$job;$dx++){ echo $dx; } }
echo"</td></tr>";  }

echo"
<tr><td align=center><a href='?menu=job'><input type=submit value=Finish
class=button></a></td></tr>
</table><br><br>";
}
?>
</body>
</html>
```