**BAB 2 :**

* 1. **Definisi Penjadwalan *secara umum***

Secara umum penjadwalan menurut Baker (1974) didefinisikan sebagai proses pengalokasian sumber-sumber dalam jangka waktu tertentu untuk melakukan sekumpulan pekerjaan. Definisi ini mengandung dua arti yang berbeda, yaitu :

1. Penjadwalan merupakan fungsi pengambilan keputusan, yaitu menentukan jadwal.
2. Penjadwalan merupakan suatu teori, yaitu sekumpulan prinsip-prinsip dasar, model-model, teknik-teknik, dan kesimpulan-kesimpulan logis dalam proses pengambilan keputusan yang memberikan dalam fungsi penjadwalan (nilai konseptual).

Menurut Conway *et. al.* (1967) Penjadwalan adalah proses pengurutan pembuatan produk secara menyeluruh pada beberapa mesin. Menurut Morton dan Pentico penjadwalan adalah proses pengorganisasian, pemilihan dan pemberian waktu dalam penggunaan sumber dayanya untuk melaksanakan aktivitas yang diperlukan dalam menghasilkan output yang diinginkan dengan memenuhi waktu yang diinginkan pula.

Persoalan penjadwalan timbul apabila jumlah mesin dan peralatan yang dimiliki terbatas sedangkan terdapat beberapa pekerjaan yang dapat dikerjakan secara bersama. Untuk mendapat hasil yang optimal dengan keterbatasan sumber daya yang dimiliki, maka diperlukan adanya penjadwalan sumber-sumber tersebut secara efisien.

Tujuan penjadwalan secara umum Baker (1974) adalah :

1. Meningkatkan produktivitas mesin, yaitu dengan mengurangi waktu menganggur mesin.
2. Mengurangi terhadap persediaan barang setengah jadi, dengan mengurangi rata-rata pekerjaan yang menunggu dalam antrian karena mesin sibuk oleh pekerjaan lain.
3. Mengurangi keterlambatan (*tardiness*). Dalam banyak hal, beberapa atau semua pekerjaan mempunyai batas waktu penyelesaian (*duedate*). Apabila suatu pekerjaan melewati batas waktu tersebut, maka akan dikenai pinalti. Keterlambatan dapat diperkecil dengan mengurangi maksimal *tardiness* atau mengurangi pekerjaan yang terlambat (*number of tardy job*).

Terdapat target utama yang ingin dicapai melalui penjadwalan *flow shop* ini yaitu jumlah output yang dihasilkan (*throughput*) berupa *makespan*. Penjadwalan *flow shop* didefinisikan sebagai penjadwalan dimana setiap *job* mempunyai pola aliran atau rute proses yang tetap pada seluruh mesin.

**2.1.1 Perbedaan Flow Shop dengan Job Shop**

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.
2. Pola aliran proses, dapat berupa aliran identik atau sembarang.
3. Pola kedatangan pekerjaan, Statis atau Dinamis.
4. Sifat informasi yang diterima, dapat berupa Deterministik atau Stokastik.

Pada jenis keadaan pertama, jumlah mesin dapat dibedakan atas mesin tunggal dan mesin majemuk. Model mesin tunggal merupakan model dasar dan biasanya dapat diterapkan dalam kasus mesin majemuk.

Pada model kedua, pola aliran dapat dibedakan atas *Flow Shop* dan *Job Shop*. Pada *Flow Shop* dijumpai pola aliran pemrosesan dari suatu mesin ke mesin yang lain dalam urutan (*routing*) tertentu. Semua pekerjaan yang mengalir pada saat produksi yang sama tanpa boleh melewatinya disebut dengan *pure Flow Shop*. Tetapi jika pekerjaan yang datang kedalam *Flow Shop* tidak harus dikerjakan pada semua mesin, jenis *Flow Shop* ini disebut dengan *General Flow Shop*. Contoh pola aliran *Pure Flow Shop* dan contoh pola aliran *Gereral Flow Shop* ditunjukkan pada Gambar 2.1 dan Gambar 2.2.

Input

(pekerjaan baru)

Output

(pekerjaan lengkap)

Mesin m

Mesin m-1

Mesin3

Mesin2

Mesin1

**Gambar 2.1** *Pola Aliran Pure Flow Shop*

Sumber : Baker (1974)

Input Input Input Input Input

Output Output Output Output Output

Mesin2

Mesin3

Mesin m-1

Mesin m

Mesin1

**Gambar 2.2** *Pola Aliran General Flow shop*

Sumber : Baker (1974)

Pada *Job Shop* setiap pekerjaan mempunyai *routing* yang berbeda. Alir proses yang tidak searah ini mengakibatkan setiap pekerjaan yang akan diproses pada suatu mesin dapat merupakan pekerjaan baru atau pekerjaan yang sedang dikerjakan (*work in proses*).

Pekerjaan-pekerjaan baru

Pekerjaan dalam proses pekerjaan dalam proses

Pekerjaan-pekerjaan lengkap

Mesin k

**Gambar 2.3** *Pola Aliran Job Shop*

Sumber : Baker (1974)

Pada model ketiga, pola kedatangan pekerjaan dapat dibedakan atas pola kedatangan Statis dan Dinamis. Pada pola Statis, pekerjaan datang secara bersamaan pada waktu nol, siap dikerjakan pada mesin-mesin yang juga sudah siap untuk bekerja atau kedatangan pekerjaan yang tidak bersamaan tetapi saat kedatangan telah diketahui sejak waktu nol. Sedangkan pola Dinamis mempunyai kedatangan pekerjaan tidak menentu, dijumpai adanya variable waktu sebagai faktor pengaruh.

Pada model keempat, perilaku elemen-elemen penjadwalan dapat dibedakan atas Deterministik dan Stokastik. Model Deterministik dapat dilihat dari adanya kepastian atas informasi tentang beberapa aspek. Sedangkan pada model Stokastik, mengandung unsur ketidakpastian. Aspek yang dimaksud adalah :

1. Karakteristik pekerjaan dari segi kedatangan, jumlah (kuantitas) pekerjaan, batas waktu penyelesaian (*duedate*) dan perbedaan kepentingan antar pekerjaan.
2. Karakteristik pekerjaan dari segi banyaknya operasi, susunan mesin dan waktu proses.
3. Karakteristik mesin dari segi jumlah dan kapasitas mesin, kemampuan dan kecocokan tiap mesin dengan pekerjaan yang diberikan.

**2.1.2 Beberapa Istilah dalam Penjadwalan *Flow Shop***

Penjadwalan *Flow shop* dapat dijelaskan sebagai berikut. Jika terdapat n *job* {*j*1, *j*2, …, *j*n}, maka harus diproses pada m mesin {m1, m2, …, mm}. Waktu yang diperlukan untuk memproses *job* ***i*** pada mesin ***j***adalah t***ij***. Jadi permasalahan penjadwalan adalah menentukan urutan *job* yang memberikan solusi terbaik berdasarkan kriteria tertentu.

Beberapa istilah yang digunakan dalam masalah penjadwalan yaitu :

1. Waktu proses (*processing time* = **tj**):yaitu rentang waktu yang dibutuhkan untuk menyelesaikan suatu operasi pada *job* **j**.

1. *Ready Time* (**rj),** yaitu saat mulai suatu *job* **j** dapat dikerjakan.
2. *DueDate* (**dj**), yaitu batas waktu akhir suatu *job* harus sudah terselesaikan. Bila melewati batas ini, suatu *job* dikatakan terlambat (*tardy*)
3. Waktu penyelesaian (*completion time* = **Cj**) : saat *job* **j**telah selesai dikerjakan.
4. Waktu tinggal (*flow time* = **Fj**) : lamanya *job* **j** berada dilantai pabrik (*shop*). Flow time dihitung sejak *job* siap dijadwalkan sampai *job* selesai dikerjakan.
5. *Lateness* (**Lj**), yaitu merupakan penyimpangan waktu penyelesaian saat *job* terhadap *duedate* *job* yang bersangkutan. *Lateness* dihitung dengan persamaan Lj = Cj - dj.

Lj < 0, saat penyelesaian memenuhi batas akhir (*earliness*).

Lj > 0, saat penyelesaian melewati batas akhir (*tardiness*).

1. *Slack* (**SLj**), yaitu waktu yang tersedia bagi suatu pekerjaan.

SLj = dj - tj.

1. *Tardiness* (**Tj**), yaitu merupakan keterlambatan penyelesaian suatu *job*  terhadap *duedate* *job* tersebut.

Tj = max {0,Lj}.

1. *Makespan* (**Ms**), yaitu waktu dimana semua pekerjaan terakhir selesai (MaxLj).
   * 1. **Aturan Prioritas Penjadwalan (*Priority Dispatching Rules*)**

Penentuan prioritas (*Priority Dispatching*) dari pekerjaan (*job*) dalam pendekatan penjadwalan *heuristic* adalah untuk memilih satu dari beberapa operasi yang mengalami gangguan. Penentuan prioritas tersebut sangat ditentukan dari tujuan yang ingin dicapai.

Beberapa aturan prioritas diberikan menurut French (1980), adalah :

1. **Random**

Pemilihan operasi secara random artinya setiap pekerjaan kemungkinan yang sama untuk dipilih.

1. **Most Work Remaining (MWR)**

Prioritas tertinggi diberikan pada operasi dengan sisa waktu proses terlama. Aturan prioritas ini cocok digunakan untuk menghasilkan jadwal dengan *makespan* terpendek.

1. **First-Come-First-Serve (FCFS)**

Prioritas diberikan kepada pesanan yang tiba lebih dahulu disumber. Aturan ini jika konsumen mementingkan waktu pelayanan.

1. **Earliest Due Date (EDD)**

Prioritas diberikan kepada pesanan dengan *due date* terpendek. Aturan ini dapat berjalan dengan baik jika batas waktu proses *job* relatif sama.

1. **Shortest Processing Time (SPT)**

Prioritas diberikan kepada pesanan dengan waktu proses yang lebih kecil.

1. **Fewest Operation (FO)**

Prioritas tertinggi diberikan pada produk dengan sisa operasi paling sedikit. Aturan ini biasanya mengurangi WIP, waktu penyelesaian rata-rata produk (*mean flow time*) dan rata-rata keterlambatan (*mean lateness*).

1. **Shortest Total Processing Time Remaining (STPT)**

Prioritas tertinggi diberikan kepada pesanan yang mempunyai waktu sisa yang lebih kecil.

1. **Slack Time (ST)**

Prioritas tertinggi diberikan pada produk dengan slack time lebih kecil. ST = dd (*duedate*)- Pd (*Present date*) – TOTR (*Total Operation Remaining*).

1. **Critical Ratio (CR)**

Prioritas diberikan kepada pesanan dengan CR yang lebih kecil. CR adalah perbandingan antara waktu yang tersisa sebelum saat kirim dibagi dengan waktu yang tersisa.

**2.1.4 Penjadwalan *Flow Shop***

Menurut Baker (1974), teknik penjadwalan *Flow Shop* dapat dibagi ke dalam 2 teknik, yaitu:

1. **Teknik Optimasi**
   1. Algoritma Johnson untuk *makespan*.
   2. Algoritma Branch and Bound.
2. **Teknik Heuristik**

Karakteristik *Flow Shop* yang utama adalah adanya aliran mesin yang searah, waktu pengerjaan *job* dapat dilihat dengan membuat *routing*.

* + - 1. **Routing**

Suatu karakteristik utama dari disiplin penugasan adalah tipe mesin yang diperlukan untuk mengerjakan suatu *job* yang disebut *routing*. *Job* yang akan dijadwalkan, ditabulasikan kedalam table matriks *routing* mesin dan table matriks waktu proses.

**Tabel 2.1** Contoh *Routing* Sheet

|  |  |  |  |
| --- | --- | --- | --- |
| *Job* | Operasi | Mesin | Waktu proses |
| A | 1 | M1 | 4 |
| B | 1 | M1 | 6 |
| 2 | M2 | 2 |
| C | 1 | M1 | 5 |
| 2 | M1 | 3 |
| 3 | M2 | 1 |

Perbedaan antara *Job Shop* dan *Flow Shop* adalah pada matriks *routing* mesin dimana pada *job shop* matriks *routing* mesinnya tidak beraturan, sedangkan  *Flow Shop* beraturan. Perbedaannya dapat dilihat pada table-tabel dibawah ini :

**Tabel 2.2** Contoh Matriks Waktu Proses

*Flow Shop*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Operasi** | | |
| **1** | **2** | **3** |
| ***Job1*** | 2 | 3 | 4 |
| ***Job2*** | 2 | 4 | 3 |
| ***Job3*** | 3 | 2 | 4 |

**Tabel 2.3** Contoh Matriks Waktu Proses

*Job Shop*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Operasi** | | |
| **1** | **2** | **3** |
| ***Job1*** | 2 | 3 | 4 |
| ***Job2*** | 2 | 4 | 3 |
| ***Job3*** | 3 | 2 | 4 |

**Tabel 2.4** Contoh Matriks *Routing* Mesin

*Flow Shop*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Operasi** | | |
| **1** | **2** | **3** |
| ***Job1*** | 1 | 2 | 3 |
| ***Job2*** | 1 | 2 | 3 |
| ***Job3*** | 1 | 2 | 3 |

**Tabel 2.5** Contoh Matriks *Routing* Mesin

*Job Shop*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Operasi** | | |
| **1** | **2** | **3** |
| ***Job1*** | 1 | 3 | 2 |
| ***Job2*** | 2 | 3 | 1 |
| ***Job3*** | 3 | 2 | 1 |

**2.2 *Ant Colony System*** **(ACS)**

**2.2.1 Definisi Algoritma *Ant Colony System*** **(ACS)**

*Any Colony System* (ACS) awalnya dikembangkan oleh Marco Dorigo *et. al.* (1996). Algoritma *Ant Colony System* (ACS) adalah algoritma yang didasarkan pada cara kerja semut untuk menentukan jarak terpendek dari sarang menuju sumber makanan. Semut dapat menemukan jarak terpendek dengan memanfaatkan jejak *pheromone* (air liur semut) yang dimanfaatkan sebagai komunikasi tidak langsung antar semut. Ketika semut berjalan, ia meninggalkan *pheromone* dalam jumlah tertentu pada jalur yang dilewatinya. Semut dapat mencium *pheromone* dan ketika memilih jalur mereka cenderung untuk memilih jalur dengan konsentrasi *pheromone* yang lebih besar adalah jarak terpendek.

Saat seekor semut yang terisolasi bergerak secara acak, semut ini akan mengikuti jejak yang telah ditinggalkan sebelumnya yang dapat dideteksi dan mempunyai tingkat probabilitas yang tinggi untuk diikuti dan melanjutkan jejak sebelumnya dengan *pheromone* baru. Tingkah laku kolektif yang muncul disebut dengan tingkah laku *Autocalystic*, dimana semut yang lain dapat mengikuti jejak yang ada dan jejak yang semakin jelas akan memudahkan bagi semut yang lain untuk mengikutinya. Proses ini secara khusus terjadi melalui kumpulan umpan balik yang positif, dimana kemungkinan semut untuk memilih pola meningkat seiring dengan jumlah semut yang sebelumnya mengikuti pola yang sama.



**Gambar 2.4** *Jalur Solusi Semut (Ant Colony System)*

**2.2.2 Analogi ACS dalam Penjadwalan *Flowshop***

Sesuai dengan contoh eksperimen pada gambar 2.5 terdapat pola saat sekelompok semut berjalan (contoh, dari sumber makanan A ke sarang E, dan sebaliknya). Secara tiba-tiba rintangan muncul dan pola menjadi terpotong. Pada posisi B, semut berjalan dari A ke E (atau pada posisi D yang bergerak dengan arah yang berlawanan) harus memutuskan apakah harus bergerak ke kiri atau ke kanan.

Pilihan ini dipengaruhi oleh intensitas jejak *pheromone* yang ditinggalkan oleh semut sebelumnya. Tingkat *pheromone* yang lebih tinggi pada pola sebelah kanan memberikan semut rangsangan yang lebih kuat dan kemungkinan yang lebih tinggi untuk berbelok ke kanan. Semut pertama mencapai titik B atau D mempunyai kemungkinan yang sama untuk belok ke kiri atau ke kanan (karena tidak terdapat *pheromone* pada dua pola alternatif tersebut).

Karena pola BCD lebih pendek dibandingkan dengan pola BHD, semut pertama yang mengikuti ini akan mencapai D sebelum semut yang mengikuti pola BHD. Hasilnya adalah semut yang bergerak dari E ke D akan mendapatkan jejak yang lebih jelas pada pola DCB, karena setengah dari semut tersebut yang memilih untuk mendekati rintangan melalui DCBA dan dengan segera akan sampai melalui BCD, mereka akan melalui pola memilih pola DCB dibandingkan pola DHB. Sebagai Konsekuensi, jumlah semut yang mengikuti pola BCD per unit waktu akan lebih banyak dibandingkan dengan jumlah semut yang mengikuti pola BHD.

Hal ini menyebabkan jumlah *pheromone* pada pola yang lebih pendek akan muncul lebih cepat dibandingkan dengan pola yang lebih jauh, dan oleh karena itu kemungkinan semut yang memilih pola yang diikuti mempunyai bias terhadap pola yang lebih pendek. Hasil akhir yang akan dipilih secara cepat akan ditunjukkan pada pola yang lebih pendek.

Algoritma yang akan dibahas pada bagian selanjutnya adalah model yang berasal dari kumpulan kehidupan nyata semut. Selanjutnya hal ini disebut dengan Sistem Semut dan algoritma yang akan dibahas dikenal dengan algoritma semut. Karakteristik semut asli untuk ketika sedang bergerak dari satu titik ke titik tujuan dapat dilihat pada kedua ilustrasi gambar berikut ini.



**Gambar 2.5** *Contoh Karakteristik semut (1)*

Contoh karakteristik semut asli (1) :

* 1. Semut-semut mengikuti jalur dari titik A ke titik E.
  2. Sebagai rintangan diletakkan hambatan ; semut-semut memilih untuk bergerak memutari salah satu sisi dengan probabilitas yang sama.
  3. Dijalur yang lebih pendek ditinggalkan lebih banyak *pheromone*.

Oleh karena itu, bahwa metafora kumpulan semut dapat digunakan untuk menjelaskan model ini. Sesuai dengan gambar 2.6 yang merupakan interpretasi sistem semut pada situasi gambar 2.6. Untuk menyempurnakan hal ini, misalkan jarak antara D dan H, diantara B dan H, dan diantara B dan D melalui C sama dengan 1, dan posisi C adalah setengah jalan antara D dan B. Sekarang hal ini dipertimbangkan sebagai interval yang berlainan pada waktu : t = 0, 1, 2

Misalkan bahwa 30 semut yang baru datang ke B dari A, dan 30 semut ke D dari E pada tiap unit waktu, kecepatan gerak semut pada kecepatan 1 per unit waktu, dan saat bergerak semut akan meninggalkan jejak pheromone pada intensitas 1, dan membuat hal ini menjadi lebih mudah, penguapan secara sempurna dan berurutan terjadi pada pertengahan interval waktu (t+1, t+2).

Pada saat t = 0 belum terdapat jejak, tetapi 30 semut berada di B dan 30 di D. Pilihan mereka tentangt jalan mana yang akan ditempuh adalah acak. Selanjutnya rata-rata 15 semut dari setiap bagian akan bergerak ke arah H dan 15 ke arah C.

Pada t = 1, 30 semut baru akan muncul ke B dari A akan menemukan jejak pada intensitas 15 dan pola mengarah ke H, yang ditinggalkan oleh 15 semut yang bergerak dari B, dan jejak pada intensitas 30 pada pola C, akan mencapai kumpulan jejak yang ditinggalkan oleh 15 semut yang bergerak dari B dan 15 semut yang mencapai B datang dari D melalui C. Kemungkinan dalam memilih pola selanjutnya bisa menjadi berat sebelah dan sejumlah semut diharapkan bergerak kearah C akan meningkat dua kali ke arah H : 20 versus 10. Hal yang sama berlaku untuk 30 semut pada D yang datang dari E.

Proses berlanjut sampai semua semut akan memilih pola yang paling pendek.



**Gambar 2.6** *Contoh Karakteristik semut (2)*

Contoh karakteristik semut asli (2)

* 1. Graf dengan jarak-jaraknya
  2. Pada saat t = 0 tidak ada jejak dijalur-jalur graf ; oleh karena itu semut-semut akan memilih apakah akan belok kekiri atau kekanan dengan probabilitas yang sama.
  3. Pada saat t = 1 jejak lebih banyak pada jalur yang lebih pendek, karena itu jalur tersebut akan ditinggalkan lebih banyak *pheromone*.

Ide yang muncul adalah jika kita memberikan nilai pada semut yang harus memilih diantara beberapa pola, yang secara kuat dipilih berdasarkan semut yang sebelumnya (dengan tingkat jejak tertinggi) dipilih dengan *probabilitas* yang lebih tinggi. Selanjutnya tingkat jejak yang lebih tinggi sejalan dengan pola yang pendek.

Karakteristik semut tersebut digunakan oleh *Ant Colony System*. Tetapi pada *Ant Colony System* dimodifikasi dengan mengubah bahwa pada awalnya setiap jalur yang ada memiliki sejumlah *pheromone* awal, semut buatan akan memilih jalur-jalur yang ada dengan probabilitas yang tergantung pada jarak dan intensitas *pheromone*.

Marco Dorigo *et. al.* (1996) mengatakan bahwa *Ant Colony System* adalah algoritma *heuristik* yang serba guna untuk memecahkan berbagai masalah optimasi. Algoritma ACS memiliki karakteristik sebagai berikut :

1. Serba guna (*versatile*), dapat dipakai untuk memecahkan masalah dengan versi yang sama, seperti TSP dan *Asymetric Travelling Salesman Problem* (ATSP).
2. Sempurna (*robust*), dapat diterapkan untuk memecahkan dengan hanya perubahan sedikit terhadap masalah optimasi yang lain, seperti *Quadratic Assigment Problem* dan *Job shop Scheduling Problem* (JSP).
3. Pendekatan yang berbasis populasi.

**2.2.3 Algoritma *Ant Colony System* (ACS) pada Penjadwalan *flowshop***

Untuk menyelesaikan permasalahan algoritma *Ant Colony System*, berikut ini adalah rumus-rumus dasar yang dipakai dalam algoritma (rumus-rumus mengacu pada masalah TSP).

  (2.1)

Keterangan :

 **:** Probabilitas semut ke-*k* untuk pergi dari node atau kota *i* ke node *j* pada saat *t*.

** :** Intensitas *pheromone* pada jalur node *i* dan *j* pada saat *t*.

** :** Visibilitas/desirability untuk pergi dari node *i* ke node *j*.

(2.2)

** :** Jumlah *pheromone* yang menghubungkan node *i* ke node *j*.

 **:** Parameter yang menyatakan kepekaan terhadap jejak 

 **:** Parameter yang menyatakan kepekaan terhadap *desirability* .

 **:** Jarak dari node *i* ke node *j*.

 **:** Himpunan node yang diperbolehkan dilewati dan boleh dilewati.

Setelah didapatkan jalur terpendek, maka jalur tersebut akan di *update*. Jalur tersebut di *update* dengan tujuan untuk mengurangi jumlah *pheromone* pada semua sisi sebesar dan menambah jumlah *pheromone* pada sisi yang memberikan jarak terpendek dari satu node ke node berikutnya sebesar . Sedangkan sisi yang lain tidak mendapatkan penambahan *pheromone*. Berikut ini persamaan untuk meng*update* jumlah *pheromone*.

 (2.3)

Keterangan :

 : Parameter eveporasi/penguapan (0<<1).

 : Penambahan jejek *pheromone*.

 : Intensitas *pheromone* pada saat *t + 1*.

Ada pula yang disebut *Tabu List*  yaitu sebuah matriks berukuran (jumlah semut)  jumlah node. *Tabu List* ini berfungsi untuk sebagai memory semut, kemana saja semut tersebut sudah bergerak (node mana saja) sehingga jalur mana saja yang dilalui setiap semut dapat diketahui.

Algoritma umum ACS adalah sebagai berikut (mengacu pada masalah TSP):

Langkah 1 Inisialisasi

*t = 0 ; NC = 0*

untuk setiap jalur (*ij*) berikan *pheromone* awal 

Langkah 2 Tempatkan  semut di node awal.

Langkah 3 Untuk semua semut.

Hitung probabilitas yang ada dengan persamaan (2.1)

Langkah 4 Hitung probabilitas komulatif .

Langkah 5 Bangkitkan bilangan random.

Langkah 6 Untuk setiap semut hitung jalur yang telah ditempuh.

Langkah 7 Untuk setiap jalur  terbaik lakukan perbaikan *pheromone* dengan menggunakan persamaan (2.3)

Langkah 6 Jika *NC* < *NCmaks*.

Maka kosongkan semua *TabuList*.

Kembali ke langkah 2.

Jika tidak, tampilkan jadwal yang memberikan nilai *makespan*.