PENJADWALAN *JOB SHOP* MENGGUNAKAN ALGORITMA SEMUT

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar sarjana dalam bidang ilmu komputer

Oleh :

Fahmi Hamdani I. Adha
0810962005-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2011

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PENJADWALAN *JOB SHOP* MENGGUNAKAN ALGORITMA SEMUT

oleh:
Fahmi Hamdani I. Adha
0810962005-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 26 Mei 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Dian Eka Ratnawati, S.Si, M.Kom	Drs. Marji, MT
NIP. 197306192002122001	NIP. 196708011992031001

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP.196709071992031001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**:**	Fahmi Hamdani I. Adha
NIM	**:**	0810962005-96
Jurusan	**:**	Matematika
Program Studi	**:**	Ilmu Komputer
Penulis tugas akhir berjudul	**:**	Penjadwalan *Job Shop* Menggunakan Algoritma Semut

Dengan ini menyatakan bahwa :

1. **Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila dikemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 26 Mei 2011
Yang menyatakan,

Fahmi Hamdani I. Adha
NIM. 0810962005

PENJADWALAN *JOB SHOP*
MENGGUNAKAN ALGORTIMA SEMUT

ABSTRAK

Penjadwalan sangat dibutuhkan pada dunia industri seperti pada industri *manufacturing* yang menggunakan mesin untuk proses produksi. Proses penjadwalan timbul jika terdapat keterbatasaan sumber daya yang dimiliki sehingga diperlukan adanya pengaturan sumber daya yang efisien. Penjadwalan *job shop* merupakan masalah pengalokasian sejumlah pekerjaan (*job*) yang harus diproses melalui sejumlah mesin. Salah satu metode yang dapat digunakan untuk mengoptimasi penjadwalan *job shop* adalah metode heuristik. Algoritma semut termasuk salah satu metode pemecahan masalah yang bersifat heuristik yang diambil dari perilaku koloni semut dalam mencari rute terpendek antara sarang dan sumber makanan. Untuk mengetahui tingkat optimasi algoritma semut dalam permasalahan penjadwalan *job shop* maka dilakukan pengujian pengaruh parameter algoritma semut terhadap *makespan* yang dihasilkan. Pada uji coba digunakan data dari *OR-Library* yaitu *Fisher and Thompson 6x6 Instance* yang berisi 6 *job* dan 6 mesin yang telah diketahui mempunyai *makespan* optimum sebesar 55. Berdasarkan uji coba yang dilakukan, nilai *makespan* terbaik diperoleh dengan menggunakan parameter algoritma semut yaitu jumlah semut 50, nilai alpha 10, nilai betha 10, nilai Q 250, intensitas *pheromone* awal 0.05, dan jumlah siklus 2000. Dari kombinasi parameter tersebut dihasilkan nilai *makespan* sebesar 56,6 yang mendekati nilai *makespan* data *benchmark.*

JOB SHOP SCHEDULLING
USING ANT ALGORITHM

ABSTRACT

Scheduling is needed in the industry such as in manufacturing industries that use machinery for production procces. Process of scheduling arise if the resource is limited so efficient resource management is needed. Job shop scheduling is a problem of allocating number of jobs that is processed through number of machines. One method that can be used to optimize the job shop scheduling problems is a heuristic methods. And ant colony algorithm is the one of heuristic methods that inspired from the behaviour of ants in search of shortest route between nest and food sources. To know the level of ant algorithm optimization in job shop scheduling problems the influences of ant algorithm parameter to makespan value is tested. This test uses "Fisher and Thompson 6x6 instance" data from the OR-Library that contains 6 jobs and 6 machines that have been known optimum makespan of this problems is 55. Based on experiments performed, minimum makespan is resulted when the number of ants is 50, alpha values is 10, betha value is 10, Q value is 250, initial pheromone intensity is 0.05, and the number of cycles is 2000. The best makespan value resulted from good parameters combination is 56,6 that close to the makespan value of *benchmark* data.

KATA PENGATAR

Puji syukur penulis ucapkan atas kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahnya, skripsi yang berjudul "Penjadwalan *Job Shop* Menggunakan Algoritma Semut" ini dapat diselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer.

Dalam menyelesaikan skripsi ini, penulis telah mendapat banyak bantuan dari banyak pihak. Atas bantuan tersebut penulis ingin mengucapkan terima kasih yang sedalam-dalamnya kepada :

1. Dian Eka Ratnawati, S.Si, M.Kom, selaku pembimbing utama penulisan skripsi ini.
2. Drs. Marji, MT, selaku pembimbing pendamping dan selaku ketua program studi Ilmu Komputer.
3. Dr. Abdul Rouf Alghofari, M.Sc, selaku ketua jurusan Matematika.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di program studi Ilmu Komputer jurusan Matematika FMIPA Universitas Brawijaya.
5. Segenap staf dan karyawan di jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu penulis dalam penulisan skripsi ini.
6. Orang tua dan keluarga yang memeberikan dukungan berupa materi dan doa restu yang tak terbatas kepada penulis.
7. Kawan-kawan di program studi Ilmu Komputer yang telah banyak memberikan saran dan bantuan demi kelancaran penulisan skripsi ini.
8. Dan semua pihak yang telah membantu dalam penulisan skripsi ini yang tidak dapat penulis sebutkan satu persatu.

Semoga penulisan skripsi ini bermanfaat bagi pembaca sekalian. Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, dan ada kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, Mei 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCE CODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penjadwalan sangat dibutuhkan pada dunia industri seperti pada industri *manufacturing* yang menggunakan mesin untuk proses produksi. Pada proses produksi penjadwalan memegang peranan penting untuk meningkatkan sumber daya dan mengurangi waktu tunggu produksinya, sehingga waktu total proses bisa berkurang dan produktifitasnya dapat meningkat. Selain itu, penjadwalan dapat mengurangi beberapa keterlambatan pada pekerjaan yang mempunyai batas waktu penyelesaian.

Proses penjadwalan timbul jika terdapat keterbatasan sumber daya yang dimiliki sehingga diperlukan adanya pengaturan sumber daya tersebut secara efisien. Berbagai model penjadwalan telah dikembangkan untuk mengatasi persoalan tersebut. Model tersebut antara lain mesin yang digunakan dapat berupa proses dengan mesin tunggal atau proses dengan mesin majemuk, pola aliran proses dapat berupa aliran identik atau sembarang (*flow shop* atau *job shop*), pola kedatangan pekerjaan statis atau dinamis, sifat informasi yang diterima dapat bersifat deterministik atau stokastik (Baker, 1974).

Penjadwalan job shop merupakan masalah pengalokasian sejumlah pekerjaan (*job*) yang harus diproses melalui sejumlah mesin. Proses dari suatu *job* pada suatu mesin disebut operasi. Setiap *job* terdiri dari satu set operasi dengan waktu tertentu dan urutan-urutan operasi pada mesin sudah terspesifikasi (Rosnani, 2009). Salah satu kriteria performansi yang digunakan untuk mengevaluasi ukuran keberhasilan dari suatu penjadwalan adalah *makespan. Makespan* sendiri merupakan total waktu yang diperlukan untuk menyelesaikan seluruh *job.*

Manfaat menggunakan bantuan komputer dalam penjadwalan *job shop* adalah waktu yang dibutuhkan untuk memperoleh alternatif mesin tidak lama sehingga dapat mempermudah dalam membuat penjadwalan *job shop* untuk beberapa mesin sekaligus. Selain itu juga keakuratan dalam pemilihan alternatif mesin lebih konsisten sehingga meminimalkan kegagalan dalam proses produksi.

Terdapat bermacam-macam metode yang dapat digunakan untuk masalah optimasi penjadwalan *job shop*. Salah satu metode yang dapat digunakan untuk menangani masalah waktu dan biaya yang sangat besar dalam penjadwalan *jobshop* adalah metode heuristik.

Dan algoritma semut *(ant colony optimization algorithm)* merupakan salah satu metode pemecahan masalah yang bersifat heuristik.

Algoritma semut diambil dari perilaku koloni semut dalam pencarian rute terpendek antara sarang dan sumber makanan yang dikenal dengan sistem semut (Dorigo, 1996). Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki (*pheromone*) pada lintasan yang dilalui. Semakin banyak semut yang melalui suatu lintasan maka akan semakin jelas bekas jejak kakinya sehingga untuk menemukan jalur selanjutnya digunakan pemilihan acak berdasarkan besarnya *pheromone* yang dimiliki oleh jalur tersebut.

Implementasi algoritma semut telah banyak dilakukan pada permasalahan seperti TSP, s*panning tree*, dan penjadwalan. Pada penelitian yang dilakukan oleh Dorigo (1996) pada permasalahan *Traveling Saleman Problem* (TSP) algoritma semut dapat menghasilkan solusi yang optimum.

Oleh karena itu pada penelitian kali ini, ingin dicoba mengimplentasikan algoritma semut untuk menyelesaikan penjadwalan *job shop* dan mengetahui solusi optimal yang bisa dihasilkan.

Berdasarkan latar belakang yang telah di jelaskan di atas maka judul yang diambil untuk skripsi ini adalah "***PENJADWALAN JOB SHOP DENGAN MENGGUNAKAN ALGORITMA SEMUT***".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang ada, maka dapat dirumuskan permasalahannya, yaitu:

1. Bagaimana merancang sebuah sistem aplikasi penjadwalan *job shop* dengan menggunakan algoritma semut untuk menghasilkan *makespan* yang minimum.
2. Berapa nilai parameter jumlah semut, paramater intensitas *pheromone* awal, Parameter α, Parameter β, Parameter ρ, parameter Q, dan parameter jumlah siklus yang menghasilkan *makespan* minimum dalam penjadwalan *job shop*.

1.3 Batasan Masalah

Pada skripsi ini permasalahan yang ada dibatasi pada :

1. Optimasi penjadwalan *job shop* berdasarkan total waktu proses minimum (minimum *makespan*).
2. Pada waktu yang sama mesin hanya boleh melakukan satu proses.
3. Penjadwalan *job shop* yang digunakan adalah *job shop* statis.
4. Data yang digunakan untuk uji coba berasal dari *OR-Library* yaitu *Fisher and Thompson 6x6 instance* yang berisi 6 *job* dan 6 mesin.

1.4 Tujuan

Tujuan yang ingin dicapai dari penulisan skripsi ini adalah :

1. Merancang dan membuat aplikasi penjadwalan *job shop* dengan menggunakan algoritma semut untuk menghasilkan *makespan* yang minimum.
2. Melakukan analisis terhadap parameter jumlah semut, paramater intensitas *pheromone* awal, Parameter α, Parameter β, Parameter ρ, parameter Q, dan paramater jumlah siklus sehingga diketahui nilai parameter algoritma semut yang dapat menghasilkan makespan yang minimum.

1.5 Manfaat

Manfaat yang diharapkan bisa diambil dari penulisan skripsi ini adalah memberikan metode alternatif dalam permasalahan penjadwalan *job shop* dengan menggunakan algoritma semut untuk mendapatkan urutan prioritas mesin dengan *makespan* yang minimum.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah :

1. Studi literatur
   Mempelajari teori-teori yang berhubungan dengan penjadwalan *job shop* dan Algoritma semut dari berbagai referensi.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan analisis sistem
   Membuat perancangan perangkat lunak dan mengimplementasikan hasilnya untuk membuat perangkat lunak tersebut.
4. Uji coba dan analisa hasil implementasi

Menguji perangkat lunak dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya untuk kemudian di evaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut :

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, metodologi pemecahan masalah, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori penunjang yang berhubungan dengan konsep dasar penjadwalan *job shop* dan algoritma semut yang akan digunakan untuk menyelesaikan permasalahan optimasi penjadwalan *job shop*.
3. BAB III METODOLOGI DAN PERANCANGAN SISTEM
   Pada bab ini akan dijelaskan mengenai langkah-langkah yang digunakan dalam optimasi penjadwalan *job shop* dengan menggunakan algoritma semut.
4. BAB IV IMPLEMENTASI DAN UJI COBA SISTEM
   Pada bab ini akan dilakukan implementasi sistem, pengujian dan analisa sistem perangat lunak yang dibangun.
5. BAB V PENUTUP
   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan untuk pengembangan.

BAB II
TINJAUAN PUSTAKA

2.1 Konsep Dasar Penjadwalan

Penjadwalan adalah pengurutan pembuatan produk secara menyeluruh yang dikerjakan pada beberapa buah mesin (Rosnani, 2009). Dengan demikian masalah *sequencing* senantiasa melibatkan pengerjaan sejumlah komponen yang sering disebut dengan istilah *job. Job* sendiri merupakan komposisi dari sejumlah elemen-elemen dasar yang disebut aktivitas atau operasi. Tiap aktivitas atau operasi ini membutuhkan alokasi sumber daya tertentu selama periode waktu tertentu yang sering disebut waktu proses.

2.1.1 Tujuan Penjadwalan

Menurut Bedworth (1987), beberapa tujuan dari aktivitas penjadwalan adalah sebagai berikut :

1. Meningkatkan penggunaan sumber daya atau mengurangi waktu tunggunya, sehingga total waktu proses dapat berkurang dan produktivitas dapat meningkat.
2. Mengurangi persediaan barang setengah jadi atau mengurangi sejumlah pekerjaan yang menunggu dalam antrian ketika sumber daya yang ada masih mengerjakan tugas yang lain.
3. Mengurangi beberapa kelambatan pada pekerjaan yang mempunyai batas waktu penyelesaian sehingga akan memperkecil *penalty cost* (biaya kelambatan).
4. Membantu pengambilan keputusan mengenai perencanaan kapasitas pabrik dan jenis kapasitas yang dibutuhkan sehingga penambahan biaya yang mahal dapat dihindarkan.

2.1.2 Model penjadwalan

Proses penjadwalan timbul jika terdapat keterbatasan sumber daya yang dimiliki sehingga diperlukan adanya pengaturan sumber-sumber daya secara efisien. Berbagai model penjadwalan telah dikembangkan untuk mengatasi persoalan penjadwalan tersebut.

Menurut Baker(1974), model penjadwalan dapat dibedakan menjadi 4 jenis keadaan, yaitu :

1. Mesin yang digunakan dapat berupa proses dengan mesin tunggal atau proses dengan mesin majemuk. Model mesin tunggal adalah mesin dasar dan biasanya dapat diterapkan pada kasus mesin

majemuk.
2. Pola aliran proses dapat berupa aliran identik (*flow shop*) atau aliran sembarang (*job shop*).
3. Pola kedatangan pekerjaan statis atau dinamis. Pada pola statis, pekerjaan datang bersamaan pada waktu nol dan siap dikerjakan atau kedatangan pekerjaan bisa tidak bersamaan tetapi saat kedatangan telah diketahui sejak waktu nol. Pada pola dinamis mempunyai sifat kedatangan pekerjaan tidak menentu, artinya terdapat variabel waktu sebagai faktor yang berpengaruh.
4. Sifat informasi yang diterima dapat bersifat deterministik atau stokastik. Model deterministik memiliki kepastian informasi tentang parameter dalam model, sedangkan model stokastik mengandung unsur ketidakpastian.

2.1.3 Input dan Output Penjadwalan

Pekerjaan-pekerjaan yang merupakan alokasi kapasitas untuk *order-order*, penugasan prioritas *job,* dan pengendalian jadwal produksi membutuhkan informasi terperinci, di mana informasi-informasi tersebut akan menyatakan input dari sistem penjadwalan. Kualitas dari keputusan-keputusan penjadwalan sangat dipengaruhi oleh ketetapan estimasi input-input tersebut.

Untuk memastikan bahwa suatu aliran kerja yang lancar pada saat melalui tahapan produksi, maka sistem penjadwalan harus membentuk aktivitas-aktivitas sebagai berikut :

1. Pembebanan (*loading*)
   Pembebanan melibatkan penyesuaian kebutuhan kapasitas untuk *order-order* yang diterima dengan kapasitas yang tersedia.
2. Pengurutan *(Sequencing)*
   Pengurutan merupakan penugasan tentang order-order mana yang diprioritaskan untuk diproses dahulu bila suatu fasilitas harus memproses banyak *job*.
3. Prioritas *job (dispaching)*
   Dispaching merupakan prioritas kerja tentang *job-job* mana yang diprioritaskan untuk diproses.
4. Pengendalian kinerja penjadwalan
   Pengendalian kinerja penjadwalan dilakukan dengan :
   a) Meninjau kembali status order-order pada saat melalui sistem tertentu.
   b) Mengatur kembali urutan-urutan, misalnya *expediting* order-

order yang jauh dibelakang atau mempunyai prioritas utama.

5. *Updating* jadwal
   Dilakukan sebagai refleksi kondisi operasi yang terjadi dengan merevisi prioritas-prioritas.

2.1.4 Jenis-jenis Penjadwalan

Menurut Rosnani (2009), terdapat 3 jenis penjadwalan berdasarkan sistem produksinya. Jenis penjadwalan tersebut antara lain :

1. Penjadwalan *flow shop*
   Karakteristik dari penjadwalan *flow shop* yaitu mempunyai pola aliran yang linier dan semua *job* memiliki urutan proses yang sama.
2. Penjadwalan *batch*
   Karakteristik dari penjadwalan batch yaitu mempunyai pola aliran yang tidak begitu rumit dan banyak terjadi perulangan.
3. Penjadwalan *job shop*
   Karakteristik dari penjadwalan *job shop* yaitu mempunyai pola aliran yang rumit dan setiap *job* memiliki urutan proses yang berbeda satu sama lain.

2.1.5 Penjadwalan *job shop*

Menurut Rosnani (2009), penjadwalan *job shop adalah* pengalokasian sejumlah *n* pekerjaan (*job*) yang harus diproses melalui sejumlah *m* mesin. Permasalahan pada penjadwalan *job shop* sangat sulit dibandingkan dengan penjadwalan *flow shop*. Hal ini disebabkan oleh 3 alasan, yaitu :

1. *job shop* menangani variasi produk yang sangat banyak, dengan pola aliran yang berbeda-beda melalui pusat-pusat kerja.
2. Peralatan pada *job shop* digunakan bersama-sama oleh bermacam-macam order pada prosesnya, sedangkan peralatan pada *flow shop* digunakan khusus untuk satu jenis produk.
3. *job-job* yang berbeda mungkin ditentukan oleh prioritas yang berbeda pula. Sedangkan pada *flow shop* tidak terjadi permasalahan seperti tersebut karena keseragaman output yang diproduksi.

Masalah *job shop* dengan *n job* dengan *m* mesin akan menghasilkan $(n!)^m$ buah kemungkinan solusi atau jadwal, namun tidak semua jadwal tersebut layak digunakan. Sebuah jadwal

dikatakan valid jika urutan pengerjaan operasi-operasi dalam suatu *job* mememenuhi *routing* yang telah ditetapkan dan tidak ada *overlap* waktu pengerjaan dari operasi-operasi yang dikerjakan pada mesin yang sama. Masalah *job shop* berhasil diselesaikan jika waktu awal pengerjaan dari setiap operasi telah diperoleh dan kedua syarat diatas terpenuhi.

Ketika terdapat *order-order* dari pada suatu *job shop* , kegiatan pertama dari penjadwalan adalah menugaskan *order-order* tersebut kepada bermacam-macam pusat kerja untuk diproses. Permasalahan *loading* menjadi lebih sederhana ketika suatu *job* tidak dapat dipisah. Untuk permasalahan yang sederhana dapat diasumsikan tidak ada pemisahan *job*, maka permasalahan *job shop* dapat dibuat dengan mudah dengan menggunakan *gantt chart* dan metode penugasan. Metode penugasan merupakan cara pembebanan pekerja-pekerja untuk *job-job* yang tersedia dengan tujuan meminimasi total waktu kerja dan total biaya kerja.

Setelah beberapa *job* telah ditugaskan pada pusat kerja tertentu, maka langkah berikutnya adalah menentukan urutan-urutan prosesnya. Pemrosesan order merupakan hal yang penting karena mempengaruhi lamanya suatu *job* akan diproses dalam sistem tertentu.

Penjadwalan *job shop* meliputi aturan-aturan prioritas *sequencing.* Aturan-aturan ini diaplikasikan untuk seluruh *job* yang sedang menunggu dalam antrian. Beberapa aturan *sequencing* yang umum antara lain sebagai berikut :

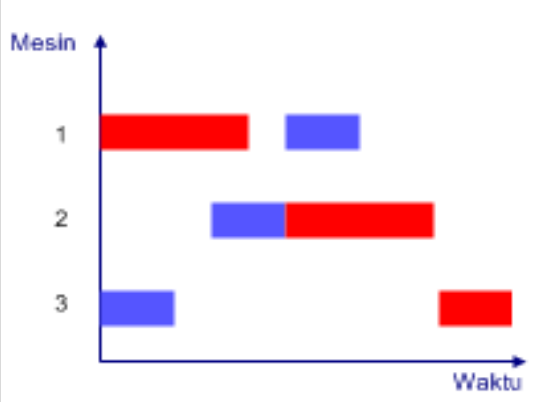
1. *First Come First Served* (FCFS)
   job yang datang diproses sesuai dengan *job* mana yang datang terlebih dahulu.
2. *Earliest Due Date* (EDD)
   Prioritas diberikan kepada *job-job* yang mempunyai tanggal batas waktu penyerahan *(due date)* paling awal.
3. *Shortest Processing Time* (SPT)
   job dengan waktu proses terpendek akan diproses lebih dahulu demikian berlanjut untuk *job* yang waktu prosesnya terpendek kedua. Aturan SPT ini tidak memperdulikan *due date* kedatangan order baru.

Penjadwalan yang optimal adalah apabila fungsi tujuan pada penjadwalan tercapai. Fungsi tujuan yang paling fundamental sekaligus digunakan untuk mengevaluasi penjadwalan *job shop*

adalah *makespan* (C_{maks}) yang didefinisikan sebagai (C_1,...,C_n). *Makespan* adalah waktu *job* terakhir yang keluar dari sistem. Fungsi tujuan yang lain adalah *Completion time* (C_i) yaitu waktu yang dibutuhkan untuk menyelesaikan *job* mulai dari saat tersedianya *job* (t=0) sampai pada *job* tersebut selesai dikerjakan.

2.1.6 *Gantt Chart*

Masalah penjadwalan sebenarnya masalah murni pengalokasian pekerjaan dan dengan bantuan model matematis akan dapat ditentukan solusi optimal. Sebagai alat bantu yang digunakan dalam menyelesaikan masalah penjadwalan dikenal satu model yang sederhana dan umum digunakan secara luas yakni *gantt chart.* Menurut Rosnani (2009), *gantt chart* merupakan grafik hubungan antara alokasi sumber daya dan waktu. Pada sumbu vertikal digambarkan jenis sumber daya yang digunakan dan sumbu horizontal digambarkan satuan waktu.



Gambar 2.1 *Gantt Chart*

Dari *gantt chart* kemudian ditentukan urutan (*sequence*) dari *job* yang memberikan kriteria penjadwalan terbaik, misalnya waktu pemrosesan tersingkat, utilitas mesin tertinggi, *idle time* minimum, dan lain-lain.

2.2 Algoritma Semut

Algoritma semut diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut (Dorigo, 1996). Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat sumber makanan. Koloni semut dapat menemukan

rute terpendek antara sarang dan tempat sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas jejak kakinya. Hal ini akan menyebabkan lintasan yang jejak kakinya sedikit, semakin lama akan semakin berkurang kepadatan semut yang melewatinya, atau bahkan tidak dilewati sama sekali. Sebaliknya lintasan yang jejak kakinya banyak, semakin lama akan semakin bertambah kepadatan semut yang melewatinya, atau bahkan semua semut akan melewati lintasan tersebut.

Dalam algoritma semut terdapat beberapa parameter masukan sebagai inisialisasi awal untuk melakukan proses optimasi. Beberapa parameter tersebut adalah :

n = banyak node yang ada
Q = konstanta jumlah *pheromone*
α = tetapan pengendali intensitas jejak semut
β = tetapan pengendali visibilitas
η_{ij} = visibilitas antar titik = $1/d_{ij}$
m = banyak semut
ρ = tetapan penguapan jejak semut
Nc_{max} = jumlah siklus maksimum

Dalam kasus TSP pertama kali setiap semut disebar pada beberapa node secara random sedangkan pada kasus *schedulling,* posisi awal semut terletak di posisi ke 0 yang selanjutnya semut-semut tersebut dibiarkan memilih rute perjalanannya. Untuk menjaga kelayakan dari solusi yang ditemukan dan juga merupakan mekanisme pemanfaatan informasi jejak, maka semut-semut diatur untuk memilih sebuah node berikutnya dengan probabilitas sesuai dengan persamaan 2.1.

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^{a}.[\frac{1}{d_{ij}}]^{\beta}}{\sum [\tau_{ij}(t)]^{a}.[\frac{1}{d_{ij}}]^{\beta}} \qquad (2.1)$$

Keterangan
τ_{ij} = intensitas *pheromone* pada jalur antara node $(job)_i$ dan $(job)_j$
d_{ij} = jarak heuristik antara node $(job)_i$ dan node $(job)_j$
p_{ij} = probabilitas jalur dari node $(job)_i$ dan node $(job)_j$

Jika probabilitas node terpilih yang dituju jumlahnya lebih besar dari nilai random yang dibangkitkan maka node tersebut dipilih sebagai node yang dikunjungi semut dan node tersebut akan disimpan sebagai node yang telah dikunjungi oleh semut, sehingga untuk menentukan node selanjutnya tidak akan dimasukkan dalam proses pencarian.

Setelah semua semut telah melalui semua node yang tersedia pada satu siklus, proses selanjutnya adalah melakukan pembaharuan jejak semut pada setiap jalur yang telah dilalui oleh semut dan jalur yang dipilih adalah jalur yang dilalui oleh semut dengan *makespan* terbaik. Persamaan 2.2 merupakan rumusan untuk melakukan perhitungan pembaharuan jejak semut.

$$\tau_{ij} baru = (1-\rho).\tau_{ij} lama + (\rho.\Delta\tau_{ij}) \quad (2.2)$$

Persamaan 2.3 digunakan untuk menghitung nilai $\Delta\tau_{ij}$ jika node i,j dilalui oleh semut sedangkan jika tidak maka nilai $\Delta\tau_{ij}$ adalah 0.

$$\Delta\tau_{ij} = \frac{Q}{makespan terbaik} \quad (2.3)$$

Keterangan

τ_{ij} baru = intensitas *pheromone* pada jalur antara node $(job)_i$ dan $(job)_j$ yang diperbaharui.

τ_{ij} lama = intensitas *pheromone* pada jalur antara node $(job)_i$ dan $(job)_j$ yang lama.

ρ = tetapan penguapan jejak semut.

$\Delta\tau_{ij}$ = jumlah *pheromone* yang ditambahkan pada jalur terbaik.

Q = konstanta jumlah *pheromone.*

Persamaan 2.2 dan persamaan 2.3 akan memberikan sejumlah jejak semut dari hasil node yang telah dikunjungi oleh semut. Semakin banyak semut yang melalui jalur tersebut maka akan berbanding lurus dengan jumlah jejak semut.

2.2.1 Pendekatan Algoritma Semut Pada Penjadwalan

Model yang digunakan untuk menotasikan masalah penjadwalan *job shop* pada n-*job* dan m-mesin adalah (van der Zwan dan Marques 1999) **N/m/G/C_{max}**. Dimana N mendefinisikan banyaknya *job* yang akan diproses, m menunjukkan banyaknya mesin yang dimiliki, C_{max} merupakan minimum *makespan time* dari suatu produksi dan G berisikan aturan urutan proses mesin untuk setiap *job* dan waktu prosesnya. Proses mesin dimodelkan dalam bentuk matriks T, misal untuk T dengan n=2 dan m=3 dapat dilihat pada gambar 2.2 :

$$T = \begin{bmatrix} M1 & M2 & M3 \\ M2 & M3 & M1 \end{bmatrix}$$

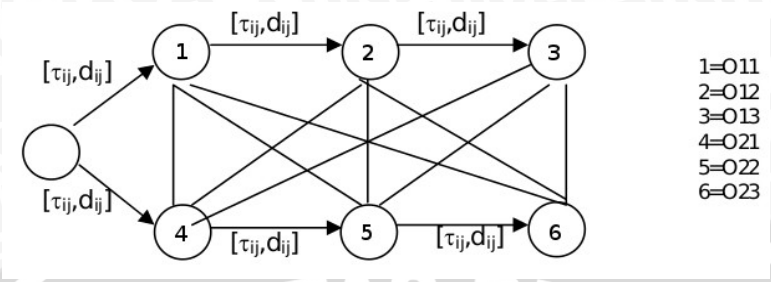
Gambar 2.2 Matriks T

Sedangkan untuk merepresentasikan waktu proses setiap operasi dimodelkan dengan matriks P :

$$p = \begin{bmatrix} t(O_{11}) & \dots & t(O_{1m}) \\ t(O_{21}) & \dots & t(O_{2m}) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ t(O_{n1}) & \dots & t(O_{nm}) \end{bmatrix}$$

Gambar 2.3 Matriks P

Untuk merepresentasikan *job shop* pada algoritma semut digunakan *graph* sebagai model. Tiap node dari graph ini yang nantinya akan digunakan oleh semut untuk membuat rute perjalanan sehingga akan mempermudah dalam pencarian minimum waktu penyelesaian. Model graph dari 2/3/G/C_{max} *job shop* dapat dilihat pada gambar 2.4



Gambar 2.4 *Graph* Penjadwalan *Job Shop*

Pada gambar 2.4 setiap node memiliki batasan node yang dapat dikunjungi karena terdapat beberapa garis satu arah maka suatu node tidak memiliki hak untuk melalui jalur yang berlawanan arah dengan tanda arah panah tersebut. Dengan demikian setiap operasi penentuan node tujuan harus terlebih dahulu memeriksa batasan node yang boleh dilalui. Selain itu pemilihan node tujuan yang layak harus difilter dengan batasan operasi yang harus terurut. Misalkan jika *job* ke-1 operasi ke-2 hanya boleh dipilih sebagai node tujuan jika *job* ke-1 operasi ke-1 telah dipilih atau dilalui oleh semut. Node yang telah dikunjungi oleh semut disimpan dalam suatu variabel yaitu $tabu_k$, sedangkan batasan node yang boleh dikunjungi disimpan dalam variabel S_m.

Terdapat beberapa pengembangan dan modifikasi pada algoritma semut yang dilakukan untuk menghindari hasil konvergen yang terlalu cepat. Menurut Kumar (2003) untuk menghindari hal tersebut, perlu dilakukan penambahan parameter p_0 pada proses algoritma semut dan nilai p_0 ini dihitung dengan menggunakan persamaan 2.4.

$$p_0 = \frac{\log(NC)}{\log(NC_{max})} \tag{2.4}$$

Pada persamaan 2.4 *NC* adalah nilai iterasi yang sedang berlangsung dan NC_{max} adalah nilai dari seluruh iterasi yang harus dilakukan. Nilai yang akan dihasilkan p_0 akan selalu berada diantara 0 sampai 1 ($0 <= p_0 <= 1$). Pada saat nilai *NC* rendah maka nilai p_0 masih mendekati nilai 0 dan akan meningkat sejalan dengan meningkatnya nilai NC. Nilai p_0 akan dibandingkan dengan nilai *p*

yang merupakan nilai yang dibangkitkan secara acak. Dengan meningkatnya nilai *NC* maka kemungkinan nilai $p > p_0$ akan menurun. Dari fakta tersebut terlihat jelas pada saat nilai *NC* rendah kemungkinan penentuan rute baru secara acak oleh semut akan sangat tinggi, sebaliknya dengan meningkatnya nilai *NC* kemungkinan semut untuk membuat rute baru secara acak akan menurun. Dengan cara ini maka konvergensi yang cepat pada awal proses algoritma semut bisa dihindari.

Langkah-langkah pemecahan masalah menggunakan algoritma semut seperti yang diungkapkan oleh Dorigo (1996) yang kemudian disesuaikan pada aplikasi penjadwalan *job shop* adalah :

1. Inisialisasi.
   a) Representasikan masalah penjadwalan *job shop* kedalam graph.
   b) Inisialisasi setiap parameter algoritma semut.
   c) Berikan nilai NC = 1
   d) Berikan nilai τ_{ij} awal dengan nilai bilangan positif yang sangat kecil.
   e) Berikan nilai $\Delta\tau_{ij} = 0$ pada setiap node.
   f) Berikan $tabu_m = \{\}$;
   g) Berikan nilai $p_0 = 0$
2. Jika $NC > NC_{max}$ lanjutkan ke langkah 8 jika tidak maka lanjutkan ke langkah 3.
3. Jika $m > m_{max}$ lanjutkan ke langkah 7 jika tidak maka lanjutkan ke langkah 4.
4. Jika $tabu_m$ penuh maka lanjutkan ke langkah 6 jika tidak maka lanjutkan ke langkah 5.
5. Pemilihan node yang akan dikunjungi.
   a) Masukkan node yang boleh dikunjungi dalam S_m.
   b) Bangkitkan nilai random p ($0 <= p <= 1$).
   c) Jika $p >= p_0$ maka lanjutkan ke langkah 5(d) jika tidak lanjutkan ke langkah 5(e).
   d) Pilih node secara random dalam S_m, lanjutkan ke langkah 5(k).
   e) Bandingkan nilai peluang tiap node yang didapat dengan persamaan 2.1 (p^m_{ij}).
   f) Pilih node dengan peluang terbesar.
   g) Bangkitkan nilai acak *q* ($0 <= q <= 1$).
   h) Jika nilai $q >= p^m_{ij}$ maka lanjutkan ke langkah 5(i) jika tidak lanjutkan ke langkah 5 (j).
   i) Pilih node secara random dalam S_m, lanjutkan ke langkah 5(k).

j) Pilih node dengan nilai p^m_{ij} terbesar .
k) Gunakan node pilih sebagai node yang akan dikunjungi.
l) Masukkan node pilih ke dalam $tabu^k$, hapus dari G_k dan S_k, lalu kembali ke langkah 4.

6. m = m + 1 (pilih semut selanjutnya), kembali ke langkah 3.
7. Pembaharuan Jejak
   a) Cari *makespan* terbaik tiap siklus.
   b) Bandingkan dengan *makespan* terbaik global, tentukan *makespan* terbaik..
   c) Perbarui jejak semut τ_{ij} dengan persamaan 2.2.
   d) Kosongkan semua *tabulist* semut.
   e) NC = NC +1
   f) p_0 = log(NC) / log(NC_max), kembali ke langkah 2.
8. Keluaran berupa *makespan* terbaik semua siklus.

Keterangan

NC	= Iterasi siklus yang sedang berlangsung.
NC_{max}	= jumlah total iterasi siklus.
S_m	= daftar node yang boleh dikunjugi semut m.
$tabu_m$	= daftar node yang telah dikunjungi semut ke m.
m	= Semut yang sedang melakukan perjalanan.
m_{max}	= Jumlah total semut.
p_0	= parameter untuk menghindari konvergen terlalu cepat.
p	= bilangan acak pembanding p_0.
p^m_{ij}	= peluang dari node i ke node j oleh semut m.
q	= bilangan acak pembanding p^m_{ij}.

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan sistem ini akan dibahas tahapan-tahapan yang dilakukan dalam pembuatan aplikasi penjadwalan *job shop* menggunakan algoritma semut. Tahapan-tahapan tersebut antara lain :

1. Mencari dan mempelajari literatur-literatur yang berkaitan dengan metode algoritma semut yang digunakan dalam pembuatan aplikasi penjadwalan *job shop.*
2. Melakukan analisa dan menerapkan metode algoritma semut untuk masalah penjadwalan *job shop* seperti yang dijelaskan pada bab II.
3. Membuat aplikasi penjadwalan *job shop* dengan menggunakan metode algoritma semut sesuai dengan analisis yang telah dilakukan.
4. Melakukan ujicoba terhadap aplikasi penjadwalan *job shop* tersebut.
5. Melakukan evaluasi terhadap penerapan algoritma semut pada aplikasi penjadwalan job shop dengan menggunakan data *job shop* dari *OR-Library yaitu Fisher and Thompson 6x6 instance* yang berisi data 6 job dan 6 mesin.

3.1. Analisis Sistem

3.1.1 Deskripsi Sistem

Sistem yang akan dibuat bertujuan untuk mengimplementasikan dan mengetahui kinerja dari metode algoritma semut dalam masalah penjadwalan *job shop* agar dapat menghasilkan *makespan* minimum yang merupakan salah satu kriteria untuk pemilihan penjadwalan yang baik.

3.1.2 Metode Analisis

Untuk melihat proses aplikasi yang mencakup proses input dan proses output akan digunakan diagram alir (*flowchart*). Pada tahap ini akan digunakan notasi-notasi untuk menggambarkan arus data dimana akan sangat membantu dalam proses komunikasi dengan pemakai.

3.1.3 Analisis Kebutuhan Sistem

Dari mempelajari dan menganalisis literatur-literatur yang berhubungan dengan algoritma semut dan penjadwalan *job shop* yang terdiri dari analisis kebutuhan proses, analisis kebutuhan masukan, dan analisis kebutuhan keluaran dapat diperoleh beberapa data pada masing-masing analisis yaitu:

3.1.3.1 Analisis Kebutuhan Proses

Kebutuhan proses dalam sistem penjadwalan *job shop* menggunakan algoritma semut antara lain :

1. Proses merepresentasikan permasalahan penjadwalan job shop ke dalam bentuk graph.
2. Proses penentuan rute perjalanan tiap-tiap semut pada node-node yang ada dalam *graph.*
3. Proses perhitungan *makespan* yang dihasilkan masing-masing semut pada setiap siklus berdasarkan jarak yang ditempuh.
4. Proses pemilihan *makespan* yang minimum.

3.1.3.2 Analisis Kebutuhan Masukan

Masukan untuk aplikasi penjadwalan *job shop* ini berupa data-data yang diperlukan untuk diproses dengan algoritma semut, data tersebut adalah :

1. Data penjadwalan yang berupa urutan mesin dan waktu operasi pada setiap job.
2. Parameter – parameter yang diperlukan dalam perhitungan algoritma semut yaitu :
   a) Intensitas jejak semut antar node (τ_{ij}) dan perubahannya ($\Delta\tau_{ij}$).
      τ_{ij} harus diinisialisasi sebelum memulai siklus. τ_{ij} digunakan dalam persamaan probabilitas node yang akan dikunjungi. $\Delta\tau_{ij}$ diinisialisaikan setelah satu siklus dan digunakan untuk menentukan siklus $\Delta\tau_{ij}$ selanjutnya.
   b) Tetapan siklus semut (Q)
      Q merupakan konstanta yang digunakan dalam persamaan untuk menentukan $\Delta\tau_{ij}$. Nilai Q ditentukan oleh pengguna.
   c) Tetapan pengendali intensitas jejak semut (α)
      α digunakan dalam persamaan probabilitas node yang akan dikunjungi dan berfungsi sebagai pengendali intensitas jejak semut. Nilai α ditentukan oleh pengguna.
   d) Tetapan pengendali visibilitas antar node (β)
      β digunakan dalam persamaan probabilitas node yang akan

dikunjungi dan berfungsi sebagai pengendali visibilitas antar node. Nilai β ditentukan oleh pengguna.

e) Visibilitas antar node (η_{ij}).
   η_{ij} digunakan dalam persamaan probabilitas kota yang akan dikunjungi. Nilai η_{ij} merupakan hasil dari $1/d_{ij}$ (waktu operasi).

f) Banyak semut (m)
   m merupkan banyak semut yang akan melakukan siklus dalam algoritma semut. Nilai m ditentukan oleh pengguna.

g) Tetapan penguapan jejak semut (ρ)
   ρ digunakan untuk menentukan τ_{ij} untuk siklus selanjutnya. Nilai ρ ditentukan oleh pengguna.

h) Jumlah siklus maksimum (NC_{max}) adalah jumlah siklus yang akan berlangsung. Siklus akan berhenti sesuai dengan NC_{max} yang telah ditentukan atau telah terjadi keadaan yang konvergen. Nilai NC_{max} ditentukan oleh pengguna.

3.1.3.3 Analisis Kebutuhan Keluaran

Data keluaran yang diperoleh dari proses pada aplikasi penjadwalan *job shop* ini adalah *makespan* yang merupakan total waktu penyelesaian *job-job* mulai dari urutan pertama sampai kepada urutan pekerjaan terakhir pada mesin. *Makespan* ini digunakan untuk menentukan susunan dari *job* yang terbaik dari suatu kasus penjadwalan.

3.1.4 Analisis Perancangan Antarmuka

Pada perancangan antarmuka akan digunakan Lazarus IDE. karena dengan menggunakan antarmuka yang berbasis grafis akan lebih memudahkan pengguna untuk menggunakan aplikasi ini. Selain itu untuk menghilangkan kesulitan dari pengetikan perintah-perintah yang biasanya digunakan dalam aplikasi yang menggunakan antarmuka *command line.*

3.2 Perancangan Sistem

Untuk menerapkan algoritma semut pada sistem penjadwalan *job shop* ini terdapat beberapa proses yang harus dilakukan sehingga bisa didapatkan keluaran berupa penjadwalan terbaik. Gambaran proses penjadwalan *job shop* menggunakan algoritma semut secara umum dapat dilihat pada gambar 3.1.

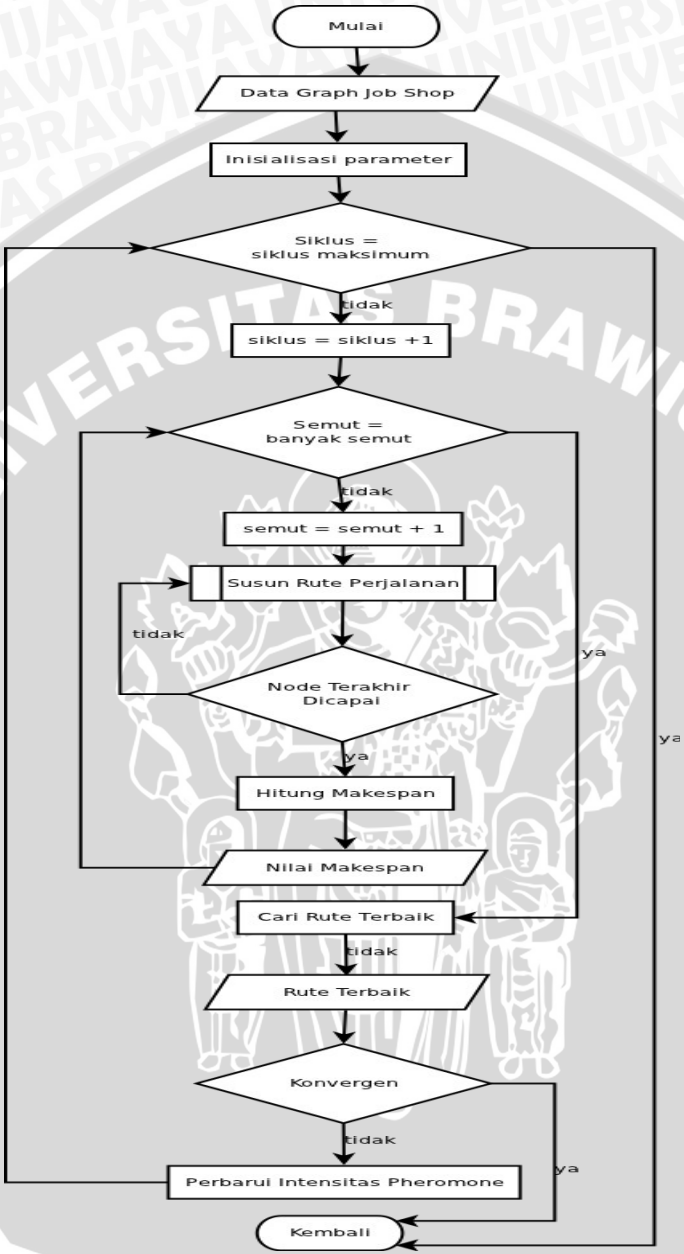
Gambar 3.1 Proses Penjadwalan

Penjelasan dari *flowchart* pada gambar 3.1 adalah sebagai berikut :

1. Memasukkan data penjadwalan untuk sistem yang berupa data jumlah *job*, urutan mesin, dan data waktu proses tiap mesin pada setiap operasi.
2. Pengolahan data penjadwalan tersebut menggunakan Algoritma semut untuk menghasilkan *makespan* yang selanjutnya akan dibandingkan untuk diperoleh *makespan* yang minimum.
3. Menghasilkan keluaran berupa urutan *job* yang baik untuk diterapkan pada penjadwalan.

3.2.1 Penjadwalan *Job Shop* Menggunakan Algoritma Semut

Tujuan digunakannya algoritma semut pada penjadwalan *job shop* ini adalah untuk menemukan *makespan* yang minimum berdasarkan jarak total yang ditempuh semut pada saat melewati semua node. Dengan tujuan tersebut maka dibuat sebuah *flowchart* untuk menjelaskan proses yang terjadi pada algoritma semut untuk masalah penjadwalan seperti pada gambar 3.2



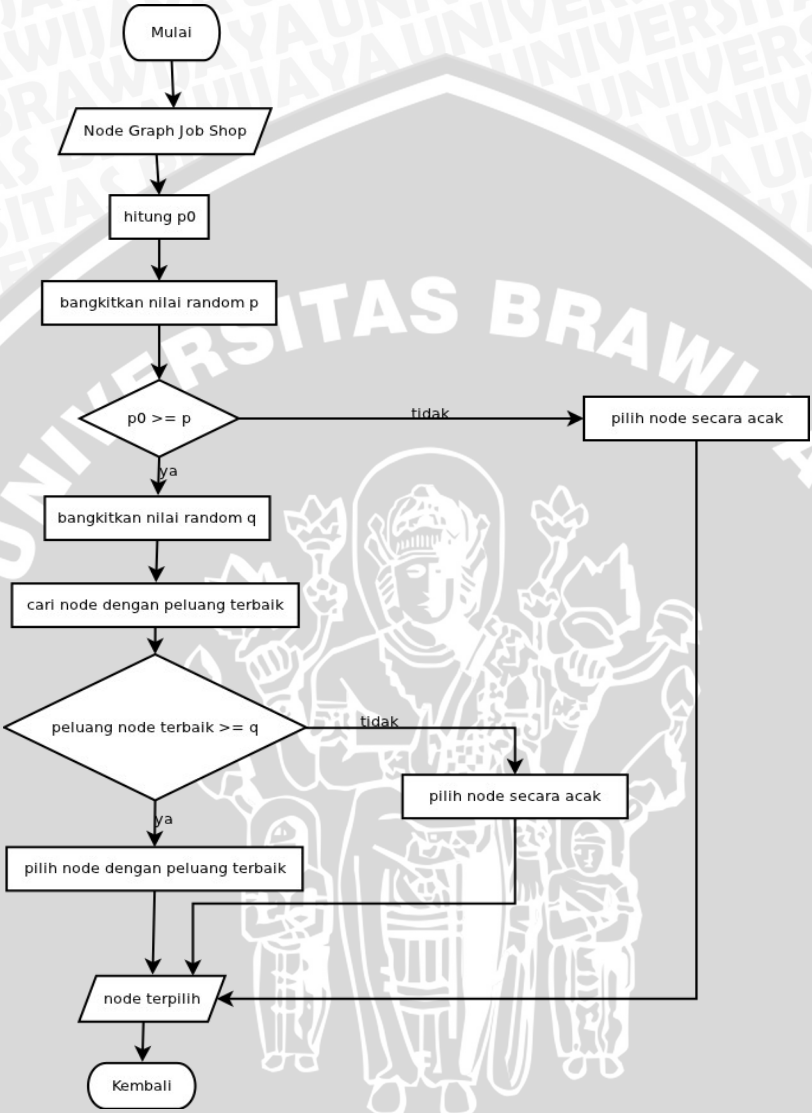
Gambar 3.2 Proses Algoritma Semut.

Penjelasan dari *flowchart* proses algoritma semut pada gambar 3.2 adalah sebagai berikut.

1. Inisialisasi awal parameter-parameter yang dibutuhkan oleh algoritma semut untuk memproses data yang telah dimasukkan.
2. Pengecekan jumlah siklus jika belum mencapai siklus maksimum maka siklus akan bertambah dan dilanjutkan.
3. Pengecekan jumlah semut jika semut belum mencapai jumlah semut maksimum maka semut akan bertambah.
4. Penyusunan rute kunjungan setiap semut ke semua node operasi yang akan dijelaskan pada sub bab 3.2.2.
5. Setelah semut selesai mengunjungi semua node dilakukan penghitung *makespan*.
6. Setelah semua semut selesai melakukan perjalanan dilakukan pemilihan rute dengan *makespan* terbaik.
7. Perbarui intensitas *pheromone* (jejak kaki semut) pada node rute terbaik untuk digunakan pada siklus selanjutnya.
8. Pada setiap siklus akan ditentukan *makespan* yang terbaik dari tiap semua semut yang akan dibandingkan dengan *makespan* terbaik seluruh siklus. Hasil terbaik dari perbandingan tersebut dijadikan keluaran dari sistem. Perhitungan algoritma semut ini akan berhenti jika siklus maksimum sudah tercapai atau *makespan* terbaik yang dihasilkan sudah konvergen.

3.2.2 Penyusunan Rute Kunjungan

Semut akan mulai melakukan perjalanan dari titik pertama yaitu O menuju titik berikutnya sebagai tujuan. Kemudian perjalanan semut akan dilakukan dilakukan dengan pemilihan titik yang tidak terdapat pada $tabu_m$. Untuk masalah penjadwalan diperlukan suatu variabel untuk menampung node tujuan yaitu S_m yang boleh dikunjungi oleh semut saat berada pada suatu node tertentu agar operasi-operasi pada tiap *job* bisa dijadwalkan secara terurut. *Flowchart* untuk proses peyusunan rute kunjungan bisa dilihat pada gambar 3.2.



Gambar 3.3 Penyusunan Rute Perjalanan

Penjelasan dari *flowchart* proses pemilihan node pada gambar 3.3 adalah sebagai berikut.

1. Penghitungan nilai p_0 yang dilakukan dengan menggunakan persamaan 2.4

2. Melakukan perbandingan nilai p_0 yang dihasilkan dengan bilangan acak p yang dibangkitkan.
3. Jika nilai p_0 lebih kecil dari p maka akan dipilih node secara acak sebagai node tujuan selanjutnya.
4. Jika nilai p_0 lebih besar dari nilai p maka akan dipilih probabilitas terbaik dari tiap node yang berada di S_k. Nilai probabilitas tersebut dihitung dengan persamaan 2.1.
5. Selanjutnya nilai probabilitas terbaik tersebut akan dibandingkan dengan bilangan acak q yang dibangkitkan.
6. Jika nilai probabilitas terbaik lebih besar dari q maka node dengan probabilitas tersebut akan dipilih sebagi node selanjutnya.
7. Jika nilai probablitas lebih kecil dari nilai q maka akan dipilih node secara acak sebagai node tujuan selanjutnya.

3.3 Contoh Proses Penjadwalan Menggunakan Algoritma Semut

Pada sub bab ini akan ditunjukkan contoh pemrosesan data penjadwalan yang dilakukan dengan menggunakan algoritma semut sehingga dapat menghasilkan urutan *job* yang baik dan menghasilkan *makespan* yang minimum. Data yang akan diproses adalah contoh data *job shop* yang terdiri dari 3 *job* dan 3 mesin. Untuk lebih jelasnya data-data tersebut ditampilkan pada tabel 3.1 dan tabel 3.2.

Tabel 3.1 Data Urutan Mesin

JOB	PROSES		
	1	**2**	**3**
1	M1	M2	M3
2	M2	M3	M1
3	M3	M1	M2

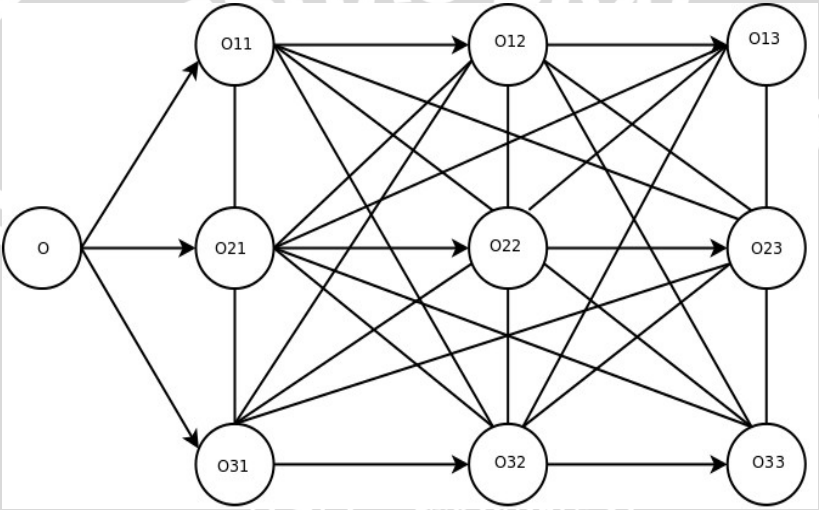
Tabel 3.2 Data Waktu Proses

JOB	PROSES		
	1	**2**	**3**
1	10	3	7
2	5	8	4
3	6	9	2

3.3.1 Penentuan Rute Kunjungan Semut

Seperti disebutkan pada sub bab 3.1.3.2 bahwa selain data *job shop,* juga diperlukan inisialisasi parameter awal yang dibutuhkan oleh algoritma semut untuk proses perhitungan. Dan parameter awal yang digunakan adalah $\alpha = 1$, $\beta = 2$, $\rho = 0{,}5$, T_{ij} Awal = 0,01, $NC_{max} = 3$, $Q = 1$, $m = 3$.

Untuk melakukan penentuan rute kunjungan masing-masing semut, data penjadwalan pada tabel 3.1 dan tabel 3.2 harus direpresentasikan terlebih dahulu dalam bentuk *graph* seperti pada gambar 3.4.



Gambar 3.4 Representasi Graph Data *Job Shop*

Untuk nilai waktu tiap operasi dalam *job* pada gambar 3.4 dapat dilihat pada tabel 3.3 yang tiap kolom dan barisnya merepresentasikan node dan garis hubungan antar node pada gambar 3.4

Tabel 3.3 Data Waktu Operasi Dalam *Graph*

	O	O11	O12	O13	O21	O22	O23	O31	O32	O33
O	0	10	3	7	5	8	4	6	9	2
O11	0	0	3	7	5	8	4	6	9	2
O12	0	10	0	7	5	8	4	6	9	2
O13	0	10	3	0	5	8	4	6	9	2
O21	0	10	3	7	0	8	4	6	9	2
O22	0	10	3	7	5	0	4	6	9	2
O23	0	10	3	7	5	8	0	6	9	2
O31	0	10	3	7	5	8	4	0	9	2
O32	0	10	3	7	5	8	4	6	0	2
O33	0	10	3	7	5	8	4	6	9	0

Dari nilai waktu operasi pada tabel 3.3 dapat dihitung visibilitas antar node (η_{ij}) dengan rumus $1/d_{ij}$ yang nantinya akan digunakan dalam perhitungan peluang node sesuai dengan persamaan 2.1. Hasil perhitungan visibilitas antar node dapat dilihat pada tabel 3.4

Tabel 3.4 Visibilitas Antar Node

	O	O11	O12	O13	O21	O22	O23	O31	O32	O33
O	0	0,1	0,333	0,14	0,2	0,125	0,25	0,167	0,111	0,5
O11	0	0	0,333	0,14	0,2	0,125	0,25	0,167	0,111	0,5
O12	0	0,1	0	0,14	0,2	0,125	0,25	0,167	0,111	0,5
O13	0	0,1	0,333	0	0,2	0,125	0,25	0,167	0,111	0,5
O21	0	0,1	0,333	0,14	0	0,125	0,25	0,167	0,111	0,5
O22	0	0,1	0,333	0,14	0,2	0	0,25	0,167	0,111	0,5
O23	0	0,1	0,333	0,14	0,2	0,130	0	0,167	0,111	0,5
O31	0	0,1	0,333	0,14	0,2	0,125	0,25	0	0,111	0,5
O32	0	0,1	0,333	0,14	0,2	0,125	0,25	0,167	0	0,5
O33	0	0,1	0,333	0,14	0,2	0,125	0,25	0,167	0,111	0

Setelah itu siklus pertama pada algoritma semut ini dimulai dengan diawali oleh penentuan rute yang akan dilalui tiap semut

berdasarkan nilai peluang yang dihitung dengan persamaan 2.1 yang hasilnya akan dipilih peluang terbaik dan dibandingkan dengan nilai yang dibangkitkan secara acak. Jika nilai node yang dipilih lebih besar dari bilangan acak akan maka node tersebut dipilih sebagai node tujuan hal ini dapat dilakukan dengan syarat nilai p_0 yang dihitung dengan persamaan 2.6 lebih besar dari nilai p yang merupakan nilai yang dibangkitkan secara acak. Jika kedua syarat tersebut tidak terpenuhi maka akan dilakukan pemilihan node dalam S_1 secara acak. Berikut ini adalah contoh proses pemilihan rute oleh semut ke 1 pada siklus yang pertama.

1. Node = O
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{11},O_{21},O_{31}\}$
   $Tabu_1 = \{O\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0{,}01^1 * 0{,}1^2) + (0{,}01^1 * 0{,}2^2) + (0{,}01^1 * 0{,}167^2)$
   $= 0{,}00077889$
   $prob(O_{11}) = (0{,}01^1 * 0{,}1^2) / 0{,}00077889 = 0{,}128$ (indeks = 1)
   $prob(O_{21}) = (0{,}01^1 * 0{,}2^2) / 0{,}00077889 = 0{,}514$ (indeks = 2)
   $prob(O_{31}) = (0{,}01^1 * 0{,}167^2) / 0{,}00077889 = 0{,}357$ (indeks = 3)
   $p_0 = \log(1) / \log(3) = 0$
   $p = 0{,}617$
   $p_0 <= p$: pilih node secara random
   Random indeks pilih = 2 (indeks dari node dalam S_1)
   sehingga node yang dipilih adalah O_{21}

2. Node = O_{21}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{11},O_{22},O_{31}\}$
   $Tabu_1 = \{O,O_{21}\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0{,}01^1 * 0{,}1^2) + (0{,}01^1 * 0{,}125^2) + (0{,}01^1 * 0{,}167^2)$
   $= 0{,}00053514$
   $prob(O_{11}) = (0{,}01^1 * 0{,}1^2) / 0{,}00053514 = 0{,}187$ (indeks = 1)
   $prob(O_{22}) = (0{,}01^1 * 0{,}125^2) / 0{,}00053514 = 0{,}292$ (indeks = 2)
   $prob(O_{31}) = (0{,}01^1 * 0{,}167^2) / 0{,}00053514 = 0{,}519$ (indeks = 3)
   $p_0 = \log(1) / \log(3) = 0$
   $p = 0{,}965$
   $p_0 <= p$: pilih node secara random
   Random indeks pilih = 1 (indeks dari node dalam S_1)

sehingga node yang dipilih adalah O_{11}

3. Node = O_{11}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{12},O_{22},O_{31}\}$
   $Tabu_1 = \{O,O_{21},O_{11}\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0,01^1 * 0,333^2) + (0,01^1 * 0,125^2) + (0,01^1 * 0,167^2)$
   $= 0,00152414$
   $prob(O_{12}) = (0,01^1 * 0,333^2) / 0,00152414 = 0,729$ (indeks = 1)
   $prob(O_{22}) = (0,01^1 * 0,125^2) / 0,00152414 = 0,103$ (indeks = 2)
   $prob(O_{31}) = (0,01^1 * 0,167^2) / 0,00152414 = 0,182$ (indeks = 3)
   $p_0 = \log (1) / \log (3) = 0$
   $p = 0,201$
   $p_0 <= p$: pilih node secara random
   Random indeks pilih = 2 (indeks dari node dalam S_1)
   sehingga node yang dipilih adalah O_{22}

4. Node = O_{22}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{12},O_{23},O_{31}\}$
   $Tabu_1 = \{O,O_{21},O_{11},O_{22}\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0,01^1 * 0,333^2) + (0,01^1 * 0,25^2) + (0,01^1 * 0,167^2)$
   $= 0,00201278$
   $prob(O_{12}) = (0,01^1 * 0,333^2) / 0,00201278 = 0,552$ (indeks = 1)
   $prob(O_{23}) = (0,01^1 * 0,25^2) / 0,00201278 = 0,311$ (indeks = 2)
   $prob(O_{31}) = (0,01^1 * 0,167^2) / 0,00201278 = 0,138$ (indeks = 3)
   $p_0 = \log (1) / \log (3) = 0$
   $p = 0,611$
   $p_0 <= p$: pilih node secara random
   Random indeks pilih = 3 (indeks dari node dalam S_1)
   sehingga node yang dipilih adalah O_{31}

5. Node = O_{31}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{12},O_{23},O_{32}\}$
   $Tabu_1 = \{O,O_{21},O_{11},O_{22},O_{31}\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0,01^1 * 0,333^2) + (0,01^1 * 0,25^2) + (0,01^1 * 0,111^2)$
   $= 0,0018571$
   $prob(O_{12}) = (0,01^1 * 0,333^2) / 0,0018571 = 0,598$ (indeks = 1)

$prob(O_{23}) = (0{,}01^1 * 0{,}25^2) / 0{,}0018571 = 0{,}337$ (indeks = 2)
$prob(O_{32}) = (0{,}01^1 * 0{,}111^2) / 0{,}0018571 = 0{,}066$ (indeks = 3)
$p_0 = \log(1) / \log(3) = 0$
$p = 0{,}805$
$p_0 <= p$: pilih node secara random
Random indeks pilih = 2 (indeks dari node dalam S_1)
sehingga node yang dipilih adalah O_{23}

6. Node = O_{23}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{12},O_{32}\}$
   $Tabu_1 = \{O,O_{21},O_{11},O_{22},O_{31},O_{23}\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0{,}01^1 * 0{,}333^2) + (0{,}01^1 * 0{,}111^2)$
   $= 0{,}0012321$
   $prob(O_{12}) = (0{,}01^1 * 0{,}333^2) / 0{,}0012321 = 0{,}902$ (indeks = 1)
   $prob(O_{32}) = (0{,}01^1 * 0{,}111^2) / 0{,}0012321 = 0{,}1$ (indeks = 2)
   $p_0 = \log(1) / \log(3) = 0$
   $p = 0{,}949$
   $p_0 <= p$: pilih node secara random
   Random indeks pilih = 1 (indeks dari node dalam S_1)
   sehingga node yang dipilih adalah O_{12}

7. Node = O_{12}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{13},O_{32}\}$
   $Tabu_1 = \{O,O_{21},O_{11},O_{22},O_{31},O_{23},O_{12}\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0{,}01^1 * 0{,}143^2) + (0{,}01^1 * 0{,}111^2)$
   $= 0{,}0003277$
   $prob(O_{13}) = (0{,}01^1 * 0{,}143^2) / 0{,}0003277 = 0{,}623$ (indeks = 1)
   $prob(O_{32}) = (0{,}01^1 * 0{,}111^2) / 0{,}0003277 = 0{,}377$ (indeks = 2)
   $p_0 = \log(1) / \log(3) = 0$
   $p = 0{,}69$
   $p_0 <= p$: pilih node secara random
   Random indeks pilih = 1 (indeks dari node dalam S_1)
   sehingga node yang dipilih adalah O_{13}
8. Node = O_{13}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{32}\}$
   $Tabu_1 = \{O,O_{21},O_{11},O_{22},O_{31},O_{23},O_{12},O_{13}\}$

$\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0{,}01^1 * 0{,}111^2) = 0{,}00012321$
$prob(O_{32}) = (0{,}01^1 * 0{,}111^2) / 0{,}00012321 = 1{,}00$ (indeks = 1)
$p_0 = \log(1) / \log(3) = 0$
$p = 0{,}127$
$p_0 <= p$: pilih node secara random
Random indeks pilih = 1 (indeks dari node dalam S_1)
sehingga node yang dipilih adalah O_{32}

9. Node = O_{32}
   $G_1 = \{O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}\}$
   $S_1 = \{O_{33}\}$
   $Tabu_1 = \{O,O_{21},O_{11},O_{22},O_{31},O_{23},O_{12},O_{13},O_{32}\}$
   $\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta} = (0{,}01^1 * 0{,}5^2) = 0{,}0025$
   $prob(O_33) = (0{,}01^1 * 0{,}5^2) / 0{,}0025 = 1{,}00$ (indeks = 1)
   $p_0 = \log(1) / \log(3) = 0$
   $p = 0{,}427$
   $p_0 <= p$: pilih node secara random
   Random indeks pilih = 1 (indeks dari node dalam S_1)
   sehingga node yang dipilih adalah O_{33}

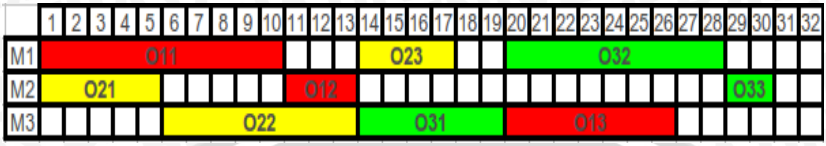
   $Tabu_1 = \{O,O_{21},O_{11},O_{22},O_{31},O_{23},O_{12},O_{13},,O_{32},O_{33}\}$

Setelah *tabu list* penuh maka sudah didapatkan urutan *job* yang didapatkan oleh semut ke-1 pada siklus ke-1. Urutan job dan waktu proses yang didapatkan oleh semut ke-1 pada siklus ke-1 ini seperti dalam tabel 3.5.

Tabel 3.5 Urutan *Job* dan Waktu Proses Semut Ke-1

Job	O_{21}	O_{11}	O_{22}	O_{31}	O_{23}	O_{12}	O_{13}	O_{32}	O_{33}
waktu	5	10	7	6	4	3	7	9	2

Dari data pada tabel 3.5 dapat dibuat sebuah *gantt chart* untuk mengetahui nilai *makespan* yang dihasilkan. Gambar 3.5 merupakan *gantt chart* dari proses ini.



Gambar 3.5 *Gantt Chart* Urutan *Job* Oleh Semut Ke-1

Dari gambar 3.5 dapat diketahui waktu *start* dan *finish* proses mesin tiap-tiap operasi pada semua *job* oleh semut 1. Waktu *start* dan *finish tersebut* ditunjukkan pada tabel 3.6

Tabel 3.6 Waktu *Start* dan *Finish* Mesin Oleh Semut Ke-1

Mesin 1	**Job**	**1**	**2**	**3**
	Operasi	**1**	**3**	**2**
	Start	1	14	20
	Finish	10	17	28
Mesin 2	**Job**	**2**	**1**	**3**
	Operasi	**1**	**2**	**3**
	Start	1	11	29
	Finish	5	13	30
Mesin 3	**Job**	**2**	**3**	**1**
	Operasi	**2**	**1**	**3**
	Start	6	14	20
	Finish	13	19	26

Pada tabel 3.6 waktu proses masing-masing operasi didapatkan dengan mencocokkan nama *job* dan operasi ke- pada tabel 3.5 Operasi pertama yang dijadwalkan adalah *job* ke-2 pada mesin ke-2 dengan waktu proses 5 selanjutnya penjadwalan akan dilakukan sampai pada waktu operasi terakhir yaitu job ketiga pada mesin ke-1 dan waktu finish yang didapatkan sebesar 30 yang merupakan *makespan* dari semut ke-1 pada siklus ke-1.

Selanjutnya langkah yang sama akan dilakukan oleh semut ke-2 sampai semut ke-3. Setelah semut ke-3 selesai melakukan perjalannya maka selanjutnya akan dilakukan pemilihan *makespan*

terbaik yang dihasilkan masing-masing semut pada siklus ke-1 dan akan digunakan sebagai salah satu parameter untuk melakukan pembaharuan jejak semut pada setiap node yang telah dilewati. Pada tabel 3.7 ini adalah data rute perjalanan semua semut pada siklus ke-1 dan nilai *makespan* yang dihasilkan dan *makespan* terbaik untuk siklus ke-1 ini adalah 29 yang dihasilkan oleh semut ke-3.

Tabel 3.7 Rute Perjalanan Semut dan Nilai *Makespan*

Semut	Rute	*Makespan*
1	$O,O_{21},O_{11},O_{22},O_{31},O_{23},O_{12},O_{13},O_{32},,O_{33}$	30
2	$O,O_{11},O_{21},O_{22},O_{31},O_{12},O_{23},O_{13},O_{32},,O_{33}$	30
3	$O,O_{31},O_{11},O_{21},O_{12},O_{22},O_{23},O_{32},O_{13},,O_{33}$	**29**

Proses pemilihan node pada siklus awal oleh setiap semut semuanya dilakukan secara acak hal ini dikarenakan setiap semut belum memiliki pengetahuan tentang jalur yang akan dilauinya. Dengan meningkatnya jumlah siklus maka kemungkinan semut memilih jalur secara acak akan berkurang.

3.3.2 Perhitungan Pembaharuan Jejak Semut

Setelah semua semut melakukan perjalanannya pada satu siklus maka selanjutnya akan dilakukan perhitungan pembaharuan jejak dari semut yang menghasilkan *makespan* terbaik menggunakan persamaan 2.2 dan persamaan 2.3 pada setiap node yang dilewatinya. Berikut ini contoh perhitungan pembaharuan jejak pada siklus ke-1

Rute terbaik = $O,O_{31},O_{11},O_{21},O_{12},O_{22},O_{23},O_{32},O_{13},,O_{33}$

1. Jejak dari node O ke node O_{31}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

2. Jejak dari node O_{31} ke node O_{11}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

3. Jejak dari node O_{11} ke node O_{21}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

4. Jejak dari node O_{21} ke node O_{12}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

5. Jejak dari node O_{12} ke node O_{22}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

6. Jejak dari node O_{22} ke node O_{23}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

7. Jejak dari node O_{23} ke node O_{32}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

8. Jejak dari node O_{32} ke node O_{13}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

9. Jejak dari node O_{13} ke node O_{33}
   τ_{ij} lama = 0,01
   $\Delta\tau_{ij}$ = 1/29 = 0,034
   τ_{ij} baru = (1-0,5)*0,01 + (0,5*0,034) = 0,022

Perubahan jejak semut setelah dilakukan pembaharuan dapat dilihat pada tabel 3.8

Tabel 3.8 Data Pembaharuan Jejak Semut Pada Siklus 1

	O	**O11**	**O12**	**O13**	**O21**	**O22**	**O23**	**O31**	**O32**	**O33**

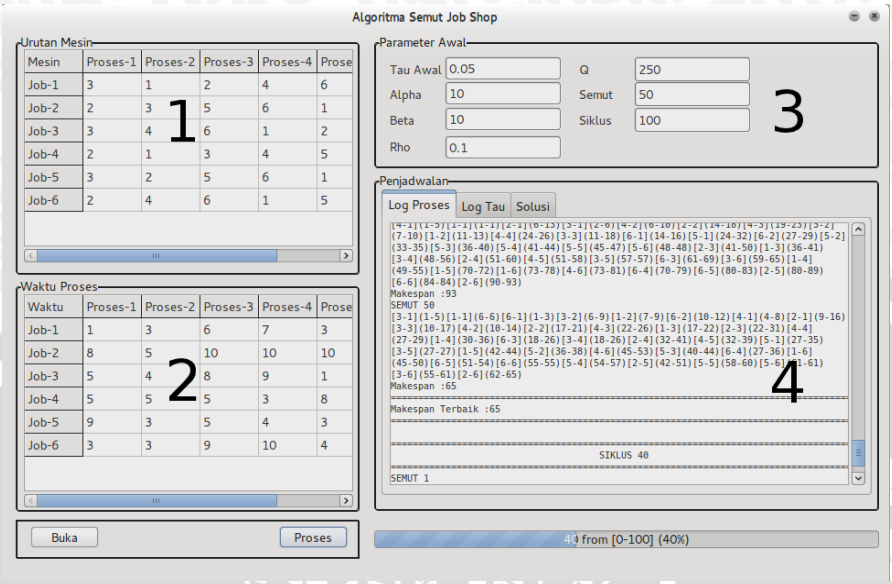
O	0,005	0,005	0,005	0,005	0,005	0,005	0,005	**0,022**	0,005	0,005
O11	0,005	0,005	0,005	0,005	**0,022**	0,005	0,005	0,005	0,005	0,005
O12	0,005	0,005	0,005	0,005	0,005	**0,022**	0,005	0,005	0,005	0,005
O13	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	**0,022**
O21	0,005	0,005	**0,022**	0,005	0,005	0,005	0,005	0,005	0,005	0,005
O22	0,005	0,005	0,005	0,005	0,005	0,005	**0,022**	0,005	0,005	0,005
O23	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	**0,022**	0,005
O31	0,005	**0,022**	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005
O32	0,005	0,005	0,005	0,005	0,005	0,005	0,005	**0,022**	0,005	0,005
O33	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005

Pada tabel 3.8 terdapat beberapa nilai yang dicetak tebal dan merupakan rute perjalanan yang dihasilkan oleh semut yang menghasilkan *makespan* terbaik yaitu semut ke-3. Data-data dalam tabel tersebut nantinya akan digunakan sebagai nilai jejak semut (τ_{ij}) untuk melakukan pemilihan rute pada siklus selanjutnya.

Proses penjadwalan menggunakan algoritma semut ini akan berhenti jika siklus sudah mencapai nilai maksimum (NC_{Max}) atau sudah dalam keadaan konvergen. Pada setiap siklus akan dipilih *makespan* terbaik yang akan dibandingkan dengan *makespan* terbaik yang didapat pada semua siklus dan akan dijadikan sebagai keluaran dari sistem ini.

3.4 Perancangan Antar Muka

Rancangan antar muka dari aplikasi penjadwalan ini dibuat dengan menggunakan Lazarus IDE dan hanya terdiri dari satu form yang didalamnya terdapat beberapa panel yang digunakan sebagai tempat meletakkan komponen masukan dan keluaran dari aplikasi ini. Gambar rancangan awal antar muka aplikasi adalah seperti pada gambar 3.3 berikut ini.



Gambar 3.6 Rancangan Antar Muka Aplikasi Penjadwalan

Penjelasan tentang fungsi dari tiap panel yang ada pada rancangan aplikasi pada gambar 3.6 adalah sebagai berikut :

1. Panel Data Urutan Mesin
   Panel ini merupakan panel yang berisi data urutan mesin yang ditampilkan pada sebuah *stringgrid*.
2. Panel Waktu Proses
   Panel ini merupakan panel yang berisi data waktu proses yang ditampilkan pada sebuah stringgrid. Tombol Buka berfungsi untuk membuka data penjadwalan yang berasal dari file berformat teks (*.txt), sedangkan tombol proses digunakan untuk memproses data penjadwalan tersebut.
3. Panel Parameter awal
   Panel ini berisi komponen masukan parameter awal yang harus dimasukkan pengguna yang nantinya digunakan dalam perhitungan menggunakan algoritma semut.
4. Panel Hasil Komputasi
   Panel ini digunakan untuk menampilkan hasil komputasi data penjadwalan yang berisi data siklus, data semut, data rute perjalanan semut dan nilai makespan yang dihasilkan tiap semut.

3.5 Perancangan Uji Coba

Uji coba dilakukan untuk mengetahui efektitifitas parameter algoritma semut dalam meminimumkan *makespan* untuk penjadwalan *job shop*. Parameter yang diuji adalah jumlah semut, intensitas *pheromone* awal, Parameter α, Parameter β, Parameter ρ, parameter Q, dan jumlah siklus. Nilai terbaik dari tiap paramater akan digunakan pada pengujian parameter selanjutnya. Data yang digunakan sebagai uji coba merupakan data *job shop* yang berasal dari *OR-Library* yaitu *Fisher and Thompson 6x6 instance* yang berisi data 6 job dan 6 mesin. Tabel 3.9, tabel 3.10, tabel 3.11, tabel 3.12, tabel 3.13, tabel 3.14, tabel 3.14, dan tabel 3.15 merupakan tabel rancangan uji coba dan nilai parameter yang akan diujikan terhadap algoritma semut untuk menghasilkan rata-rata *makespan* yang minimum.

Tabel 3.9 Rancangan Uji Coba Parameter Jumlah Semut

Semut	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
10						
20						
30						
40						
50						

Tabel 3.10 Rancangan Uji Coba Parameter Intensitas *Pheromone* Awal

Tau awal	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
0.01						
0.03						
0.05						
0.07						
0.09						

Tabel 3.11 Rancangan Uji Coba Parameter α

Alpha	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
10						
20						
30						
40						
50						

Tabel 3.12 Rancangan Uji Coba Parameter β

Betha	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
10						
20						
30						
40						
50						

Tabel 3.13 Rancangan Uji Coba Parameter ρ

Rho	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
0.1						
0.3						
0.5						
0.7						
0.9						

Tabel 3.14 Rancangan Uji Coba Parameter Q

Q	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
50						
100						
150						
200						
250						

Tabel 3.15 Rancangan Uji Coba Parameter Jumlah Siklus

Siklus	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
100						
500						
1000						
1500						
2000						

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dibahas seluruh proses yang telah dirancang pada bab sebelumnya, tampilan antarmuka dan bagian *source code* yang telah dibuat, serta analisis terhadap hasil uji coba data yang telah dihasilkan oleh program yang selesai dibangun.

4.1 Lingkungan Implementasi

Implementasi merupakan proses transformasi representasi rancangan ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Lingkungan implementasi yang akan dijelaskan meliputi lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem optimasi penjadwalan *job shop* dengan menggunakan algoritma semut ini adalah sebagai berikut :

1. Prosesor AMD TURION X2 2.00 GHz
2. RAM 3072 MB
3. *Harddisk* dengan kapasitas 160 GB
4. LCD Monitor 14.1”
5. Keyboard
6. Touchpad

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem ini adalah :

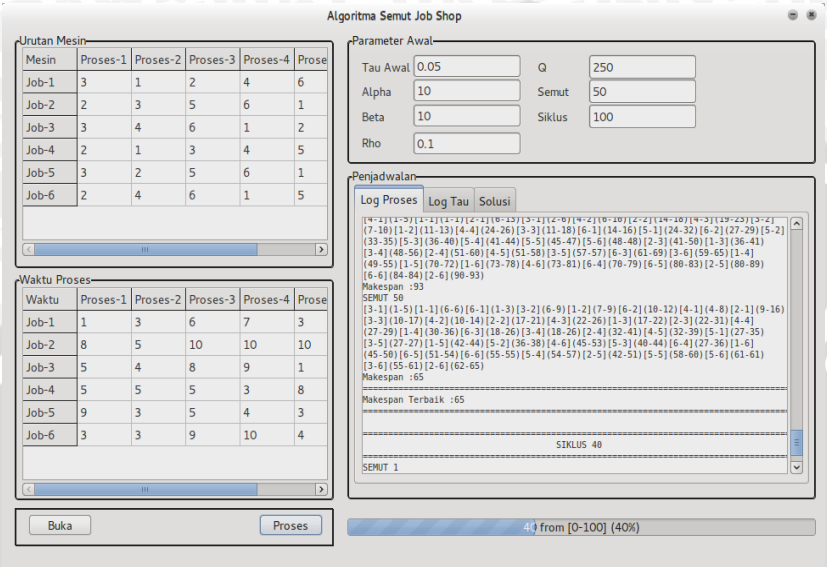
1. Sistem Operasi ARCHLINUX 2010.05 Kernel 2.6.38
2. Lazarus IDE sebagai alat untuk membangun aplikasi penjadwalan *jobshop* dengan menggunakan Algoritma Semut ini.
3. Bahasa pemrograman yang digunakan adalah Free Pascal.

4.2 Implementasi

Implementasi perangkat lunak berupa aplikasi pemrograman yang menggunakan metode Algoritma Semut dalam mengoptimasi penjadwalan *job shop.*

Tampilan utama dari aplikasi penjadwalan *job shop*

menggunakan algoritma semut ini dapat dilihat pada gambar 4.1



Gambar 4.1 Tampilan Aplikasi Penjadwalan.

4.2.1 Deskripsi Program

Berdasarkan perancangan dan analisa proses yang terdapat pada BAB III, maka pada sub bab ini akan dijelaskan implementasi dari proses-proses tersebut ke dalam suatu aplikasi.

4.2.1.1 Proses Representasi Data *Job Shop* Kedalam Graf

Proses ini dilakukan untuk merepresentasikan data *job shop* yang dimasukkan ke dalam bentuk node-node yang saling berhubungan sehingga dapat digunakan oleh semut dalam menentukan rute perjalanan. Prosedur untuk merepresentasikan data *job shop* kedalam bentuk graf bisa dilihat pada *source code* 4.1.

```
1  procedure TForm1.SetGrapNode(Node:TNode);
2  var i,j,k,l,x:integer;
3  begin
4   x:=0;
5   for i:=1 to JmlJob do
6    for j:=1 to JmlProses do
7      begin
8       inc(x);
```

```
 9       xGrapNode[x].id:=Node[i,j].id; //masukkan id
10       xGrapNode[x].job:=Node[i,j].job;
11       xGrapNode[x].index:=x;
12      end;
13   for i:=0 to (JmlJob*JmlProses) do
14    begin
15     for k:=1 to JmlJob do
16      for l:=1 to JmlProses do
17        begin
18         xGrapNode[i].node_tujuan[k,l]:=Node[k,l];
19         if Trim(xGrapNode[i].id)=
20           Trim(xGrapNode[i].node_tujuan[k,l].id) then
21            begin
22              xGrapNode[i].node_tujuan[k,l].waktu:=0;
23              xGrapNode[i].node_tujuan[k,l].status:=4;
24             end;
25           end;
26     end;
27 end;
```

***Source Code* 4.1** Prosedur Representasi Data Ke Dalam Bentuk Graf

Pada *source code* 4.1 baris 5 sampai baris 12 digunakan untuk menentukan node awal dari graf. Sedangkan baris 13 sampai baris 29 digunakan untuk menentukan node yang bisa dituju dari node awal graf.

4.2.1.2 Proses Penghitungan Peluang Tiap Node

Proses ini dilakukan untuk menghitung peluang dari node-node yang boleh dikunjungi semut dari posisi semut pada suatu node. Proses ini dihitung dengan persamaan 2.1. Hasil dari proses ini nanti akan digunakan oleh semut untuk menentukan node mana yang akan dikunjungi. Pada proses ini terdapat dua prosedur dan satu fungsi yang saling berhubungan yaitu prosedur untuk menghitung nilai etha, prosedur untuk menghtiung pembagi peluang dan fungsi untuk mengambil node dengan peluang terbesar. Prosedur untuk menghitung nilai etha dapat dilihat pada *source code* 4.2, sedangkan prosedur untuk menghitung pembagi dapat dilihat pada *source code* 4.3, dan fungsi untuk mencari peluang terbesar dapat dilihat pada *source code* 4.4.

```
1  procedure TForm1.HitungEtha();
2  var i,j,k,l:integer;
```

```
3  begin
4   for i:=0 to (JmlJob*JmlProses) do
5    begin
6     for j:=1 to JmlJob do
7      for k:=1 to JmlProses do
8       begin
9        With xGrapNode[i].node_tujuan[j,k] do
10        begin
11         if waktu > 0 then etha:=1/waktu
12          else etha:=0;
13        end;
14       end;
15   end;
16 end;
```

***Source Code* 4.2** Prosedur Menghitung Nilai Etha

Source code 4.2 digunakan untuk menghitung nilai etha yang dihitung dengan cara membagi 1 dengan waktu proses dari node yang akan dituju. Etha ini merupakan nilai visibilitas antar node yang nantinya akan digunakan untuk menghitung peluang node.

```
1  Function
2  TForm1.HitungPembagiPeluang(id:integer):Real;
3  var pembagi:Real;i,j,k:integer;
4  begin
5   pembagi:=0;
6   for j:=1 to JmlJob do
7    for k:=1 to JmlProses do
8      begin
9       if xGrapNode[id].node_tujuan[j,k].status=1 then
10         pembagi:=pembagi+
11   ((power(xGrapNode[id].node_tujuan[j,k].tau,Alpha))*
12  (power(xGrapNode[id].node_tujuan[j,k].etha,Betha)));
13      end;
14    Result:=pembagi;
   end;
```

***Source Code* 4.3** Prosedur Menghitung Nilai Pembagi Peluang

```
1  function TForm1.GetNodePeluang(id:integer):TRNode;
2  var peluang,pembagi:Real;j,k:integer;
```

```
3  begin
4   peluang:=0;
5   pembagi:=HitungPembagiPeluang(id);
6   for j:=1 to JmlJob do
7    for k:=1 to JmlProses do
8     begin
9     if xGrapNode[id].node_tujuan[j,k].status = 1 then
10      begin
11       xGrapNode[id].node_tujuan[j,k].peluang:=((powe
12 r(xGrapNode[id].node_tujuan[j,k].tau,Alpha))*
13       (power(xGrapNode[id].node_tujuan[j,k].etha,Beth
14 a)))/pembagi;
15 if xGrapNode[id].node_tujuan[j,k].peluang >= peluang
16 then
17    begin
18     peluang:=xGrapNode[id].node_tujuan[j,k].peluang;
19     Result:=xGrapNode[id].node_tujuan[j,k];
20    end;
21   end;
22  end;
23 end;
```

***Source Code* 4.4** Prosedur Menghitung Nilai Peluang Node.

Source code 4.3 dan *source code* 4.4 digunakan untuk menghitung peluang sesuai dengan persamaan 2.1 lalu memilih peluang terbaik. parameter id pada baris pertama merupakan posisi semut. Pada baris ke-8 terdapat parameter status yang digunakan untuk melihat node yang bisa dikunjungi dari posisi semut tersebut dan baris selanjutnya digunakan untuk mengitung peluang dari node-node yang bisa dikunjungi sesuai persamaan 2.1. Pada *source code* 4.4 Baris 15 dan seterusnya digunakan untuk memilih peluang terbesar.

4.2.1.3 Proses Penentuan Rute Kunjungan Semut

Setelah peluang tiap node yang boleh dikunjungi dihitung mka selanjutnya semut akan melakukan pemilihan node yang akan dikunjunginya dan menyimpannya dalam sebuah daftar node kunjung. Proses ini sesuai dengan *flowchart* pada gambar 3.3. Dan proses ini berlangsung sampai semut melewati seluruh node yang tersedia. Prosedur untuk penentuan rute kunjungan ini dapat dilihat pada *source code* 4.5.

```
for k:=1 to (JmlJob*JmlProses) do
 begin
  random_p:=Random;
  if xSiklus[i].p0 >= random_p then
  begin
   random_q:=Random;
 if GetNodePeluang(index_kunjung).peluang>= random_q
 then xSemut[j].node_kunjung[k]:=
                      GetNodePeluang(index_kunjung)
 else
 xSemut[j].node_kunjung[k]:=
                      GetNodeRandom(index_kunjung);
   end else
   xSemut[j].node_kunjung[k]:=
                       GetNodeRandom(index_kunjung);
  SetStatusNode(index_kunjung,xSemut[j].
                                  node_kunjung[k].id);
  index_kunjung:=xSemut[j].node_kunjung[k].index;
```

***Source Code* 4.5** Prosedur Penentuan Rute Kunjungan.

Kode pada baris ke-1 sampai baris terakhir merupakan kode untuk pemilihan node oleh semut. Dan hal ini berlangsung sampai seluruh node selesai dikunjungi oleh semut sesuai dengan *flowchart* pada gambar 3.3.

4.2.1.4 Proses Penghitungan Makespan

Setelah seluruh node selesai dikunjungi oleh semut maka selanjutnya akan dilakukan penghitungan makespan dari urutan node-node tersebut. Proses ini mengolah data waktu proses dari urutan node yang dikunjungi sehingga diperoleh nilai *makespan*. Prosedur untuk proses ini dapat dilihat pada *source code* 4.6

```
xSemut[j].makespan:=0;

  xSemut[j].node_kunjung[k].start:=0;
  xSemut[j].node_kunjung[k].finish:=0;
 if CekFinish(xSemut[j].node_kunjung[k],
         xSemut[j].node_kunjung)=0 then
  begin
   xSemut[j].node_kunjung[k].start:=1;
xSemut[j].node_kunjung[k].finish:=
xSemut[j].node_kunjung[k].waktu;
  end
 else
```

```
13   begin
14    xSemut[j].node_kunjung[k].start:=
15    CekFinish(xSemut[j].node_kunjung[k],
16      xSemut[j].node_kunjung)+1;
17    xSemut[j].node_kunjung[k].finish:=
18    CekFinish(xSemut[j].node_kunjung[k],
19    xSemut[j].node_kunjung)+
20    xSemut[j].node_kunjung[k].waktu;
21   end;
22          if    xSemut[j].node_kunjung[k].finish    >=
23 xSemut[j].makespan
24     then xSemut[j].makespan:=
25     xSemut[j].node_kunjung[k].finish
26   else xSemut[j].makespan:=xSemut[j].makespan;
27
28   rute:=rute+'['+xSemut[j].node_kunjung[k].id +']
29   ('+IntToStr(xSemut[j].node_kunjung[k].start)+'-'+
30   IntToStr(xSemut[j].node_kunjung[k].finish)+')';
```

***Source Code* 4.6** Prosedur Penghitungan Nilai *Makespan*

4.2.1.5 Proses Pembaharuan Nilai Jejak Semut

Setelah semut melakukan perjalanan ke seluruh node maka akan dilakukan proses pembaharuan *pheromone* (jejak) pada setiap node yang telah dilewati oleh semut. Dan nilai *pheromone* yang baru ini nanti akan digunakan untuk perghitungan peluang node semut pada siklus selanjutnya. Prosedur untuk memperbaharui jejak ini dapat dilihat pada *source code* 4.7.

```
1  for k:=0 to (JmlJob*JmlProses) do
2   begin
3    for l:=1 to JmlJob do
4      for m:=1 to JmlProses do
5       begin
6        if xGrapNode[k].node_tujuan[l,m].status=5 then
7          begin
8           xSemut[j].delta_tau:=
9                    NilaiQ/xSemut[j].makespan;
10          tau_lama :=
11                  xGrapNode[k].node_tujuan[l,m].tau;
12          xGrapNode[k].node_tujuan[l,m].tau:=
13       ((1-Rho)*tau_lama)+ (Rho*xSemut[j].delta_tau);
14         end else
15          begin
16           xSemut[j].delta_tau:=0;
17       tau_lama := xGrapNode[k].node_tujuan[l,m].tau;
18          xGrapNode[k].node_tujuan[l,m].tau:=
```

```
19        ((1-Rho)*tau_lama)+(Rho*xSemut[j].delta_tau);
20            end;
21         end;
22   end;
```

***Source Code* 4.7** Prosedur Pembaharuan Jejak Semut.

Status pada baris ke-6 digunakan untuk mencari node yang mempunyai status sama dengan 5 yang merupakan node yang dilewati semut terbaik. Setelah diketahui status tiap node maka selanjutnya akan dilakukan pembaharuan *pheromone* (jejak semut) sesuai dengan persamaan 2.2 dan persamaan 2.3.

4.2.1.6 Proses Pencarian Makespan Terbaik dan Solusi

Setelah seluruh semut selesai melakukan perjalanan pada suatu siklus dan telah diketahui *makespan* dari masing-masing semut, maka proses selanjutnya adalah mencari *makespan* terbaik dari seluruh semut tersebut untuk dijadikan *makespan* terbaik siklus. Prosedur untuk mencari *makespan* terbaik siklus tersebut seperti pada *source code* 4.8.

```
1   if j=1 then
2     begin
3      xSiklus[i].makespan_terbaik:=xSemut[1].makespan;
4      xSemut[1].status:=1;
5      index_mks_lok:=1;
6     end
7     else
8     begin
9   if     xSemut[j].makespan<xSiklus[i].makespan_terbaik
10  then
11    begin
12     xSemut[index_mks_lok].status:=0;
13     xSiklus[i].makespan_terbaik:=xSemut[j].makespan;
14     xSemut[j].status:=1;
15     index_mks_lok:=j;
16     end else xSemut[j].status:=0;
      end;
```

***Source Code* 4.8** Prosedur Pencarian *Makespan* Terbaik Siklus

Selanjutnya dari *makespan* terbaik dari tiap-tiap siklus akan dibandingkan untuk memperoleh nilai *makespan* terbaik keseluruhan yang merupakan solusi dari sistem ini. Prosedur untuk mencari solusi dapat dilihat pada *source code* 4.9.

```
1  if xSemut[j].makespan < xSolusi[1].makespan then
2   begin
3    xSolusi[1].makespan:=xSemut[j].makespan;
4    xSolusi[1].rute:=rute;
5    xSolusi[1].siklus:=i;
6    xSolusi[1].semut_ke:=j;
7    index_solusi:=1;
8    xSemut_Global:=xSemut[j];
9    Memo3.Clear;
10   Memo3.Lines.Add(IntToStr(index_solusi)+
11    '.Siklus :'+IntToStr(xSolusi[index_solusi].siklus)
12 +
13                '      Semut      :       '       +
14 IntToStr(xSolusi[index_solusi].semut_ke) +
15                '            Makespan          :
16 '+IntToStr(xSolusi[index_solusi].makespan));
17  end else
18 if xSemut[j].makespan = xSolusi[1].makespan then
19  begin
20   Inc(index_solusi);
21  xSolusi[index_solusi].makespan:=xSemut[j].makespan;
22  xSolusi[index_solusi].rute:=rute;
23  xSolusi[index_solusi].siklus:=i;
24  xSolusi[index_solusi].semut_ke:=j;
25  Memo3.Lines.Add(IntToStr(index_solusi)+
26   ' .Siklus :'+IntToStr(xSolusi[index_solusi].siklus)
    +
           '      Semut      :       '       +
    IntToStr(xSolusi[index_solusi].semut_ke) +
                  '            Makespan          :
    '+IntToStr(xSolusi[index_solusi].makespan));
    end;
```

***Source Code* 4.9** Prosedur Pencarian Solusi

4.3 Uji Coba Parameter Algoritma Semut.

Pada aplikasi ini akan dianalisis perubahan nilai *makespan* yang dihasilkan berdasarkan perubahan nilai parameter awal algoritma semut. Parameter yang akan dianalisis adalah jumlah semut, intensitas *pheromone*, betha, alpha, rho, Q, dan jumlah siklus. Data yang akan digunakan adalah data *benchmark job shop* yaitu *Fisher and Thompson 6x6 instance* yang berisi data 6 job dan 6 mesin dengan diketahui nilai *makespan* optimum yang bisa dicapai adalah 55. Data tersebut dapat dilihat pada tabel 4.1 dan tabel 4.2.

Tabel 4.1 Urutan Mesin Data Uji

JOB	PROSES					
	1	2	3	4	5	6
1	M3	M1	M2	M4	M6	M5
2	M2	M3	M5	M6	M1	M4
3	M3	M4	M6	M1	M2	M5
4	M2	M1	M3	M4	M5	M6
5	M3	M2	M5	M6	M1	M4
6	M2	M4	M6	M1	M5	M3

Tabel 4.2 Waktu Proses Data Uji

JOB	PROSES					
	1	2	3	4	5	6
1	1	3	6	7	3	6
2	8	5	10	10	10	4
3	5	4	8	9	1	7
4	5	5	5	3	8	9
5	9	3	5	4	3	1
6	3	3	9	10	4	1

Pada pengujian parameter yang akan dilakukan, tiap perubahan nilai parameter akan diuji sebanyak 5 kali dengan menggunakan nilai parameter yang bisa dilihat pada tabel 4.3.

Tabel 4.3 Nilai Parameter Uji Coba

Alpha	Betha	Rho	Q	Tau	Siklus
10	10	0,1	50	0,01	100

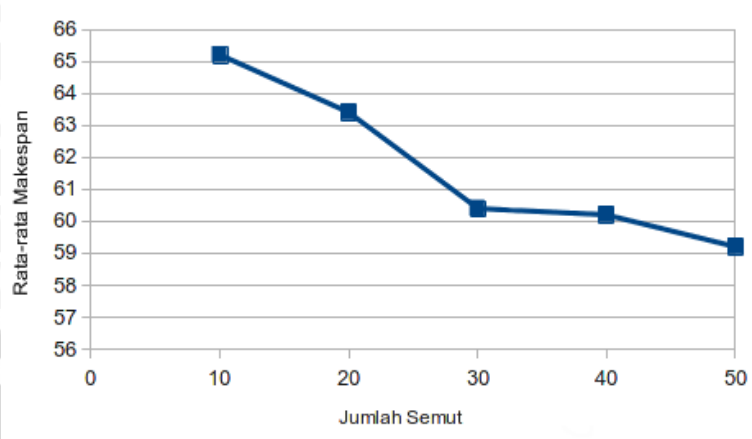
4.3.1 Pengujian Parameter Jumlah Semut

Pada Uji coba ini akan diuji parameter jumlah semut dengan cara memasukkan nilai yang berbeda pada parameter tersebut selanjutnya akan dicari rata-rata *makespan* yang paling minimum dari uji coba parameter jumlah semut ini. Nilai parameter jumlah semut yang akan diuji adalah 10, 20, 30, 40, 50. Hasil dari uji coba parameter jumlah semut dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil Uji Coba Parameter Jumlah Semut.

Jumlah semut	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
10	65	67	65	64	65	65,2
20	65	60	62	67	63	63,4
30	60	60	61	60	61	60,4
40	61	61	60	59	60	60,2
50	59	59	60	59	59	59,2

Dari tabel 4.4 nilai parameter jumlah semut yang menghasilkan nilai rata-rata *makespan* paling minimum adalah 50 dengan nilai rata-rata *makespan* yang dihasilkan sebesar 59,2. Untuk grafik pengaruh jumlah semut pada nilai rata-rata *makespan* dapat dilihat pada gambar 4.2.



Gambar 4.2 Grafik Pengaruh Jumlah Semut.

Dari gambar 4.2 terlihat dengan meningkatnya jumlah semut maka *makespan* yang dihasilkan akan semakin minimum. Hal ini dikarenakan dengan semakin banyak semut maka variasi rute perjalanan yang bisa dijadikan solusi akan semakin banyak.

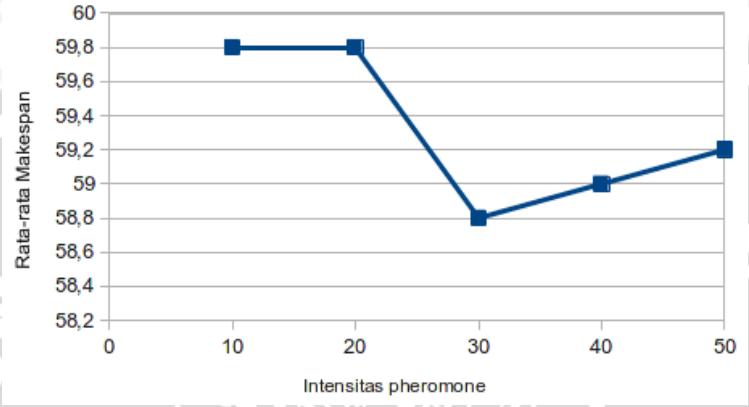
4.3.2 Pengujian Parameter Intensitas *Pheromone* awal (tau)

Pengujian ini dilakukan untuk mencari nilai intensitas *pheromone* awal yang menghasilkan *makespan* minimum. Nilai yang akan diuji adalah 0.01, 0.03, 0.05, 0.07, dan 0.09. Pada pengujian intensitas *pheromone* ini digunakan nilai parameter jumlah semut terbaik dari uji coba sebelumnya yaitu 50. Hasil pengujian intensitas *pheromone* awal dapat dilihat pada tabel 4.5

Tabel 4.5 Hasil Uji Coba Parameter Intensitas *Pheromone.*

tau	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
0,01	60	59	60	60	60	59,8
0,03	60	60	61	59	59	59,8
0,05	57	61	60	61	55	58,8
0,07	60	59	60	58	58	59
0,09	60	58	60	60	58	59,2

Pada tabel 4.5 terlihat bahwa nilai intensitas *pheromone* yang menghasilkan nilai rata-rata *makespan* terbaik adalah 0,05 dengan nilai 58,8. Grafik hubungan antara pengaruh intensitas *pheromone* terhadap nilai rata-rata *makespan* dapat dilihat pada gambar 4.6



Gambar 4.3 Grafik Pengaruh Intensitas Pheromone.

Dari gambar 4.3 terlihat bahwa pertambahan intensitas *pheromone* awal tidak terlalu berpengaruh pada rata-rata *makespan* dan cenderung menghasilkan nilai yang tidak stabil. Hal ini dikarenakan intensitas *pheromone* awal ini hanya digunakan pada siklus awal pada algoritma semut dan setelah itu akan berubah karena dipengaruhi oleh tingkat penguapan dan jumlah semut yang melewati node.

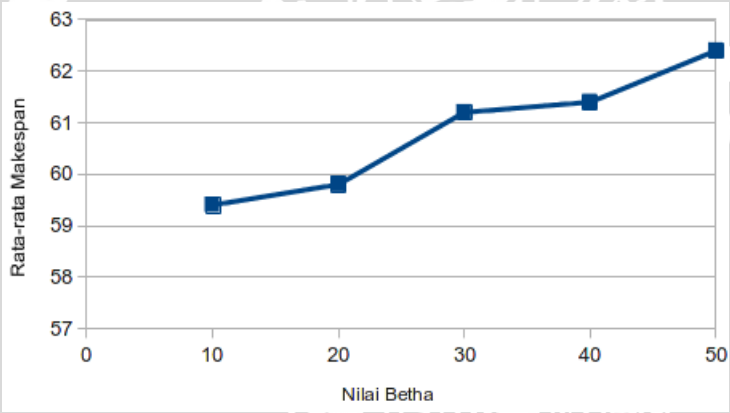
4.3.3 Pengujian Parameter Betha

Parameter betha diuji coba dengan nilai 10, 20, 30, 40, 50. Untuk nilai alpha yang digunakan adalah 10 dan nilai rho yang digunakan adalah 0,1. Uji coba dilakukan masing-masing 5 kali untuk setiap perubahan nilai parameter betha lalu diambil rata-rata *makespan* yang dihasilkan untuk diambil yang paling minimum. Uji coba ini dilakukan dengan menggunakan nilai intensitas *pheromone* terbaik dari ujicoba sebelumnya sehingga dapat diketahui kombinasi terbaik dari parameter-parameter ini. Hasil ujicoba parameter betha dapat dilihat pada tabel 4.6.

Tabel 4.6 Hasil Uji Coba Parameter Betha

betha	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
10	60	59	59	60	59	59,4
20	60	60	60	59	60	59,8
30	59	63	64	59	61	61,2
40	60	61	60	65	61	61,4
50	63	61	63	64	61	62,4

Pada tabel 4.6 terlihat bahwa nilai rata-rata *makespan* terbaik yang dihasilkan oleh parameter betha saat bernilai 10 yaitu 59,4. Dan grafik hubungan antara parameter betha dengan nilai rata-rata *makespan* dapat dilihat pada gambar 4.7.



Gambar 4.4 Grafik Pengaruh Nilai Betha

Pada gambar 4.4 terlihat bahwa dengan meningkatnya nilai betha maka nilai rata-rata *makespan* yang didapatkan akan cenderung semakin naik walaupun selisih antar nilai rata-rata *makespan* tidak terlalu jauh. Meningkatnya nilai *makespan* tersebut disebabkan dengan bertambah besarnya nilai betha daripada alpha maka nilai jarak heuristik pada perhitungan peluang node akan semakin besar dan komunikasi semut lewat *pheromone* akan semakin kecil. Sehingga pertimbangan semut untuk memilih node

berdasarkan jarak node tujuan terdekat lebih besar daripada memilih node menggunakan intensitas *pheromone*.

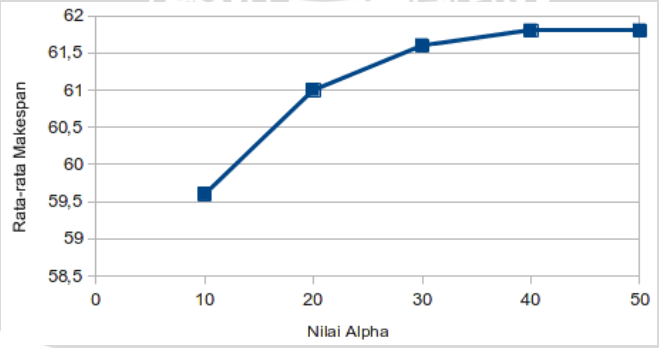
4.3.4 Pengujian Parameter Alpha

Pengujian parameter ini dilakukan untuk mengetahui nilai alpha yang menghasilkan makespan paling minimum. Pengujian dilakukan sebanyak 5 kali percobaan dengan menggunakan nilai rho = 0,1 dan nilai parameter betha yang digunakan mengacu pada nilai sebelumnya yaitu 10. Hasil uji coba parameter alpha ini dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil Uji Coba Parameter Alpha

alpha	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
10	60	60	60	60	58	59,6
20	59	59	60	60	67	61
30	62	65	60	60	61	61,6
40	60	64	65	60	60	61,8
50	63	60	61	60	65	61,8

Pada tabel 4.7 terlihat bahwa dengan melakukan pengujian nilai alpha antara 10 sampai 50 didapatkan rata-rata *makespan* paling optimal yaitu 59,6 pada saat nilai alpha sama dengan 10. Grafik hubungan pengaruh nilai alpha terhadap rata-rata *makespan* dapat dilihat pada gambar 4.5



Gambar 4.5 Grafik Pengaruh Nilai Alpha

Dari gambar 4.5 terlihat bahwa *makespan* paling minimum

dihasilkan pada saat nilai alpha sama dengan nilai betha sedangkan pada saat nilai alpha lebih besar dari betha terjadi peningkatan nilai *makespan.* Nilai alpha ini berpengaruh pada jumlah intensitas *pheromone* pada saat dilakukan penghitungan peluang node. Dan semakin lebih besar nilai alpha daripada nilai betha maka semut akan mempertimbangkan pemilihan node lebih memilih node berdasarkan jumlah *pheromone* daripada jarak antar node.

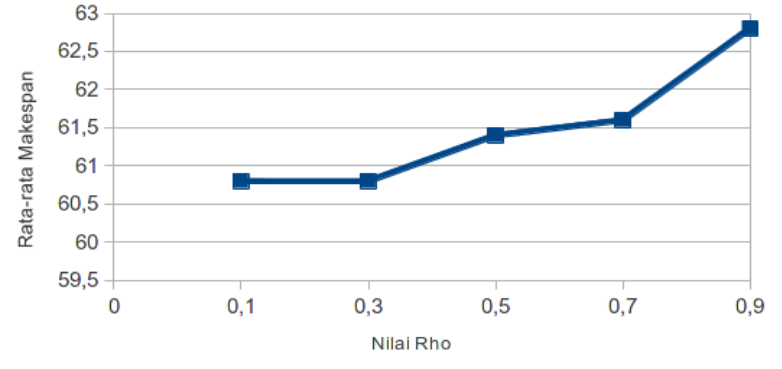
4.3.5 Pengujian Parameter Rho

Setelah didapat nilai alpha dan betha yang menghasilkan nilai makespan minimum maka selanjutnya akan dilakukan pengujian terhadap parameter rho sesuai dengan nilai alpha = 10 dan betha= 10. Hasil dari uji coba parameter rho ini dapat dilihat pada tabel 4.8

Tabel 4.8 Hasil Uji Coba Parameter Rho

Rho	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
0,1	61	62	61	59	61	60,8
0,3	59	62	60	63	60	60,8
0,5	61	62	65	60	59	61,4
0,7	66	59	60	62	61	61,6
0,9	64	65	61	62	62	62,8

Pada tabel 4.8 terlihat bahwa dengan melakukan pengujian nilai rho antara 0,1 sampai 0,9 didapatkan nilai rata-rata *makespan* paling optimal yaitu 60,8 pada saat nilai rho sama dengan 0,1. Grafik hubungan pengaruh nilai rho terhadap rata-rata *makespan* dapat dilihat pada gambar 4.6.



Gambar 4.6 Grafik Pengaruh Nilai Rho

Dari gambar 4.6 terlihat bahwa semakin kecil nilai rho maka nilai *makespan* yang dihasilkan akan semakin minimum hal ini disebabkan oleh nilai rho yang kecil maka penguapan intensitas *pheromone* (jejak) pada saat dilakukan pembaharuan jejak juga sedikit. Dan dengan jumlah jejak yang banyak pada suatu node maka akan mampu menarik minat semut untuk menuju node tersebut.

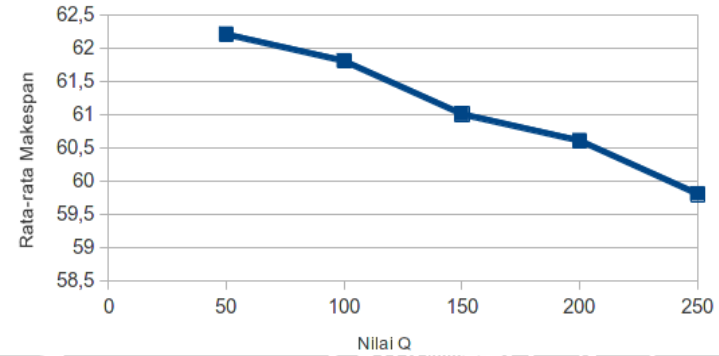
4.3.6 Pengujian Parameter Q

Uji coba selanjutnya adalah menguji parameter Q dengan nilai terbaik dari uji coba sebelumnya yang dapat menghasilkan nilai makespan yang minimum yaitu alpha= 10, betha= 10, *pheromone*=0,05 , dan rho = 0,1. Hasil dari uji coba ini dapat dilihat pada tabel 4.9.

Tabel 4.9 Hasil Uji Coba Parameter Q

Q	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
50	63	61	62	64	61	62,2
100	59	63	61	63	63	61,8
150	60	62	63	60	60	61
200	60	61	60	61	61	60,6
250	61	61	59	59	59	59,8

Pada tabel 4.9 terlihat bahwa dengan melakukan pengujian nilai Q antara 50 sampai 250 didapatkan nilai rata-rata *makespan* paling optimal yaitu 59,8 pada saat nilai Q sama dengan 250. Grafik hubungan pengaruh nilai Q terhadap rata-rata *makespan* dapat dilihat pada gambar 4.7.



Gambar 4.7 Grafik Pengaruh Nilai Q

Seperti terlihat pada gambar 4.7 semakin besar nilai Q maka *makespan* yang dihasilkan juga semakin minimum. Dengan bertambah besarnya nilai Q maka bertambah besar pula jumlah *pheromone* yang ditambahkan pada saat dilakukan pembaharuan intensitas *pheromone* sesuai persamaan 2.2 dan persamaan 2.3. Sehingga membuat peluang node terbaik untuk dipilih lagi akan semakin besar.

4.3.7 Pengujian Parameter Jumlah Siklus

Setelah itu parameter terakhir yang akan diuji coba adalah jumlah siklus dan diuji coba dengan menggunakan nilai dari parameter-parameter sebelumnya yang telah menghasilkan nilai *makespan* paling minimum. Nilai parameter-parameter tersebut dapat dilihat pada tabel 4.10. Dan hasil uji coba dari parameter jumlah siklus ini dapat dilihat pada tabel 4.11.

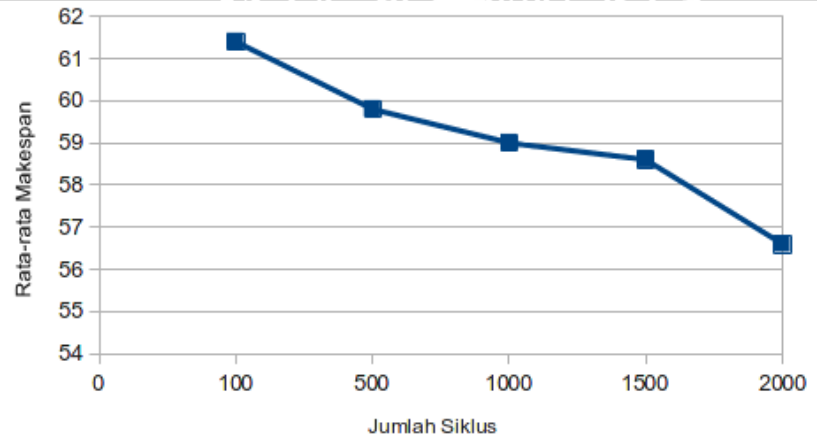
Tabel 4.10 Hasil Uji Coba Parameter Terbaik

Alpha	Betha	Rho	Q	Tau	Semut
10	10	0,1	250	0,05	50

Tabel 4.11 Hasil Uji Coba Parameter Jumlah Siklus

siklus	Uji Coba					Rata-rata *makespan*
	1	2	3	4	5	
100	62	61	62	62	60	61,4
500	60	59	61	60	59	59,8
1000	59	61	59	59	57	59
1500	60	57	59	57	60	58,6
2000	55	57	55	59	57	56,6

Dari tabel 4.11 terlihat bahwa dengan melakukan pengujian terhadap parameter jumlah siklus dengan nilai antara 100 sampai 2000 didapatkan *makespan* paling minimum sebesar 56,6 yang didapatkan pada saat jumlah siklus sama dengan 2000. Untuk grafik pengaruh nilai parameter jumlah siklus terhadap rata-rata nilai *makespan* dapat dilihat pada gambar 4.8.



Gambar 4.8 Grafik Pengaruh Jumlah Siklus.

Dari gambar 4.8 terlihat bahwa dengan meningkatnya jumlah siklus maka nilai rata *makespan* yang dihasilkan semakin membaik dengan nilai rata *makespan* paling minimum yang didapat adalah 56,6 dari 2000 siklus. Hal ini dikarenakan semakin banyak siklus maka akan semakin banyak pula jalur baru yang akan dibuka oleh

semut sehingga pilihan solusi akan semakin banyak. Dan hasil rata-rata *makespan* yang paling minimum dari uji coba parameter jumlah siklus mendekati nilai *makespan* minimum dari data uji yang besarnya 55.

4.4 Analisis Hasil

Pada algoritma semut terdapat parameter jumlah semut (m), parameter jumlah siklus (Nc_{max}), intensitas *pheromone* (τ_{ij}), tetapan pengendali intensitas *pheromone* (α), tetapan pengendali informasi heuristik (β), tetapan penguapan jejak semut (ρ), dan konstanta jumlah *pheromone* (Q). Untuk menghasilkan *makespan* yang optimum maka dibutuhkan pemilihan kombinasi parameter yang tepat. Oleh karena itu untuk dapat menentukan kombinasi parameter yang baik tersebut maka dilakukan pengujian sebanyak 5 kali untuk setiap perubahan nilai parameter.

Jumlah semut menunjukkan banyaknya semut yang melakukan perjalanan pada setiap siklus. Semakin banyak jumlah semut maka semakin banyak kombinasi solusi yang dapat dipilih pada suatu siklus. Sedangkan jumlah siklus menunjukkan jumlah perjalanan yang harus dilakukan oleh semut. Berdasarkan *makespan* yang dihasilkan interaksi antara jumlah semut dan jumlah siklus yang semakin besar maka solusi yang dihasilkan akan semakin baik. Hal ini dibuktikan dengan interaksi antara 2000 siklus dengan 50 semut menghasilkan rata-rata *makespan* yang lebih rendah daripada jumlah interaksi jumlah siklus dan jumlah semut lainnya.

Pada uji coba parameter itensistas *pheromone* awal (τ_{ij}) *makespan* yang dihasilkan cenderung tidak stabil hal ini dikarenakan nilai intensitas pheromone ini akan berubah pada saat terjadi pembaharuan jejak.

Parameter alpha menunjukkan pengaruh semut sebelumnya dan parameter betha menunjukkan pengaruh informasi heuristik. Kedua parameter ini berpengaruh saat semut melakukan pemilihan node yang akan dikunjungi. Dari interaksi kedua parameter ini terlihat bahwa nilai alpha dan betha yang sama akan menghasilkan nilai *makespan* yang baik. Hal ini dikarenakan dengan nilai parameter alpha dan betha yang sama maka proporsi semut untuk memilih berdasarkan intensitas *pheromone* dan berdasarkan informasi heuristik akan sama jadi semut benar-benar menggunakan keduanya dalam pemilihan node. Jika parameter alpha lebih besar dari

parameter betha maka semut akan lebih terpengaruh pada jumlah intensitas *pheromone,* sebaliknya jika nilai parameter betha lebih besar maka semut akan lebih terpengaruh pada informasi heuristik.

Parameter rho merupakan penguapan jejak semut untuk setiap node pada setiap siklus sedangkan parameter Q menunjukkan jumlah *pheromone* yang ditambahkan dan kedua parameter ini berinteraksi pada saat pembaharuan jejak. Berdasarkan rata-rata *makespan* yang dihasilkan jumlah penguapan yang sedikit dan jumlah penambahan *pheromone* yang banyak maka solusi yang dihasilkan akan semakin baik. Hal ini dikarenakan jumlah *pheromone* pada node pada rute perjalanan terbaik akan selalu lebih besar dari pada node lainnya sehingga semut-semut akan cenderung mengikuti rute terbaik tersebut.

BAB V
PENUTUP

5.1 KESIMPULAN

Kesimpulan yang dapat diambil dari skripsi ini adalah :

1. Algoritma semut dapat digunakan untuk menyelesaikan masalah penjadwalan *job shop* dan dari pengujian nilai *makespan* yang dihasilkan mendekati nilai *makespan* optimal dari data *benchmark*.
2. Dengan meningkatnya nilai parameter jumlah semut dan jumlah siklus maka akan semakin minimum nilai *makespan* yang dihasilkan. Parameter intensitas *pheromone* awal tidak berpengaruh terhadap nilai *makespan.* Pada pengujian nilai alpha dan betha akan didapatkan *makespan* yang minimum pada saat nilai keduanya sama. Dengan tingkat penguapan yang kecil dan penambahan jumlah *pheromone* yang banyak akan menghasilkan *makespan* yang semakin minimum.

5.2 SARAN

Aplikasi yang dibangun dalam skripsi ini masih belum sempurna , dan saran yang bisa digunakan untuk pengembangan adalah Algoritma semut yang digunakan pada aplikasi yang telah dibuat masih murni dan bisa dikembangkan dengan melakukan hibridasi dengan pendekatan lain.

DAFTAR PUSTAKA

Anonim. 2007. *JSS Benchmark Results*. http://bach.istc.kobe-u.ac.jp//csp2sat/jss/, tanggal akses : 20 Desember 2010.

Barker, Kenneth R. 1974. *Introduction To Sequencing And Schedulling,* John Wiley and Son Inc: America.

Beasley J.E. *OR-library*. http://people.brunel.ac.uk/~mastjjb /jeb/orlib/jobshopinfo.html, : tanggal akses : 20 Desember 2010.

Bedworth, David D., Bailey, James E. 1987. *Integrated Production Control Systems* . John Wiley and Sons Inc:Singapore.

Castrillon, Omar. 2009. *Job Shop Methodology Based On An Ant Colony*: Portugal.

Dorigo, Marco. 1996. *The Ant System : Optimization By A Colony Of Cooperating Agents*:Italia.

Ginting Rosnani. 2009. Penjadwalan Mesin. Graha ilmu: Yogyakarta.

Kumar, R. 2003. *Schedulling Of Flexible Manufacturing Systems: An Ant Colony Approach*:India.

van der Zwaan, Sjoerd. 2001. *Ant Colony Optimization For Job Shop Schedulling*: Portugal.