SKRIPSI

ACO FLOWSHOP

Kevin Jonathan

NPM: 2014730020

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2019

UNDERGRADUATE THESIS

ACO FLOWSHOP

Kevin Jonathan

NPM: 2014730020

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2019

LEMBAR PENGESAHAN

ACO FLOWSHOP

Kevin Jonathan

NPM: 2014730020

Bandung, 10 Mei 2019

Menyetujui,

Pembimbing

Dr.rer.nat. Cecilia Esti Nugraheni

Ketua Tim Penguji

Luciana Abednego, M.T.

Anggota Tim Penguji

Rosa De Lima, M.Kom.

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

ACO FLOWSHOP

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 10 Mei 2019

Meterai
Rp. 6000

Kevin Jonathan
NPM: 2014730020

ABSTRAK

Lorem ipsum dolor sit amet, ex quo duis discere facilis, gloriatur democritum mea ex. Eu periculis voluptatum assueverit eam, in mea oblique adipisci hendrerit, illud aliquando ne ius. Cu invidunt constituam vix, vidisse moderatius reprehendunt his cu. Movet graece ne sit.

Cu omnis homero omnium nec, eu ubique mollis vivendo eos. Et soleat mandamus scriptorem duo. Ne eos alii euripidis sententiae. Soleat quaestio sit id, eam delenit fuisset assueverit an. In mei laboramus voluptatum. Usu cu possim impetus, pri inermis consulatu et, an affert abhorreant vis.

Usu an mediocrem dissentiunt delicatissimi. Cum dolores propriae cu. Ad sonet officiis perpetua vel, at legimus luptatum eos, pro dictas invidunt suscipiantur te. Ut movet dicit eleifend eam, vis ei graece splendide, est nihil splendide ex. Ad minim abhorreant cum, usu ut voluptua intellegat. Et his feugait epicurei, mazim nihil cu vix, nec amet dicam volumus ut.

Kata-kata kunci: Ayam, Den, Lapeh

ABSTRACT

Lorem ipsum dolor sit amet, ex quo duis discere facilis, gloriatur democritum mea ex. Eu periculis voluptatum assueverit eam, in mea oblique adipisci hendrerit, illud aliquando ne ius. Cu invidunt constituam vix, vidisse moderatius reprehendunt his cu. Movet graece ne sit.

Cu omnis homero omnium nec, eu ubique mollis vivendo eos. Et soleat mandamus scriptorem duo. Ne eos alii euripidis sententiae. Soleat quaestio sit id, eam delenit fuisset assueverit an. In mei laboramus voluptatum. Usu cu possim impetus, pri inermis consulatu et, an affert abhorreant vis.

Usu an mediocrem dissentiunt delicatissimi. Cum dolores propriae cu. Ad sonet officiis perpetua vel, at legimus luptatum eos, pro dictas invidunt suscipiantur te. Ut movet dicit eleifend eam, vis ei graece splendide, est nihil splendide ex. Ad minim abhorreant cum, usu ut voluptua intellegat. Et his feugait epicurei, mazim nihil cu vix, nec amet dicam volumus ut.

Keywords: Chicken, Den, Lapeh

Skripsi ini dipersembahkan untuk mamah dan papah

KATA PENGANTAR

Lorem ipsum dolor sit amet, ex quo duis discere facilis, gloriatur democritum mea ex. Eu periculis voluptatum assueverit eam, in mea oblique adipisci hendrerit, illud aliquando ne ius. Cu invidunt constituam vix, vidisse moderatius reprehendunt his cu. Movet graece ne sit.

Cu omnis homero omnium nec, eu ubique mollis vivendo eos. Et soleat mandamus scriptorem duo. Ne eos alii euripidis sententiae. Soleat quaestio sit id, eam delenit fuisset assueverit an. In mei laboramus voluptatum. Usu cu possim impetus, pri inermis consulatu et, an affert abhorreant vis.

Usu an mediocrem dissentiunt delicatissimi. Cum dolores propriae cu. Ad sonet officiis perpetua vel, at legimus luptatum eos, pro dictas invidunt suscipiantur te. Ut movet dicit eleifend eam, vis ei graece splendide, est nihil splendide ex. Ad minim abhorreant cum, usu ut voluptua intellegat. Et his feugait epicurei, mazim nihil cu vix, nec amet dicam volumus ut.

Bandung, Mei 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Penjadwalann produksi merupakan aktivitas yang tidak terpisahkan dalam suatu perusahaan *manufakturing*. Penjadwalan (*scheduling*) sendiri didefinisikan sebagai suatu proses pengalokasian sumber daya atau mesin-mesin yang ada untuk melaksanakan tugas-tugas yang ada dalam suatu waktu tertentu (Baker, 1974). Sedangkan yang dimaksud dengan proses produksi adalah serangkaian langkah-langkah yang digunakan untuk mentransformasikan *Input* menjadi *Output*.

Proses penjadwalan *Flow Shop* adalah salah satu metode penjadwalan produksi di mana urutan mesin yang digunakan untuk setiap proses dalam seluruh pekerjaan harus sama. Dalam penelitian - penelitian penjadwalan sebelumnya hanya difokuskan pada satu kriteria saja (*single*) namun pada penelitian kali ini akan menggunakan lebih dari satu kriteria (*multiple*). Banyak algoritma yang dapat digunakan untuk menentukan urutan pengerjaan pekerjaan dalam proses penjadwalan produksi *Flow Shop*. Salah satu algoritma yang dapat digunakan dalam proses penjadwalan produksi *Multi Objective Flow Shop* adalah algoritma *Ant Colony Optimization*. Algoritma *Ant Colony Optimization* adalah algoritma yang mengadopsi perilaku koloni semut yang dikenal sebagai sistem semut. Algoritma ini menyelesaikan permasalahan berdasarkan tingkah laku semut dalam sebuah koloni yang sedang mencari sumber makanan.

Penelitian ini dibuat untuk mempelajari, mengaplikasikan, serta mengukur kinerja Algoritma *Ant Colony Optimization* pada proses penjadwalan *Multi Objective Flow Shop Scheduling* (MOFSP). Pada skripsi ini juga akan dibuat perangkat lunak yang dapat menerima n job yang masing-masing terdiri atas m buah operasi dan m buah mesin. Setiap operasi hanya ditangani oleh sebuah mesin dan setiap mesin hanya bisa menangani satu operasi. Urutan operasi dari setiap job adalah sama.

1.2 Rumusan Masalah

1. Apa itu penjadwalan *Multi Objective Flowshop Scheduling* (MOFSP) ?
2. Apa itu algoritma *Ant Colony Optimization* (ACO) ?
3. Bagaimana cara kerja dan implementasi algoritma *Ant Colony Optimization* (ACO) dalam menyelesaikan permasalahan MOFSP ?
4. Bagaimana kinerja algoritma *Ant Colony Optimization* (ACO) dalam menyelesaikan permasalahan MOFSP ?

1.3 Tujuan

1. Menjelaskan penjadwalan *Multi Objective Flowshop Scheduling* (MOFSP).
2. Menjelaskan algoritma *Ant Colony Optimization* (ACO) .

3. Menampilkan cara kerja dan implementasi algoritma *Ant Colony Optimization* (ACO) dalam menyelesaikan permasalahan MOFSP .
4. Mengetahui kinerja algoritma *Ant Colony Optimization* (ACO) dalam menyelesaikan permasalahan MOFSP dengan bantuan *benchmark* tertentu.

1.4 Batasan Masalah

Batasan dan asumsi untuk penelitian ini adalah :

1. Waktu proses dari setiap pekerjaan telah diketahui dan bernilai tetap
2. Semua penyelesaian proses dari pekerjaan mengikuti alur proses yang sama dan sistematis
3. Pengerjaan proses dari pekerjaan-pekerjaan yang ada tidak dapat saling mendahului
4. Eksperimen dilakukan dengan sampel data kasus milik *Tailard Benchmark* flow shop dengan jumlah mesin yang beragam.
5. Pengukuran tingkat performansi dari algoritma menggunakan perbandingan nilai makespan dan nilai idle mesin dalam menjalankan suatu job.

1.5 Metodologi

Metodologi penyelesaian masalah dalam penelitian ini adalah :

1. Studi Literatur

   Mencari referensi dari sumber-sumber tertentu untuk memperdalam pemahaman mengenai cara kerja proses penjadwalan flow shop dan cara kerja algoritma ant colony agar kemudian dapat mengaplikasikan algoritma tersebut pada proses penjadwalan flow shop.

2. Analisa Kasus

   Menentukan cara pengaplikasian algoritma ant colony untuk optimisasi penjadwalan flow shop. Menentukan data masukan dan data keluaran yang dibutuhkan oleh perangkat lunak. Menentukan fungsi-fungsi apa saja yang dibutuhkan perangkat lunak.

3. Pengembangan perangkat lunak

   Membentuk struktur kelas dari perangkat lunak. Mendesain interface yang sesuai untuk perangkat lunak. Membangun perangkat lunak untuk optimisasi penjadwalan flow shop dengan algoritma ant colony. Melakukan pengujian fungsional pada perangkat lunak.

4. Eksperimen

   Melakukan proses optimisasi dengan menggunakan perangkat lunak pada beberapa sampel kasus flow shop. Mencatat dan mengolah data hasil proses optimisasi untuk mengukur performa perangkat lunak. Mengukur tingkat keoptimalan dari proses optimisasi yang dilakukan oleh perangkat lunak.

5. Pengambilan kesimpulan

   Mengambil kesimpulan-kesimpulan yang bisa didapatkan dari hasil eksperimen. Melakukan dokumentasi dari skripsi ini.

1.6 Sistematika Pembahasan

Sistematika penulisan karya tulis ini adalah sebagai berikut :

1. Bab 1 : Pendahuluan untuk mendefinisikan masalah yang akan dibahas, alasan pemilihan topik dan usulan solusi terhadap masalah yang ada.
2. Bab 2 : Dasar teori yang digunakan dalam penelitian ini. Pembahasan permasalahan flow shop. Pembahasan cara kerja dari algoritma ant colony. Pembahasan cara pengaplikasian algoritma ant colony untuk optimisasi proses penjadwalan flow shop.
3. Bab 3 : Analisis masalah yang akan dilakukan pada penelitian ini. Pembahasan cara penerapan algoritma ant colony dan rancangan awal dari perangkat lunak.
4. Bab 4 : Perancangan perangkat lunak. Detil informasi mengenai perangkat lunak yang telah dibuat. Struktur kelas dan desain antarmuka grafis dari perangkat lunak yang telah dibuat.
5. Bab 5 : Implementasi dan pengujian. Hasil implementasi algoritma ant colony pada perangkat lunak. Penjelasan cara penggunaan perangkat lunak. Hasil pengujian dan eksperimen kasus flow shop pada perangkat lunak
6. Bab 6 : Kesimpulan dan saran. Hal-hal yang dapat disimpulkan dari penelitian ini. Saran pengembangan yang dapat dilakukan pada penelitian selanjutnya.

BAB 2

LANDASAN TEORI

2.1 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).

Tujuan penjadwalan secara umum menurut Baker (1974) [1] 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).

Menurut Baker [2] masalah penjadwalan muncul karena keterbatasan :

- Waktu
- Tenaga Kerja
- Jumlah Mesin
- Sifat dan syarat pekerja.

2.1.1 Istilah-Istilah dalam Penjadwalan

Bedworth [3] menyebutkan beberapa istilah dalam penjadwalan yang perlu dijelaskan adalah sebagai berikut:

- Waktu Pemrosesan (*Processing Time*)
  Lamanya waktu yang dibutuhkan untuk menyelesaikan satu aktivitas pekerjaan.
- Makespan
  Keseluruhan waktu yang dibutuhkan untuk menyelesaikan aktivitas pekerjaan.

- Waktu Aliran (Flow Time)
  Rentang waktu antara satu pekerjaan dapat dimulai sampai saat dimana pekerjaan tersebut selesai dikerjakan. Sehingga waktu aliran sama dengan waktu pemrosesan ditambah dengan waktu menunggu sebelum diproses.
- Waktu Penyelesaian (Completion Time)
  Waktu dimana tugas terakhir dari satu pekerjaan tertentu selesai dikerjakan.
- Batas Waktu (Due Time)
  Batas waktu penyelesaian untuk suatu pekerjaan (order) dan jika suatu pekerjaan (order) melewati batas waktu tersebut akan dikenakan denda (penalty)
- Tardiness
  Merupakan keterlambatan penyelesaian suatu job terhadap batas waktu penyelesaian (due time) job tersebut.

2.1.2 Klasifikasi Masalah Penjadwalan

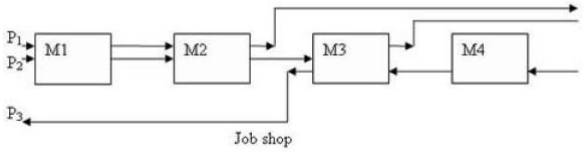
Permasalahan penjadwalan dapat dilihat dari :

1. Mesin :
   - Mesin Tunggal
   - Mesin ganda (2 mesin)
   - M mesin
2. Aliran proses
   - *Job Shop*
   - *Flow Shop*
3. Pola Kedatangan
   - Statis
   - Dinamis

2.2 Penjadwalan Flow Shop

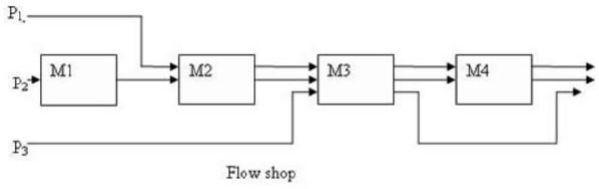
Seperti yang telah disebutkan di atas. Penjadwalan berdasarkan aliran proses terdiri atas dua penjadwalan yaitu :

1. *Job Shop*
   Setiap proses pekerjaan yang dikerjakan tidak memiliki lintasan operasi yang tetap, artinya setiap job yang datang memiliki proses operasi yang berbeda dan dapat mengalami pengulangan lintasan. Gambar 2.1 merupakan gambaran dari pola kerja job shop. Pada gambar tersebut dijelaskan bahwa P1 akan jalan melalui mesin 1 ke mesin 2. Lalu pada P3 di jelaskan job dimulai dari mesin ke 4 lalu ke mesin 3, maka dapat disimpulkan setiap job memiliki proses operasi yang berbeda.



Gambar 2.1: Job Shop

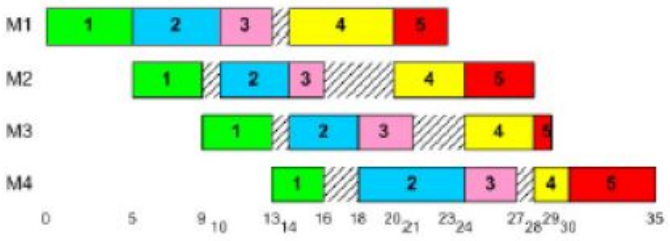
2. Flow shop
   Setiap pekerjaan yang akan diproses memiliki lintasan produksi yang searah, dan setiap operasi yang dilalui oleh setiap job memiliki urutan proses yang sama tanpa mengalami pengulangan lintasan (semua job hanya diproses sekali pada tiap mesin). Gambar 2.2 merupakan gambaran dari pola kerja flow shop.



Gambar 2.2: Flow Shop

2.2.1 Definisi Secara Umum

Permasalahan flow shop scheduling adalah bagaimana menentukan urutan pengerjaan untuk sejumlah n job dengan menggunakan sejumlah m mesin dalam urutan proses yang sama dan setiap job melewati setiap mesin dalam satu waktu tertentu. Masing-masing job memiliki lama pengerjaan yang berbeda. Dalam pemrosesan, setiap job membutuhkan sebanyak m operasi (setiap mesin satu operasi). Setiap job tidak dapat saling mendahului dan setiap mesin tidak akan diam saat job siap diproses. Tujuan utama dari flow shop scheduling adalah menentukan makespan minimum untuk mengerjakan n job dengan menggunakan m mesin. Nilai makespan dapat berubah-ubah berdasarkan urutan pengerjaan job yang dilakukan, oleh karena itu urutan pengerjaan akan sangat berpengaruh terhadap nilai makespan. Pada skripsi ini juga akan ditambahkan satu *objective* lainnya dalam mengukur performa algoritma *ant colony* terhadap permasalahan flow shop. Objective tersebut adalah meminimasi terjadinya tardiness dalam suatu penyelesaian jobİlustrasi dari Flow Shop Scheduling dapat dilihat pada gambar 2.3 berikut :



Gambar 2.3: Ilustrasi Flow Shop

Gambar 2.3 merupakan ilustrasi proses flowshop scheduling. Pada gambar tersebut urutan pengerjaan job adalah 1,2,3,4, dan 5. Masing-masing job memiliki 4 proses operasi dan masing-masing proses operasi tersebut dikerjakan oleh mesin yang berbeda. Saat proses operasi 1 dari job 1 telah selesai, mesin 2 akan mengerjakan proses operasi 2 dari job 1, dan mesin 1 akan mengerjakan proses operasi 1 dari job 2. Saat proses operasi 2 dari job 1 sudah selesai, mesin 2 akan mengerjakan proses operasi 3 dari job 1, akan tetapi mesin 2 yang seharusnya mengerjakan proses operasi 2 dari job 2 akan mengalami idle, karena proses operasi 1 dari job 2 belum selesai dikerjakan di mesin 1.

Idle adalah kondisi dimana mesin tidak melakukan proses operasi apapun, baik itu karena adanya proses operasi yang akan dilakukan pada mesin tersebut belum selesai diproses di mesin selanjutnya atau karena masih ada proses operasi yang dilakukan pada mesin selanjutnya. Idle

akan mempengaruhi besar kecilnya nilai makespan yang akan dihasilkan. Semakin banyak idle, maka makespan umumnya akan semakin besar, serta sebaliknya dimana semakin sedikit idle maka umumnya makespan yang didapat akan semakin kecil. Mesin juga mengalami idle pada saat mesin 1 telah menyelesaikan proses operasi job 3 namun ternyata pada mesin 2 proses job 2 belum selesai. Sehingga job 3 akan disimpan dalam mesin 1 hingga mesin 2 selesai mengerjakan proses operasi job 2.

2.2.2 Karakteristik Penjadwalan Flow Shop

1. Terdapat n job yang tersedia dan siap diproses pada waktu t = 0.
2. Waktu proses tiap job tidak bergantung pada urutan pengerjaan.
3. Terdapat m mesin berbeda, yang tersedia secara kontinu.
4. Operasi-operasi individual job pada tiap mesin tidak dapat dipecah-pecah.

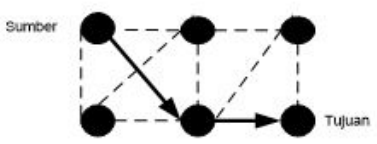
2.3 Multi Objective Flow Shop Scheduling

Pada dasarnya MOFSP adalah bagian dari *flow shop* itu sendiri. MOFSP menjelaskan kriteria optimasi yang akan diselesaikan menggunakan algoritma *Ant Colony Optimization*. Kriteria tersebut bersifat jamak *Multi Objective*. Kriteria tersebut misalnya saja minimasi waktu proses keseluruhan, minimasi waktu tunggu job, minimasi waktu nganggur mesin, dan sebagainya. Kriteria optimasi yang akan diselesaikan pada skripsi ini adalah menentukan makespan minimum dan minimasi waktu *idle* atau *tardiness*.

2.4 Ant Colony Optimization

Any Colony Optimization (ACO) awalnya dikembangkan oleh Marco Dorigo et. al. (1996). Algoritma ini 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 Optimization

Marco Dorigo et. al. (1996) mengatakan bahwa Ant Colony Optimization adalah algoritma heuristik yang serba guna untuk memecahkan berbagai masalah optimasi. Algoritma ACO 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).

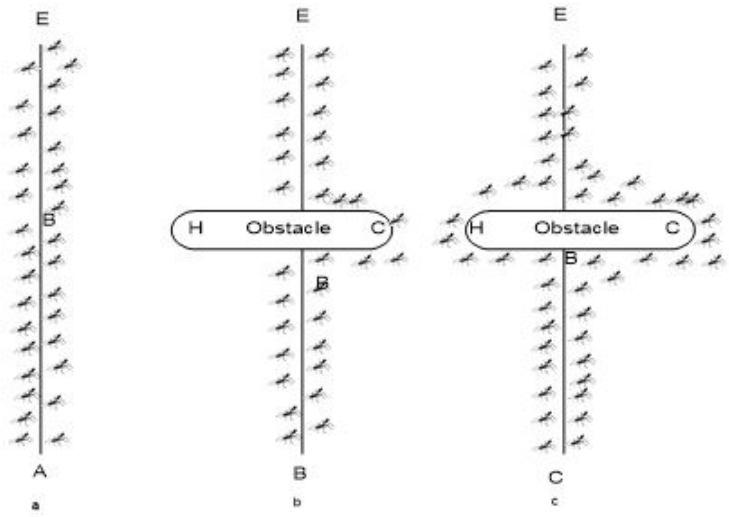
3. Pendekatan yang berbasis populasi.

2.4.1 Algoritma Ant Colony Optimization pada penjadwalan Flow Shop

- Karakteristik Semut
  Sesuai dengan contoh 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.

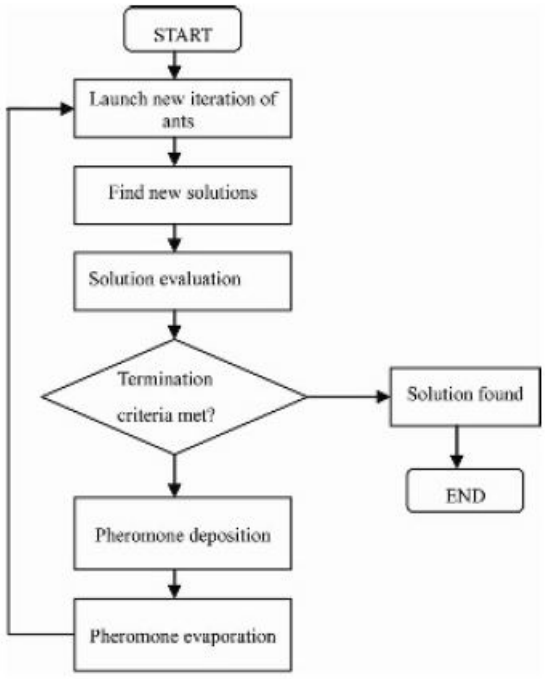


Gambar 2.5: Karakteristik semut

Penjelasan karakteristik semut :

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

- Cara Kerja Algoritma Berikut adalah flow chart dari algoritma ant colony.



Gambar 2.6: Flowchart algoritma ant colony optimization

Berdasarkan flowchart 2.6, dalam menjalankan proses pelatihan dengan algoritma ant colony maka perlu diperrsiapkan jalur-jalur yang ada perlu. Jalur-jalur tersebut merupakan kemungkinan solusi-solusi yang mungkin dipilih oleh semut. Kemungkinan dipilihnya suatu jalur akan ditentukan berdasarkan nilai feromon, oleh karena itu perlu dibentuk pula tempat penyimpanan feromon untuk setiap jalur yang mungkin dilalui.

Pada awalnya, algoritma ant colony akan membentuk sekumpulan semut, semut-semut tersebut akan disebar secara acak pada jalur-jalur yang ada. Proses penyebaran semut tersebut akan memperhatikan nilai feromon dari masing-masing jalur. Semakin besar nilai feromon dari suatu jalur, maka semakin besar pula kemungkinan dipilihnya jalur tersebut untuk dilalui semut. Jalur-jalur yang dipilih oleh semut akan disimpan dan dibandingkan dengan jalur-jalur yang telah dipilih oleh semut-semut lainnya.

Dari jalur-jalur yang telah dipilih akan dibentuk nilai feromonnya berdasarkan tingkat keoptimalan dari jalur tersebut. Nilai feromon-feromon tersebut akan ditambahkan pada jalur-jalur yang bersangkutan untuk memperbesar kemungkinan dipilihnya jalur tersebut sesuai dengan tingkat keoptimalannya. Semakin optimal suatu jalur, semakin besar pula jumlah feromon yang akan ditambahkan pada suatu jalur. Tidak lupa pula, proses evaporasi feromon akan dilakukan. Proses evaporasi feromon ini akan mengurangi jumlah feromon yang disimpan. Proses evaporasi ini bertujuan untuk mengurangi jumlah feromon dari jalur-jalur yang dianggap kurang optimal. Proses-proses tersebut akan terus dilakukan hingga suatu kondisi berhenti ditemui.

```
membentukDataFeromon()
solusiTerbaik = null
while(kondisi berhenti belum ditemui)
    koloniSemut = bentuk kumpulan semut kosong
    for(1 sampai jumlah semut yang akan disebar)
        solusiLokal = pilihSolusiSecaraAcakBerdasarkanNilaiFeromon()
        If(solusiLokal solusi yang valid)
            (opsional) localSearch(solusiLokal)
            if(hasil(solusiLokal) lebih baik dari hasil(solusiTerbaik))
                simpan solusiLokal pada solusiTerbaik
            end if
            simpan solusiLokal pada koloniSemut
        end if
    end for
    updateNilaiFeromon(koloniSemut)
end while
output : solusiTerbaik
```

Gambar 2.7: Pseudocode algoritma ant colony optimization

Dari flow chart 2.6, dibentuklah pseudocode di atas. Baris pertama dari pseudocode tersebut merupakan proses pembentukan data feromon. Proses pembentukan data feromon tersebut akan memperhatikan kasus yang ingin dicari hasil optimalnya. Nilai masingmasing jalur dan cara mengartikan nilai feromon tersebut diatur berdasarkan kasus yang ingin dicari hasil optimalnya. Jumlah feromon pada masing-masing jalur pada awalnya bernilai sama. Setelah data feromon selesai dibuat, proses pelatihan dengan menggunakan algoritma ant colony dimulai. Baris ketiga pada pseudocode menunjukkan kondisi berhenti untuk proses optimisasi dari algoritma ant colony.

Seluruh potongan kode di dalam kondisi while tersebut merupakan satu fase pelatihan dari algoritma ant colony. Fase pelatihan tersebut terdiri dari proses pembentukan jalur, penentuan solusi optimal, dan perubahan nilai feromon (penambahan dan evaporasi feromon). Fase pelatihan dari algoritma ant colony bertujuan untuk membentuk data feromon yang mampu

menghasilkan solusi yang paling optimal. Pada setiap fase pelatihan ini akan dibentuk sejumlah semut yang bertugas untuk menyimpan solusi acak/solusi lokal yang telah dipilih pada suatu fase pelatihan. Semut-semut ini akan dianggap memilih sebuah jalur/solusi secara acak berdasarkan nilai yang disimpan pada data feromon. Jika solusi acak tersebut lebih baik dari solusi optimal yang diketahui pada saat ini, maka solusi acak tersebut akan dianggap sebagai solusi optimal yang baru.

Pada saat pemilihan solusi secara acak, perlu dipastikan bahwa solusi tersebut merupakan solusi yang valid dari suatu kasus. Pada saat pemilihan solusi kita juga dapat melakukan proses local search. Proses ini bertugas untuk membandingkan solusi yang dibentuk dengan solusi yang mirip. Jika solusi lain tersebut lebih baik, maka solusi yang dipilih akan diganti menjadi solusi yang mirip tersebut. Proses ini bersifat opsional, karena tidak semua permasalahan dapat diaplikasikan dengan proses ini.

Setelah semut-semut telah selesai memilih solusi secara acak, proses perubahan nilai feromon akan dilakukan. Proses perubahan tersebut mencakup proses penambahan dan evaporasi nilai feromon. Proses penambahan nilai feromon akan memperhatikan tingkat keoptimalan dari solusi yang dipilih suatu semut. Semakin optimal suatu solusi, semakin banyak pula nilai feromon yang ditambahkan pada jalur/solusi tersebut. Setelah proses optimisasi selesai, solusi yang paling optimal akan diberikan sebagai data keluaran.

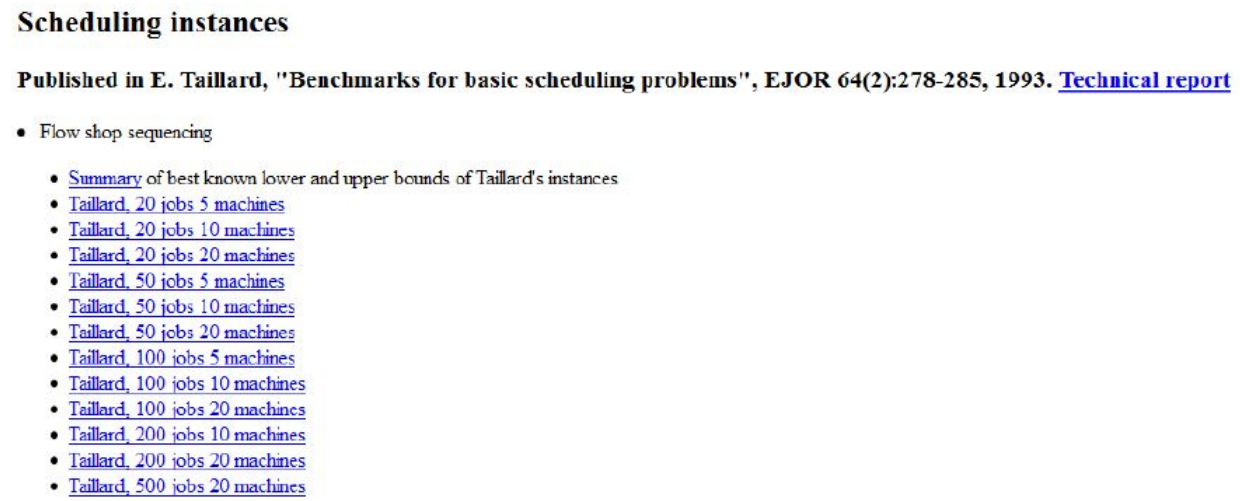
2.5 Tailard Benchmark

Dalam melakukan pengujian tingkat keoptimalan dari suatu algoritma, pengujian sebaiknya dilakukan dengan menggunakan data kasus standar. Data kasus standar tersebut akan dianggap sebagai suatu kasus unik yang memerlukan algoritma khusus untuk proses optimisasinya. Data standar tersebut akan digunakan sebagai pembanding dalam proses optimisasi. Dengan menggunakan data standar, penilaian kualitas dari suatu algoritma optimisasi dapat dilakukan. Dalam kasus penelitian ini, data kasus standar tersebut akan menggunakan data dari suatu benchmark yang bernama *Taillard Benchmark*. *Benchmark* tersebut dapat diakses lewat url berikut : [1] .

Taillard benchmark adalah sebuah web hosting yang menyediakan kumpulan data yang digunakan sebagai input dan akan menghasilkan output suatu permasalahan flow shop scheduling. Input dan output tersebut digunakan sebagai standar dalam pengujian algoritma penyelesaian flow shop scheduling sesuai dengan yang didefinisikan oleh Taillard [9]. Hal ini akan mempengaruhi penilaian suatu algoritma dalam menyelesaikan permasalahan flow shop scheduling apakah algoritma tersebut sudah efektif atau tidak.

Dalam data input yang digunakan pada permasalahan flow shop scheduling, selain banyaknya job dan mesin, ada pula initial seed yang dapat digunakan menggunakan random generator. Initial Seed ini berfungsi dalam membuat variasi waktu proses dari tiap job yang ada, sehingga kumpulan job yang ada akan memiliki waktu proses yang berbeda pada tiap prosesnya. Seperti yang telah diketahui sebelumnya, tujuan utama dari pencarian solusi permasalahan flow shop scheduling adalah pencarian waktu makespan yang paling kecil. Taillard benchmark Framework ini akan menyediakan data output yang berupa upper bound dan lower bound solusi permasalah flow shop scheduling. Lower bound menunjukkan batas makespan paling minimum sedangkan Upper bound menunjukkan batas makespan paling maksimum. Selain itu, Taillard benchmark juga akan memberikan data-data detail dari proses-proses yang dilakukan.Tampilan dari Tailard benchmark sebagai berikut :

[1] http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html



Scheduling instances

Published in E. Taillard, "Benchmarks for basic scheduling problems", EJOR 64(2):278-285, 1993. Technical report

- Flow shop sequencing
  - Summary of best known lower and upper bounds of Taillard's instances
  - Taillard, 20 jobs 5 machines
  - Taillard, 20 jobs 10 machines
  - Taillard, 20 jobs 20 machines
  - Taillard, 50 jobs 5 machines
  - Taillard, 50 jobs 10 machines
  - Taillard, 50 jobs 20 machines
  - Taillard, 100 jobs 5 machines
  - Taillard, 100 jobs 10 machines
  - Taillard, 100 jobs 20 machines
  - Taillard, 200 jobs 10 machines
  - Taillard, 200 jobs 20 machines
  - Taillard, 500 jobs 20 machines

Gambar 2.8: Tailard Benchmark

Gambar 2.8 memperlihatkan tampilan menu Taillard benchmark yang dimana kita dapat memilih kasus-kasus flow shop scheduling yang hendak digunakan. Pada setiap kasusnya, telah disediakan data-data input berupa jumlah job dan jumlah mesin dan initial seed. Selain itu diberikan pula upper bound dan lower bound dari kasus tersebut untuk menjadi tolak ukur algoritma yang telah kita buat apakah dapat memenuhi standar dari Taillard Benchmark. Taillard membagi jenis kasus-kasus flow shop scheduling berdasarkan jumlah job dan jumlah mesin. Pada tiap jenis kasus Taillard menyediakan 10 (sepuluh) kasus yang dapat dicoba.

```
number of jobs, number of machines, initial seed, upper bound and lower bound :
          20           5   873654221        1278        1232
processing times :
 54 83 15 71 77 36 53 38 27 87 76 91 14 29 12 77 32 87 68 94
 79  3 11 99 56 70 99 60  5 56  3 61 73 75 47 14 21 86  5 77
 16 89 49 15 89 45 60 23 57 64  7  1 63 41 63 47 26 75 77 40
 66 58 31 68 78 91 13 59 49 85 85  9 39 41 56 40 54 77 51 31
 58 56 20 85 53 35 53 41 69 13 86 72  8 49 47 87 58 18 68 28
number of jobs, number of machines, initial seed, upper bound and lower bound :
          20           5   379008056        1359        1290
processing times :
 26 38 27 88 95 55 54 63 23 45 86 43 43 40 37 54 35 59 43 50
 59 62 44 10 23 64 47 68 54  9 30 31 92  7 14 95 76 82 91 37
 78 90 64 49 47 20 61 93 36 47 70 54 87 13 40 34 55 13 11  5
 88 54 47 83 84  9 30 11 92 63 62 75 48 23 85 23  4 31 13 98
 69 30 61 35 53 98 94 33 77 31 54 71 78  9 79 51 76 56 80 72
number of jobs, number of machines, initial seed, upper bound and lower bound :
          20           5  1866992158        1081        1073
processing times :
 77 94  9 57 29 79 55 73 65 86 25 39 76 24 38  5 91 29 22 27
 39 31 46 18 93 58 85 58 97 10 79 93  2 87 17 18 10 50  8 26
 14 21 15 10 85 46 42 18 36  2 44 89  6  3  1 43 81 57 76 59
 11  2 36 30 89 10 88 22 31  9 43 91 26  3 75 99 63 83 70 84
 83 13 84 46 20 33 74 42 33 71 32 48 42 99  7 54  8 73 30 75
number of jobs, number of machines, initial seed, upper bound and lower bound :
          20           5   216771124        1293        1268
```

Gambar 2.9: Contoh kasus Tailard Benchmark

Gambar 2.9 memperlihatkan tampilan salah satu kasus pada flow shop scheduling dengan 20 jobs dan 5 mesin. Pada gambar tersebut kita dapat melihat data waktu proses tiap job pada tiap mesin. Waktu proses ini didapatkan seluruhnya berdasarkan initial seed yang digunakan. Berdasarkan data tersebut, algoritma yang telah dibuat sebelumnya dapat dinilai apakah telah efektif atau belum sesuai dengan upper bound dan lower bound yang diberikan.

BAB 3

ANALISA MASALAH

Pada bab ini akan dibahas mengenai hasil analisa permasalahan flow shop dan segalah hal yang berkaitan dengan analisa permasalahan tersebut. Pada bab ini juga akan dibahas sekilas mengenai rancangan awal dari perangkat lunak yang memungkinkan diaplikasikannya algoritma ant colony pada permasalahan multi objective flow shop.

3.1 Analisa Kasus

Proses penjadwalan Flow Shop adalah salah satu metode penjadwalan produksi di mana urutan mesin yang digunakan untuk setiap proses dalam seluruh pekerjaan harus sama. Pekerjaan-pekerjaan tersebut akan dikerjakan pada mesin-mesin yang ada dengan urutan pengerjaan yang dipilih. Masing - masing proses akan dikerjakan secara sistematis dan terurut. Data-data masukan yang dibutuhkan dalam menjalankan sebuah proses penjadwalan flow shop adalah:

- Banyaknya pekerjaan
- Banyaknya mesin
- Detail waktu pengerjaan setiap pekerjaan

Setiap pekerjaan memiliki waktu proses yang berbeda satu sama lain pada tiap mesin. Variasi urutan pengerjaan yang berbeda biasanya akan menghasilkan waktu pengerjaan / makespan yang berbeda pula. Proses penjadwalan ini penting untuk diperhatikan, karena mampu mempengaruhi lama waktu penggunaan fasilitas dalam suatu proses produksi. Urutan pengerjaan yang baik dapat dicari dengan menggunakan suatu algoritma optimisasi. Diharapkan dengan algoritma optimisasi tersebut, urutan pengerjaan yang menghasilkan makespan / waktu pengerjaan minimum dapat ditemukan.

Salah satu algoritma yang dapat digunakan untuk optimisasi penjadwalan flow shop adalah algoritma ant colony. Tingkat keoptimalan setiap algoritma berbeda-beda, oleh karena itu perlu dibuat sebuah perangkat lunak untuk optimisasi penjadwalan flow shop dengan menggunakan algoritma ant colony. Algoritma ant colony akan menerima data masukan dari suatu kasus flow shop, kemudian akan mencari urutan pengerjaan / solusi yang paling optimal dari kasus tersebut selain itu kita juga dapat mencari kriteria lain yang kita inginkan.

Algoritma ant colony akan mencari urutan pengerjaan yang menghasilkan makespan / waktu pengerjaan yang paling minimum. Algoritma ant colony merepresentasikan pilihan solusi-solusi yang ada sebagai jalur yang akan dilalui oleh semut dan memberikan data feromon sebagai panduan untuk melewati jalur-jalur tersebut. Dengan panduan tersebut, semut-semut tersebut diasumsikan mampu memilih jalur yang paling optimal. Proses optimisasi dengan menggunakan algoritma ant colony perlu memperhatikan beberapa hal.

Algoritma ant colony perlu mengetahui bagaimana cara merepresentasikan suatu solusi sebagai jalur, bagaimana cara menggunakan panduan feromon untuk pemilihan jalur, bagaimana cara membaca dan menyimpan suatu data feromon, serta banyak hal lainnya. Tata cara tersebut mampu mempengaruhi hasil optimisasi dari algoritma ant colony.

3.2 Analisa Algoritma Ant Colony Pada Penjadwalan Flow Shop

3.2.1 Urutan Pengerjaan Proses Sebagai Jalur Semut

Algoritma ant colony akan membantu penentuan urutan pengambilan pekerjaan yang optimal. Oleh karena itu, algoritma ini akan merepresentasikan pilihan urutan pengambilan yang ada sebagai jalur. Jalur-jalur tersebut kemudian akan dilewati oleh semut-semut yang akan mencari solusi / urutan pengerjaan yang optimal. Pilihan urutan pengerjaan yang ada dapat dicari dengan menggunakan fungsi permutasi terhadap jumlah pekerjaan.

Sebagai contoh, jika dimisalkan terdapat 3 buah pekerjaan, maka dengan fungsi permutasi akan didapatkan jalur-jalur urutan pengerjaan (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), dan (3,2,1). Fungsi permutasi ini digunakan dengan mempertimbangkan sifat dari proses penjadwalan flow shop. Masing-masing proses dari pekerjaan pasti akan dikerjakan dan hanya akan dikerjakan sebanyak satu kali.

3.2.2 Rumus - rumus ACO dalam penjadwalan Flow Shop

1. Pemilihan Jalur
   Kemungkinan dipilihnya sebuah jalur dipengaruhi oleh kekuatan feromon pada jalur tersebut. Semakin kuat / banyak feromon pada suatu jalur, maka jalur tersebut akan lebih sering / berkemungkinan lebih besar untuk dipilih. Kemungkinan terpilihnya masing-masing jalur dapat dihitung dengan menggunakan rumus yang terdapat pada rumus berikut :

$$P_k^{xy} = \frac{T_{xy}^{\alpha} \cdot n_{xy}^{\beta}}{\sum_{x \in availeble_z} (T_{xz}^{\alpha} \cdot n_{xz}^{\beta})} \tag{3.1}$$

   Keterangan dari rumus tersebut sebgai berikut :

   P_k^{xy} : nilai kemungkinan terpilihnya jalur xy oleh semut k
   T_{xy} : nilai feromon untuk jalur xy
   n_{xy} : nilai objektif untuk jalur xy
   α : persentase pengaruh nilai feromon
   β : persentase pengaruh nilai objektif

   T menunjukkan kekuatan / jumlah feromon yang terdapat pada suatu jalur, sedangkan n menunjukkan nilai objektif yang dimiliki oleh suatu jalur. Nilai objektif pada suatu jalur menunjukkan pengetahuan / nilai awal mengenai jalur tersebut. Nilai awal tersebut biasanya berupa informasi mengenai jalur tersebut yang diketahui sejak awal dan mampu memberikan pengaruh pada proses pemilihan jalur.

   Contoh informasi yang dapat menjadi nilai objektif adalah panjang jalur, banyaknya belokan, panjang jalur pertama, dan lain-lain. Nilai awal suatu jalur dapat bernilai sama dengan jalur yang lain. Apabila proses pencarian yang dilakukan tidak memiliki informasi apapun atau bersifat blind- search, nilai objektif ini dapat diabaikan. Besar pengaruh dari kekuatan feromon dan nilai objektif terhadap kemungkinan terpilihnya sebuah jalur dapat diatur dengan menggunakan rasio tertentu.

2. Update Nilai Feromon
   Ketika terdapat semut yang menyelesaikan sebuah jalur, feromon pada jalur tersebut akan diubah. Jumlah feromon pada jalur yang dilewati semut tersebut akan ditambah. Dalam proses penambahan feromon, nilai feromon yang baru pada jalur ditentukan dengan rumus berikut:

$$T_{xy} \leftarrow (1 - P) \cdot T_{xy} + \sum_{k} \Delta T_{xy}^{k} \tag{3.2}$$

Keterangan dari rumus tersebut sebgai berikut :

P : persentase evaporasi feromon (dalam desimal)
T_{xy} : nilai feromon untuk jalur xy
ΔT_{xy}^{k} : nilai feromon yang akan ditambahkan oleh semut k pada jalur xy

Pada rumus tersebut, jumlah feromon awal yang terdapat pada suatu jalur akan mengalami pengurangan berdasarkan rasio penguapan feromon P, lalu kemudian akan ditambah dengan nilai feromon masing-masing semut yang melewati jalur tersebut. Nilai feromon masing-masing semut dapat berupa sebuah nilai tetap atau nilai lainnya yang mampu memberikan nilai lebih pada jalur yang lebih cepat (lama proses, jumlah semut yang telah melewati suatu jalur, dan lain-lain). Aturan mengenai kapan dan bagaimana suatu proses penambahan feromon dilakukan dapat bervariasi. Aturan-aturan tersebut biasanya ditentukan berdasarkan jenis permasalahan yang ingin dioptimisasi. Proses penambahan feromon dapat dilakukan ketika terdapat semut yang telah selesai berjalan pada jalurnya atau pada saat seluruh semut telah selesai berjalan pada jalurnya.

Proses perubahan jumlah feromon dapat dilakukan dengan menambahkan sebuah nilai atau penambahkan nilai feromon dengan suatu persentase tertentu. Proses perubahan jumlah feromon dapat dilakukan di seluruh jalur atau hanya di jalur yang dilewati suatu semut. Jika perubahan dilakukan di seluruh jalur, fungsi / aturan khusus untuk perubahan jumlah feromon sebagai berikut.

$$\Delta T_{xy}^{k} = \{\, Q \,, \text{jika semut } \mathbf{k} \text{ menggunakan jalur } \mathbf{xy}. 0, \text{untuk lainnya.} \tag{3.3}$$

Keterangam aturan khusus di atas sebagai berikut :

ΔT_{xy}^{k} : nilai feromon yang akan ditambahkan oleh semut k pada jalur xy
Q : nilai penambahan feromon (dapat berupa suatu angka tetap / hasil perhitungan)

3. Kondisi berhenti
   Selama proses optimisasi, algoritma ant colony akan secara terus menerus melakukan fase pelatihan. Pada setiap fase pelatihan akan dibentuk semut-semut yang akan memilih suatu jalur secara acak berdasarkan nilai feromon. Setelah suatu fase pelatihan selesai, perubahan nilai feromon akan dilakukan berdasarkan jalur-jalur yang dipilih oleh masing-masing semut.

   Proses optimisasi akan diberhentikan jika suatu kondisi telah terpenuhi. Waktu berhentinya suatu proses pelatihan dapat ditentukan dengan berbagai parameter. Proses pelatihan dapat dianggap selesai jika telah melewati beberapa fase pelatihan, jumlah semut yang disebar sudah cukup banyak, atau jika sudah terdapat jalur yang dipilih oleh sebagian besar semut. Proses pelatihan juga dapat diberhentikan jika fase-fase pelatihan terakhir selalu memberikan hasil solusi yang sama / tidak mampu membentuk solusi lain yang lebih optimal.

DAFTAR REFERENSI

[1] de Berg, M., Cheong, O., van Kreveld, M. J., dan Overmars, M. (2008) *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, Berlin.

[2] Baker, K. (1974) *Introduction to sequencing and scheduling.* Wiley.

[3] Bedworth, D. D. dan Bailey, J. E. (1999) *Integrated Production Control Systems: Management, Analysis, Design*, 2nd edition. John Wiley & Sons, Inc., New York, NY, USA.

[4] van Kreveld, M. J. (2004) Geographic information systems. Bagian dari Goodman, J. E. dan O'Rourke, J. (ed.), *Handbook of Discrete and Computational Geometry.* Chapman & Hall/CRC, Boca Raton.

[5] Buchin, K., Buchin, M., van Kreveld, M. J., Löffler, M., Silveira, R. I., Wenk, C., dan Wiratma, L. (2013) Median trajectories. *Algorithmica*, **66**, 595–614.

[6] van Kreveld, M. J. dan Wiratma, L. (2011) Median trajectories using well-visited regions and shortest paths. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, USA, 1-4 November, pp. 241–250. ACM, New York.

[7] Lionov (2002) Animasi algoritma sweepline untuk membangun diagram voronoi. Skripsi. Universitas Katolik Parahyangan, Indonesia.

[8] Wiratma, L. (2010) Following the majority: a new algorithm for computing a median trajectory. Thesis. Utrecht University, The Netherlands.

[9] Wiratma, L. (2022) Coming Not Too Soon, Later, Delay, Someday, Hopefully. Disertasi. Utrecht University, The Netherlands.

[10] van kreveld, M., van Lankveld, T., dan Veltkamp, R. (2013) Watertight scenes from urban lidar and planar surfaces. Technical Report UU-CS-2013-007. Utrecht University, The Netherlands.

[11] Rekhter, Y. dan Li, T. (1994) A border gateway protocol 4 (bgp-4). RFC 1654. RFC Editor, http://www.rfc-editor.org.

[12] ITU-T Z.500 (1997) *Framework on formal methods in conformance testing.* International Telecommunications Union. Geneva, Switzerland.

[13] Version 9.0.0 (2016) *The Unicode Standard.* The Unicode Consortium. Mountain View, USA.

[14] Version 7.0 Nougat (2016) *Android API Reference Manual.* Google dan Open Handset Alliance. Mountain View, USA.

[15] Webb, R., Daruca, O., dan Alfadian, P. (2012) *Method of optimizing a text massage communication between a server and a secure element.* Paten no. EP2479956 (A1). European Patent Organisation. Munich, Germany.

[16] Wiratma, L. (2009) Median trajectory. Report for GMT Experimentation Project at Utrecht University.

[17] Lionov (2011) Polymorphism pada C++. Catatan kuliah AKS341 Pemrograman Sistem di Universitas Katolik Parahyangan, Bandung. http://tinyurl.com/lionov. 30 September 2016.

[18] Erickson, J. (2003) CG models of computation? http://www.computational-geometry.org/mailing-lists/compgeom-announce/2003-December/000852.html. 30 September 2016.

[19] AGUNG (2012) Menjajal tango 12. Majalah HAI no 02, Januari 2012.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

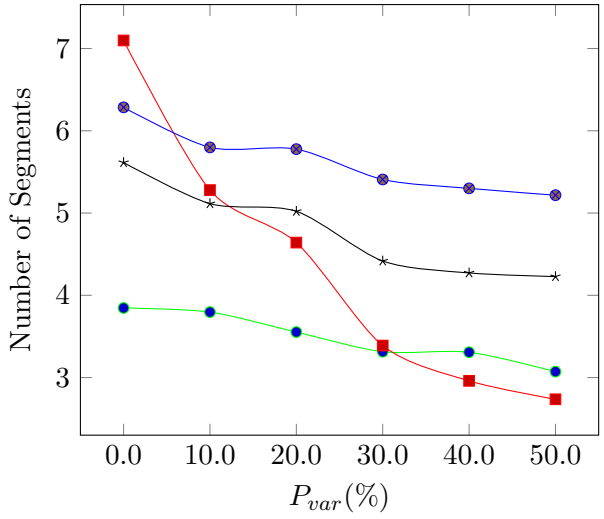
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

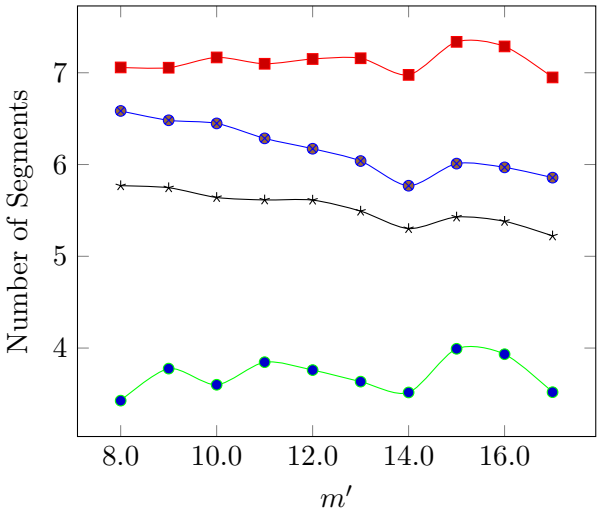
LAMPIRAN B

HASIL EKSPERIMEN

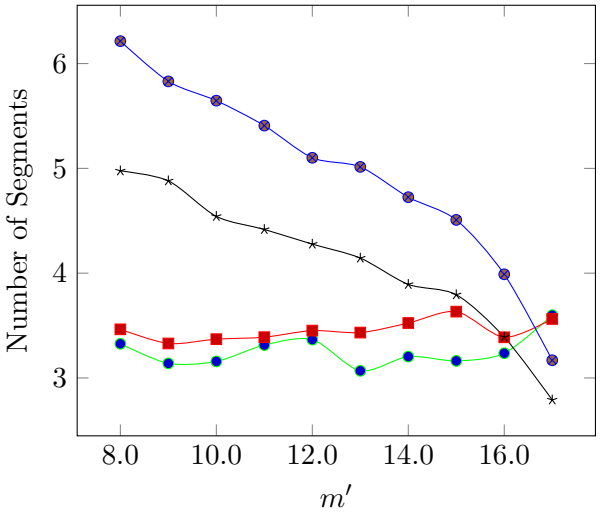
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



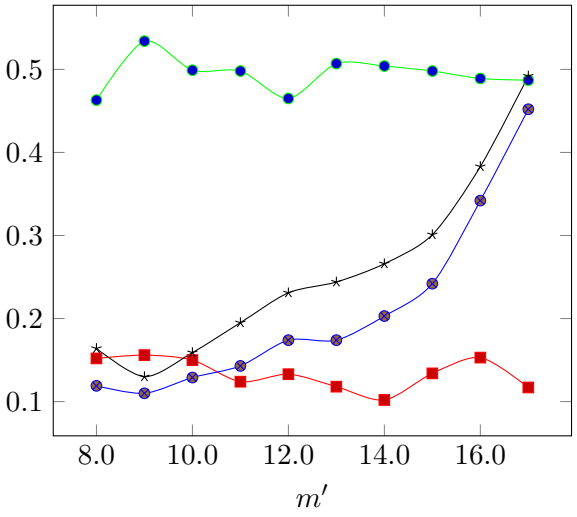
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4