SKRIPSI

PENGGUNAAN ALGORITMA ANT COLONY UNTUK OPTIMISASI PENJADWALAN HYBRID FLOW SHOP

ALEXANDER SURYA

NPM: 2012730015

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2016

UNDERGRADUATE THESIS

APPLICATION OF ANT COLONY ALGORITHM FOR AN OPTIMISATION IN HYBRID FLOW SHOP SCHEDULING

ALEXANDER SURYA

NPM: 2012730015

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2016

LEMBAR PENGESAHAN

PENGGUNAAN ALGORITMA ANT COLONY UNTUK OPTIMISASI PENJADWALAN HYBRID FLOW SHOP

ALEXANDER SURYA

NPM: 2012730015

Bandung, 20 Juli 2016

Menyetujui,

Pembimbing Tunggal

Dr. rer. nat. Cecilia Esti Nugraheni

Ketua Tim Penguji	Anggota Tim Penguji
Luciana Abednego, M.T.	Aditya Bagoes Saputra, M.T.

Mengetahui,

Ketua Program Studi

Mariskha Tri Aditia, PDEng

PERNYATAAN

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

PENGGUNAAN ALGORITMA ANT COLONY UNTUK OPTIMISASI PENJADWALAN HYBRID FLOW SHOP

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 20 Juli 2016

Meterai

Alexander Surya
NPM: 2012730015

ABSTRAK

Skripsi ini dibuat untuk mempelajari, menganalisa, dan mengukur kinerja algoritma *ant colony* dalam proses penjadwalan *hybrid flow shop*. Proses penjadwalan *hybrid flow shop* adalah penentuan urutan pengerjaan sekumpulan pekerjaan yang akan dikerjakan pada serangkaian perangkat mesin, di mana terdapat lebih dari satu mesin yang mampu mengerjakan sebuah proses. Urutan pengerjaan yang berbeda mampu menghasilkan waktu pengerjaan yang berbeda. Ada banyak algoritma yang dapat dipakai untuk menentukan urutan antrian pengerjaan pekerjaan tersebut. Salah satu algoritma yang dapat dipakai pada proses penjadwalan *hybrid flow shop* adalah algoritma *ant colony*. Algoritma *ant colony* adalah algoritma optimisasi yang mengikuti cara kerja dari koloni semut. Algoritma ini menggunakan nilai feromon untuk penentuan solusi yang optimal. Semakin besar nilai feromon suatu solusi, semakin besar pula kemungkinan dipilihnya solusi tersebut. Untuk kasus-kasus yang sederhana, algoritma ini masih mampu mencapai hasil yang optimal, akan tetapi algoritma ini akan semakin sulit mencapai hasil optimal jika kasus yang ingin dicari hasil optimalnya semakin rumit. Algoritma ini juga cenderung membentuk solusi-solusi yang mirip dari sebuah kasus yang sama. Hal ini menunjukkan bahwa algoritma ini sedang berusaha membentuk satu solusi yang sama dan solusi tersebut merupakan solusi yang dianggap optimal. Pada karya ilmiah ini akan dibahas lebih lanjut mengenai bagaimana cara pengaplikasian algoritma *ant colony* pada proses penjadwalan *hybrid flow shop* dan apa saja kelebihan serta kekurangan dari penggunaan algoritma *ant colony* pada proses penjadwalan *hybrid flow shop*.

Kata-kata kunci: Flow Shop, Ant-Colony, Hybrid, Penjadwalan

ABSTRACT

This thesis was made to study, analyze, and measure the performance of ant colony algorithm in hybrid flow shop scheduling. Hybrid flow shop scheduling is a process to determine the order of processing collection of work that will be done in a series of machine tools, where there is more than one machine capable of working on a process. Different order of job queue is capable to producing different total processing time. There are many algorithms that can be used to determine the order of the job queue. One algorithm that can be used in hybrid flow shop scheduling is ant colony algorithm. Ant colony algorithm is an optimization algorithm that follow the work procedures of an ant colony. This algorithm uses pheromone value to determine the optimal solution. The greater the value of the pheromone to one solution, the greater the possibility for solution to be chosen. For simple cases, the algorithm is still capable to achieve optimal results, but this algorithm will be more difficult to achieve optimal results if the case is more complicated. This algorithm is also likely to form similar solutions from one case. This shows that this algorithm is trying to form the same solution and the solution is a solution which is considered optimal. In this paper will be discussed further on how ant colony algorithm application in hybrid flow shop scheduling process and what are the advantages and disadvantages of the use of ant colony algorithm in the scheduling process hybrid flow shop.

Keywords: Flow Shop, Ant-Colony, Hybrid, Scheduling

Dipersembahkan untuk Tuhan Yang Maha Esa, kedua orang tua, dan Ibu Cecilia Esti Nugraheni selaku dosen pembimbing.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas berkat, rahmat, dan hidayah-Nya, penulis dapat menyelesaikan penulisan skripsi ini yang berjudul "Penggunaan Algoritma *Ant Colony* untuk Optimisasi Penjadwalan *Hybrid Flow Shop*". Penulis menyadari bahwa penyelesaian penyusunan skripsi ini juga tidak terlepas dari bantuan berbagai pihak. Penulis ingin berterima kasih kepada :

1. Ibu Dr. rer. nat. Cecilia Esti Nugraheni selaku dosen pembimbing yang telah membimbing penulis dan memberikan dukungan serta bantuan kepada penulis selama proses pengerjaan skripsi.
2. Dosen penguji yang telah menguji dan memberikan masukan untuk kelancaran penulisan skripsi kepada penulis.
3. Keluarga dan kerabat penulis yang selalu memberikan dorongan untuk menyelesaikan tugas skripsi ini.
4. Teman-teman dari Jurusan Teknik Informatika UNPAR, baik angkatan 2012 maupun kakak senior yang telah berjuang bersama dalam pengerjaan topik skripsinya masing-masing dan memberikan semangat serta membantu penulis dalam pengerjaan skripsi.
5. Kepada mereka yang tidak bisa disebutkan satu-persatu yang telah membantu penulis selama penulisan skripsi.

Penulis berharap bahwa penulisan skripsi ini dapat membantu bagi para pembaca yang sedang meneliti atau mempelajari topik penjadwalan *hybrid flow shop*. Akhir kata, penulis bersedia menerima kritik maupun saran yang dapat membangun masing-masing pihak penulis maupun pembaca agar dapat berkarya dengan lebih baik lagi. Selain itu, penulis meminta maaf jika terdapat kekurangan dalam karya tulis ini. Semoga karya tulis ini dapat bermanfaat bagi semua pihak. Terima kasih.

Bandung, Juli 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Proses penjadwalan *flow shop* adalah proses penentuan urutan pengerjaan sekumpulan pekerjaan yang akan dikerjakan pada sebuah rangkaian perangkat mesin pada rentang waktu tertentu. Masing-masing dari pekerjaan tersebut terdiri dari beberapa proses-proses kecil. Proses tersebut harus dikerjakan oleh seluruh mesin dari rangkaian perangkat mesin tersebut sesuai dengan urutan yang ditentukan. Tujuan utama dari proses penjadwalan *flow shop* adalah menentukan urutan pengerjaan yang mampu menghasilkan *makespan* yang paling optimal. *Makespan* itu sendiri adalah waktu yang dibutuhkan untuk menyelesaikan sekumpulan pekerjaan tersebut.[1]

Urutan pengerjaan yang berbeda mampu menghasilkan nilai *makespan* yang berbeda pula. Proses penjadwalan ini diperlukan agar lama pengerjaan pekerjaan-pekerjaan tersebut dapat menjadi lebih cepat, efisien, dan efektif. Pada pengerjaan suatu proses di sebuah perusahaan, ada kalanya sudah banyak pekerjaan-pekerjaan lain yang menunggu untuk dikerjakan oleh perangkat mesin. Penentuan prioritas mengenai pekerjaan mana yang harus dikerjakan terlebih dahulu akan sangat berperan dalam menunjang efisiensi dan efektifitas dari suatu perusahaan.

Penentuan prioritas tersebut akan lebih rumit jika diaplikasikan pada proses penjadwalan *hybrid flow shop*. Sama seperti proses penjadwalan *flow shop*, proses penjadwalan *hybrid flow shop* juga bertugas untuk menentukan urutan antrian pengerjaan yang paling efisien. Hal yang membedakan *hybrid flow shop* dengan *flow shop* adalah jumlah perangkat mesin yang akan mengerjakan sekumpulan pekerjaan tersebut. *Hybrid flow shop* memiliki lebih dari satu perangkat mesin yang mampu mengerjakan suatu proses, sehingga mampu mengerjakan lebih dari satu proses pekerjaan pada satu waktu.[2]

Ada banyak algoritma perhitungan yang dapat dipakai dalam menentukan urutan pengerjaan dari suatu kasus *hybrid flow shop*, di antaranya :

- *Simulated Annealing* [3]
- Algoritma genetik [4]
- Algoritma *ant colony* [5]
- *Particle Swarm Inteligence* [6]

Setiap algoritma tersebut memiliki keunggulan dan kekurangannya masing-masing. Sebagai contoh, algoritma *Simmulated Annealing* akan selalu mampu menemukan solusi yang lebih baik dari setiap prosesnya. Algoritma genetik mampu membentuk solusi acak secara cepat, sehingga proses perhitungan tingkat keoptimalan solusi tersebut dapat dilakukan dengan lebih cepat. Algoritma *ant colony* mampu dikerjakan secara paralel, sehingga dapat diatur agar mampu dioptimisasi dalam waktu yang relatif cepat.

Setiap kasus proses penjadwalan *hybrid flow shop* juga memiliki karakteristiknya masing-masing, sehingga proses optimisasinya perlu menggunakan algoritma yang cocok dengan karakteristiknya. Dalam proses penjadwalan *hybrid flow shop*, penentuan algoritma mana yang akan dipakai sangat penting, karena akan mempengaruhi ukuran prioritas. Algoritma yang digunakan juga mampu mempengaruhi penentuan urutan pengerjaan yang akan dihasilkan oleh proses optimisasi.

Salah satu algoritma yang dapat dipakai dalam proses penjadwalan *hybrid flow shop* ini adalah algoritma *ant colony*. Algoritma *ant colony* merupakan algoritma optimisasi untuk pengambilan keputusan/solusi yang optimal. Cara kerja dari algoritma ini mengikuti cara kerja dari koloni semut dalam mencari dan menginformasikan sumber makanan. Semut yang pada awalnya berpencar mencari makanan ke segala arah, pada akhirnya dapat menentukan jalur-jalur terdekat ke sumber makanan.

Selama algoritma ini menyebarkan "semut"-nya, suatu zat/nilai feromon akan disebar pada jalur-jalur yang dilalui. Nilai feromon ini akan diatur berdasarkan tingkat keoptimalan suatu jalur. Semakin optimal suatu jalur, semakin banyak nilai feromon yang diberikan. Nilai feromon ini kemudian akan digunakan oleh semut-semut selanjutnya untuk menentukan kemungkinan dipilihnya suatu jalur. Semakin besar nilai feromon dari suatu jalur, semakin besar pula kemungkinan jalur tersebut untuk dipilih.[7]

Berikut ini akan dijelaskan secara lebih lanjut mengenai algoritma *ant colony* dan pengaplikasiannya pada proses penjadwalan *hybrid flow shop*. Kelebihan dan kekurangan dari pengaplikasian algoritma *ant colony* juga akan dibahas. Tidak lupa, karakteristik kerja algoritma *ant colony* seperti kemampuannya dalam menangani kasus *hybrid flow shop* yang rumit, kemampuannya dalam menangani kasus *hybrid flow shop* dengan waktu pengerjaan yang unik, dan kemampuannya menentukan urutan antrian yang optimal juga akan dibahas.

1.2 Perumusan Masalah

Perumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Apa itu proses penjadwalan *hybrid flow shop*?
2. Bagaimana cara kerja algoritma *ant colony*?
3. Bagaimana cara pengaplikasian algoritma *ant colony* pada proses penjadwalan *hybrid flow shop*?
4. Bagaimana pengaruh algoritma *ant colony* pada optimisasi proses penjadwalan *hybrid flow shop*?

1.3 Tujuan Penulisan

Tujuan dari penelitian ini adalah:

1. Mempelajari sistematika dan cara kerja dari *hybrid flow shop*.
2. Mempelajari dasar-dasar teori dan cara kerja dari algoritma *ant colony*.
3. Mengimplementasikan algoritma *ant colony* pada proses penjadwalan *hybrid flow shop*.
4. Mengukur performa algoritma *ant colony* yang diaplikasikan pada mesin penjadwalan *hybrid flow shop*.

1.4 Batasan dan Asumsi

Batasan dan asumsi untuk penelitian ini adalah:

1. Waktu proses dari setiap pekerjaan 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. Urutan penyelesaian pekerjaan dapat berbeda dengan urutan masuk pekerjaan.
5. Eksperimen dilakukan dengan sampel data kasus milik soa.iti.es.
6. Eksperimen dilakukan menggunakan permasalahan kasus *hybrid flow shop* dengan jumlah mesin yang sama pada setiap tahap proses.
7. Pengukuran tingkat performansi dari algoritma hanya menggunakan perbandingan nilai *makespan*.

1.5 Metodologi Penyelesaian Masalah

Metodologi penyelesaian masalah dalam penelitian ini adalah :

1. Studi Literatur
   Mencari referensi dari sumber-sumber tertentu untuk memperdalam pemahaman mengenai cara kerja proses penjadwalan *hybrid flow shop* dan cara kerja algoritma *ant colony* agar kemudian dapat mengaplikasikan algoritma tersebut pada proses penjadwalan *hybrid flow shop*.
2. Analisa Kasus
   Menentukan cara pengaplikasian algoritma *ant colony* untuk optimisasi penjadwalan *hybrid 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 *hybrid 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 *hybrid 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 Penulisan

Sistematika penulisan karya tulis ini adalah sebagai berikut :

1. Bab 1 : Pendahuluan untuk mendefinisikan masalah yang akan dibahas, alasan pemilihan topik dan usulan solusi.

2. Bab 2 : Dasar teori yang digunakan dalam penelitian ini. Pembahasan permasalahan *hybrid flow shop*. Pembahasan cara kerja dari algoritma *ant colony*. Pembahasan cara pengaplikasian algoritma *ant colony* untuk optimisasi proses penjadwalan *hybrid 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 *hybrid 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

DASAR TEORI

Pada bab ini akan dibahas mengenai detil permasalahan *hybrid flow shop* yang ingin diselesaikan. Cara kerja dan sistematika algoritma *ant colony* yang akan digunakan untuk optimisasi permasalahan tersebut juga akan dibahas.

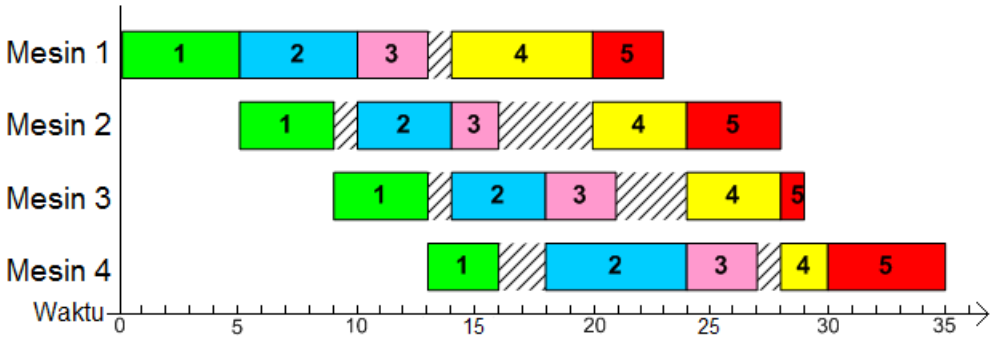
2.1 *Hybrid Flow Shop*

Proses penjadwalan *flow shop* adalah penentuan urutan pengerjaan sekumpulan pekerjaan yang akan dikerjakan pada serangkaian perangkat mesin. Masing-masing pekerjaan tersebut terdiri dari beberapa proses-proses kecil. Masing-masing dari proses tersebut harus dikerjakan oleh bagian tertentu dari rangkaian mesin sesuai dengan urutan yang ditentukan.[1]

Tujuan utama dari proses penjadwalan *flow shop* adalah menentukan urutan pengerjaan yang menghasilkan nilai *makespan* yang minimum. *Makespan* adalah waktu yang dibutuhkan untuk menyelesaikan sekumpulan pekerjaan dengan suatu urutan tertentu. Urutan pengerjaan yang berbeda mampu menghasilkan nilai *makespan* yang berbeda pula

Elemen-elemen yang terlibat pada proses penjadwalan *flow shop* adalah:

- Banyaknya pekerjaan yang ada.
- Banyaknya mesin / proses yang ada.
- Waktu pengerjaan tiap proses pada masing-masing pekerjaan.



Gambar 2.1: Contoh solusi *flow shop*

Gambar 2.1 merupakan salah satu contoh proses penjadwalan *flow shop*. Proses penjadwalan tersebut melibatkan 4 buah mesin / proses dan 5 buah pekerjaan. Urutan pengambilan pekerjaan yang digunakan adalah 1, 2, 3, 4, 5. Masing-masing proses dari pekerjaan-pekerjaan tersebut

dikerjakan oleh perangkat mesin yang berbeda. Diagram batang pada gambar 2.1 menunjukkan masing-masing proses dari tiap pekerjaan yang ada. Diagram batang yang terdapat pada sebuah baris menunjukkan bahwa proses dari suatu pekerjaan sedang dikerjakan pada mesin yang direpresentasikan oleh baris tersebut.

Proses *flow shop* akan mencari mesin yang sedang menunggu pekerjaan dan langsung memberikan pekerjaan yang mungkin diberikan kepada mesin tersebut. Saat proses 1 dari pekerjaan 1 sudah selesai, mesin 2 akan langsung mengerjakan proses 2 dari pekerjaan 1. Mesin 1 akan langsung mengerjakan pekerjaan selanjutnya, yaitu proses 1 dari pekerjaan 2. Saat proses 2 dari pekerjaan 1 sudah selesai, mesin 3 akan langsung mengerjakan proses 3 dari pekerjaan 1. Mesin 2 yang seharusnya mengerjakan proses 2 dari pekerjaan 2 akan mengalami *idle*, karena proses 1 dari pekerjaan 2 belum selesai dikerjakan di mesin 1.

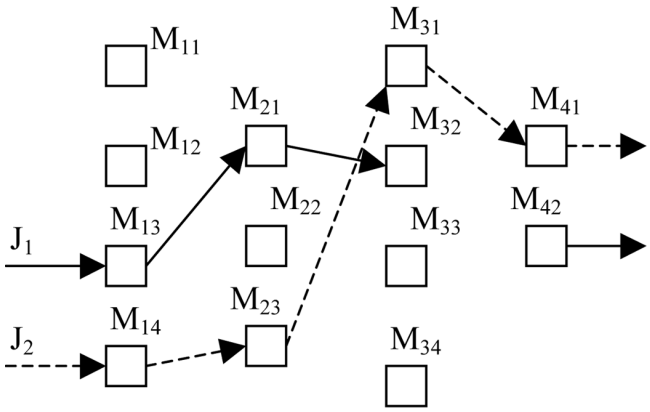
Idle itu sendiri adalah kondisi dimana mesin belum dapat melakukan pekerjaan apapun, karena adanya proses lain yang belum selesai. *Idle* akan memberikan pengaruh yang sangat besar terhadap *makespan* yang akan didapat. Semakin banyak *idle*, maka nilai *makespan* akan semakin besar.

Idle juga dapat terjadi jika terdapat mesin yang sudah selesai mengerjakan proses 1, akan tetapi mesin 2 yang akan mengerjakan proses 2 sedang dipakai. Hal ini terjadi karena, mesin 1 masih harus tetap menyimpan proses 1 tersebut sebelum diberikan ke mesin 2. Jika mesin 2 sudah mampu menerima proses 1 tersebut, maka mesin 1 akan langsung memberikan pekerjaan tersebut ke mesin 2 untuk dilanjutkan ke proses 2. Setelah proses yang disimpan diberikan ke mesin selanjutnya, mesin 1 dapat kembali menerima pekerjaan baru.

Sama seperti proses penjadwalan *flow shop*, *hybrid flow shop* juga bertugas untuk menentukan urutan antrian pengerjaan yang paling efisien. Hal yang membedakan *hybrid flow shop* dengan *flow shop* adalah jumlah perangkat mesin yang mampu mengerjakan setiap proses. Pada *hybrid flow shop* banyaknya mesin yang mampu mengerjakan setiap proses bervariasi dan dapat berjumlah lebih dari satu. *Hybrid flow shop* memungkinkan proses yang sama dari pekerjaan yang berbeda dikerjakan secara serentak.

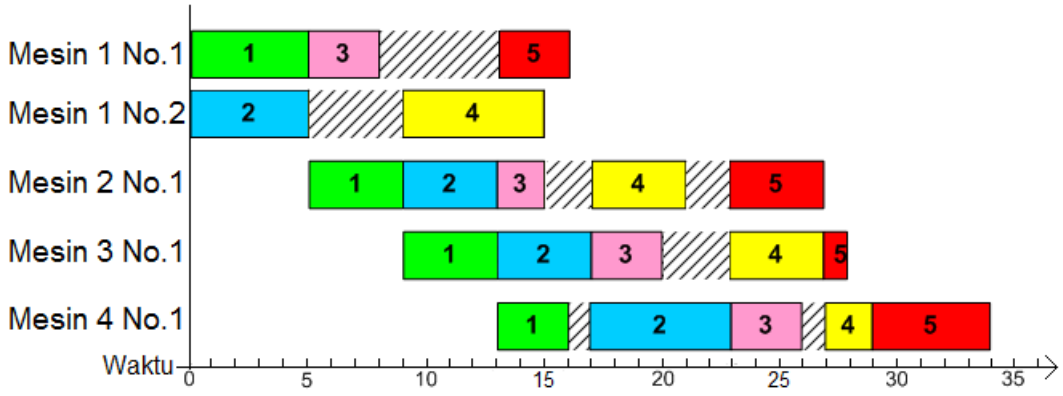
Secara singkat, karakteristik dari *hybrid flow shop* adalah [2]:

- Banyaknya mesin yang mampu mengerjakan setiap proses minimal 1 dan terdapat sebuah proses yang memiliki lebih dari 1 mesin yang mampu mengerjakannya.
- Semua penyelesaian pekerjaan mengikuti alur proses yang sama dan sistematis.
- Setiap proses pasti akan dikerjakan dan hanya akan dikerjakan sebanyak satu kali.



Gambar 2.2: Ilustrasi *hybrid flow shop*

Pada Gambar 2.2, diilustrasikan terdapat 2 buah pekerjaan yang harus diselesaikan. Pekerjaan tersebut harus melewati 4 buah rangkaian proses. Gambar 2.2 juga mengilustrasikan bahwa terdapat lebih dari 1 mesin pada masing-masing proses. Proses 1 memiliki 4 buah mesin, proses 2 memiliki 2 mesin, proses 3 memiliki 4 mesin, dan proses 4 memiliki 2 mesin. Jumlah mesin yang lebih dari 1 memungkinkan pengerjaan masing-masing proses dari pekerjaan 1 dan pekerjaan 2 secara bersamaan.



Gambar 2.3: Contoh solusi *hybrid flow shop*

Jika permasalahan *flow shop* pada gambar 2.1 dikerjakan pada penjadwalan *hybrid flow shop*, maka hasil yang diberikan dapat diilustrasikan sebagai gambar 2.3 di atas. Dimisalkan proses penjadwalan *hybrid flow shop* tersebut melibatkan 2 buah perangkat mesin yang mampu mengerjakan proses 1. *Hybrid flow shop* mampu menghasilkan *makespan* yang lebih cepat dibandingkan dengan *flow shop*. Jumlah perangkat mesin yang terlibat pada *hybrid flow shop* mampu mempengaruhi *makespan* yang dihasilkan.

Sama seperti proses penjadwalan *flow shop*, *hybrid flow shop* akan mencari mesin yang sedang menunggu pekerjaan dan langsung memberikan pekerjaan tersebut. Jika terdapat lebih dari 1 pekerjaan yang dapat dioper dari 1 proses, maka pengoperan akan disesuaikan dengan urutan pengerjaan awal. Pada proses penjadwalan *flow shop*, pekerjaan 2 baru dapat dikerjakan setelah proses 1 dari pekerjaan 1 dilanjutkan ke proses 2 dari pekerjaan 1. Pada proses penjadwalan *hybrid flow shop*, pekerjaan 2 dapat langsung dikerjakan oleh mesin 2 dari proses 1 tanpa harus menunggu pekerjaan 1 yang dikerjakan di mesin 1 dari proses 1.

2.2 Algoritma *Ant Colony*

Algoritma *ant colony* merupakan algoritma optimisasi yang bekerja dengan metode *swarm intelligence*. *Swarm intelligence* merupakan sebuah tata cara untuk menyelesaikan sebuah permasalahan dengan meniru cara interaksi antar aktor pada sebuah sistem. Algoritma *ant colony* adalah algoritma pencarian untuk mendapatkan solusi yang optimal dengan mengikuti prinsip cara kerja dari koloni semut dalam mencari sumber makanan. Dalam mencari sumber makanan, sebuah koloni semut mampu mengkomunikasikan sumber makanan yang ditemukan dengan anggota koloni semut lainnya dan membentuk jalur terpendek untuk mencapai sumber makanan tersebut.

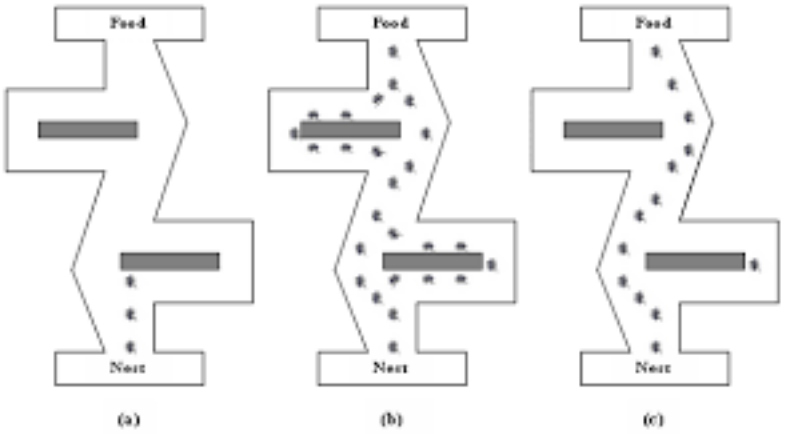
Algoritma *ant colony* merupakan algoritma optimisasi yang bersifat *metaheuristic*. Sifat *metaheuristic* tersebut dapat diartikan bahwa algoritma optimisasi ini akan mengambil beberapa pilihan solusi dari solusi-solusi yang ada dan mencari solusi yang paling optimal di antara pilihan-pilihan solusi tersebut. Algoritma *ant colony* juga memiliki sifat *stochastic*, karena algoritma ini menggunakan perhitungan probabilitas dan angka yang dibangkitkan secara acak dalam menentukan solusi yang optimal. Sifat *stochastic* tersebut mengakibatkan algoritma *ant colony* ini juga bersifat non-deterministik, yang berarti bahwa solusi optimal yang dipilih oleh algoritma ini tidak selalu sama dan belum tentu merupakan solusi yang paling optimal.

Berdasarkan 2 karangan ilmiah milik Dorigo [7] [8], cara kerja dari algoritma *ant colony* dapat dijabarkan sebagai berikut.

2.2.1 Analogi

Dalam mencari makanan, pada awalnya koloni semut akan berpencar ke segala arah. Selama berjalan dan mencari makanan, semut akan menyebarkan semacam petunjuk (feromon) yang dapat dijadikan acuan arah bagi semut-semut lain. Jika makanan telah ditemukan, semut tersebut akan berjalan kembali mengikuti jejak feromon yang telah dia tinggalkan sebelumnya.

Semut-semut selanjutnya akan mengambil jalur secara acak berdasarkan feromon-feromon yang disebar. Semakin kuat dan banyak feromon yang terdapat pada suatu jalur, maka semakin besar kemungkinan bagi suatu semut untuk melewati jalur tersebut. Saat semut melewati jalur tersebut, semut tersebut juga akan ikut menebarkan feromon selama perjalanannya dan memperkuat feromon yang terdapat pada jalur tersebut.



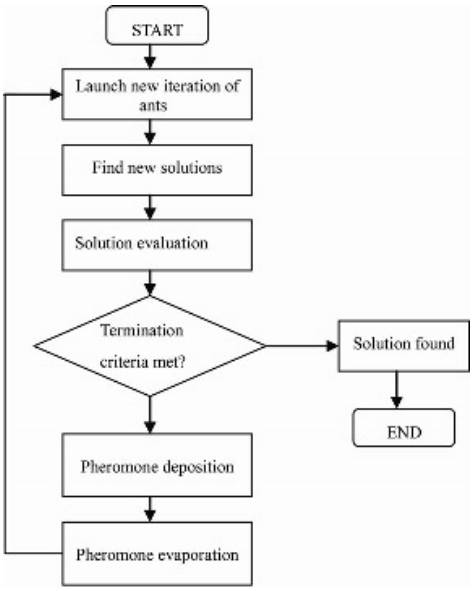
Gambar 2.4: Ilustrasi cara kerja koloni semut

Berdasarkan ilustrasi pada gambar 2.4, jalur yang dihasilkan untuk menuju suatu makanan tidak hanya satu. Selama perjalanan semut mencari makanan akan tercipta banyak jalur yang berbeda-beda dan mungkin akan tertuju pada sumber makanan yang sama seperti pada gambar 2.4 bagian a. Akan tetapi, semut tetap dapat mampu mengambil jalur terpendek untuk menuju sumber makanan. Hal ini dapat dilakukan karena semut akan menebarkan feromon selama perjalanannya seperti pada gambar 2.4 bagian b. Feromon pada suatu jalur akan menjadi semakin kuat jika dilewati oleh semut lain.

Selama proses semut mencari dan mengangkut makanan ke dalam sarangnya, jalur terpendek akan dicari dan ditentukan. Semut yang telah mengambil jalur terpendek cenderung akan lebih cepat kembali ke dalam sarang. Hal ini akan mengakibatkan jumlah feromon pada jalur terpendek tersebut akan lebih cepat bertambah dibandingkan dengan jalur yang lain. Jalur terpendek tersebut juga akan lebih sering dipilih oleh semut-semut lain yang juga akan terus menambah jumlah feromon pada jalur tersebut. Hal tersebut diilustrasikan pada gambar 2.4 bagian c. Jika makanan telah habis, maka jalur tersebut tidak lagi akan dilewati dan feromon pada jalur tersebut perlahan-lahan akan menghilang.

2.2.2 Cara Kerja Algoritma

Gambar 2.5 berikut adalah *flow chart* dari algoritma *ant colony*.



Gambar 2.5: *Flow chart* algoritma *ant colony*[1]

Berdasarkan Gambar 2.5, untuk menjalankan proses pelatihan dengan algoritma *ant colony* perepresentasikan jalur-jalur yang ada perlu dilakukan. 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
```

Berdasarkan *flow chart* pada gambar 2.5, dibentuklah *pseudocode* di atas[7]. Baris pertama dari pseudocode tersebut merupakan proses pembentukan data feromon. Proses pembentukan data feromon tersebut akan memperhatikan kasus yang ingin dicari hasil optimalnya. Nilai masing-masing 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.

2.2.3 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 [7]:

$$P_{xy}^{k} = \frac{T_{xy}^{\alpha}.n_{xy}^{\beta}}{\sum_{x \in available_z}(T_{xz}^{\alpha}.n_{xz}^{\beta})} \tag{2.1}$$

Keterangan

P_{xy}^{k} : 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.

2.2.4 *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 [8]:

$$T_{xy} \leftarrow (1 - P).T_{xy} + \sum_{k} \Delta T_{xy}^{k} \tag{2.2}$$

Keterangan

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 dapat digunakan[8].

$$\Delta T_{xy}^{k} = \begin{cases} Q & \text{jika semut } k \text{ menggunakan jalur } xy \\ 0 & \text{lainnya} \end{cases} \tag{2.3}$$

Keterangan

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)

2.2.5 Kondisi berhenti dari proses optimisasi

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.

2.3 Penerapan Algoritma *Ant Colony* pada Proses Penjadwalan *Hybrid Flow Shop*

Berdasarkan karangan ilmiah Rajendran yang berjudul *"Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs"* [5], algoritma *ant colony* mampu diaplikasikan pada proses penjadwalan *hybrid flow shop*. Salah satu cara pengaplikasian algoritma *ant colony* adalah dengan menjabarkan pekerjaan yang akan diambil setelah diambilnya suatu pekerjaan lain[9]. Algoritma *ant colony* akan menentukan pekerjaan manakah yang sebaiknya diambil setelah diambilnya suatu pekerjaan yang lain.

Masing-masing kemungkinan dipilihnya suatu pekerjaan ditentukan berdasarkan nilai feromon yang ada. Nilai feromon tersebut akan disimpan dalam matriks 2 dimensi. Matriks tersebut berukuran n $\times$ n, di mana n merupakan banyaknya pekerjaan yang ada. Indeks pertama dari matriks menunjukkan posisi saat ini. Posisi saat ini ditentukan oleh pekerjaan yang baru saja diselesaikan / diambil oleh semut. Indeks kedua dari matriks menunjukkan pekerjaan lainnya yang dapat diambil oleh semut.

Masing-masing dari indeks tersebut akan berisi nilai feromon yang merepresentasikan kemungkinan diambilnya suatu pekerjaan. Sebagai contoh, indeks matriks [2][1] merepresentasikan kemungkinan diambilnya pekerjaan 1 setelah diambilnya pekerjaan 2. Indeks matriks [i][i] akan selalu bernilai 0, karena pekerjaan i tidak akan diambil setelah selesainya / diambilnya pekerjaan i yang sama. Matriks lainnya akan memiliki nilai minimal 1 dan tidak mungkin kurang dari 1. Hal ini dilakukan agar suatu solusi / urutan pengambilan tertentu masih memiliki kemungkinan terpilih.

Pemilihan urutan pengerjaan untuk masing-masing semut akan dilakukan secara bertahap, dimulai dari pengambilan pekerjaan pertama hingga pengambilan pekerjaan terakhir. Pengambilan pekerjaan tersebut akan memperhatikan pekerjaan-pekerjaan yang telah diambil. Pengambilan pekerjaan tersebut akan mengacuhkan nilai feromon dari pekerjaan yang telah diambil. Hal ini dilakukan untuk memastikan suatu pekerjaan hanya dikerjakan satu kali. Hal ini juga dilakukan untuk memperhitungkan pengaruh urutan pengambilan pekerjaan terhadap *makespan* yang mungkin dihasilkan.

Tabel 2.1: Contoh tabel nilai feromon

0	7	1
1	0	7
1	7	0

Tabel 2.1 merupakan contoh tabel matriks nilai feromon. Tabel matriks feromon tersebut akan membantu proses pemilihan urutan pengerjaan dari kasus *hybrid flow shop* dengan 3 buah pekerjaan. Nilai feromon-feromon pada matriks tersebut mengarahkan untuk mengambil pekerjaan 2 setelah pekerjaan 1, pekerjaan 3 setelah pekerjaan 2, dan pekerjaan 2 setelah pekerjaan 3. Berdasarkan nilai feromon pada matriks tersebut, urutan pekerjaan yang paling mungkin terbentuk adalah:

- urutan pekerjaan (1,2,3)
- urutan pekerjaan (2,3,1)
- urutan pekerjaan (3,2,1)

Saat semut-semut dibuat, masing-masing semut akan langsung membentuk satu solusi (urutan pengerjaan). Solusi tersebut dipilih secara acak berdasarkan nilai feromon yang ada. Semut-semut tersebut kemudian akan disebar untuk melakukan proses pembelajaran. Proses pembelajaran tersebut dilakukan untuk mencari urutan pengerjaan / solusi yang paling optimal. Jumlah semut yang akan disebar pada proses pelatihan dapat diatur berdasarkan banyaknya kemungkinan solusi / jumlah pekerjaan yang ada.

Semut yang telah menjalankan proses *hybrid flow shop* berdasarkan solusinya masing-masing akan menghasilkan suatu nilai feromon untuk solusi yang dipilihnya. Pada saat melakukan *update* feromon, feromon-feromon yang ditambahkan disesuaikan dengan urutan pengerjaan yang diambil oleh semut. Jika semut mengambil urutan pengerjaan (3,2,1) maka nilai feromon pada indeks matriks [3][2], dan indeks matriks [2][1] akan ditambahkan.

Nilai feromon yang ditambahkan akan disesuaikan dengan hasil *makespan* yang dihasilkan oleh urutan pengerjaan tersebut. Semakin kecil *makespan* yang dihasilkan, semakin besar nilai feromon yang ditambahkan. Hal ini dapat dilakukan dengan menambahkan nilai feromon dengan selisih *makespan* dari suatu konstanta tetap. Konstanta tersebut dapat berupa *makespan* maksimal yang mungkin dihasilkan pada kasus *hybrid flow shop* yang digunakan.

Semut akan terus disebar untuk melakukan *update* feromon hingga proses optimisasi selesai. Feromon tersebut akan mempengaruhi prediksi solusi / urutan pengerjaan yang menghasilkan nilai *makespan* optimal. Suatu proses optimisasi dianggap selesai setelah suatu kondisi berhenti tercapai.

Proses pembentukan dan penyebaran semut pada algoritma *ant colony* dapat dikerjakan secara paralel. Proses pembentukan dan penyebaran semut tersebut dapat dilakukan dengan memanfaatkan kelas "Thread" pada bahasa pemrograman Java. Dengan memanfaatkan kelas "Thread" tersebut, jumlah semut yang dibentuk pada satu waktu dapat relatif lebih banyak.

2.4 Data Pengujian

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.

Soa.iti.es adalah *web hosting* yang menyediakan kumpulan data masukan dan keluaran untuk berbagai tipe proses penjadwalan *flow shop*. Dalam alamat web "soa.iti.es/problem-instances", web hosting ini menyediakan berbagai tipe data kasus yang dapat digunakan untuk pengujian tingkat keoptimalan suatu algoritma dalam suatu proses penjadwalan. Berikut ini adalah beberapa tipe data kasus penjadwalan yang disediakan pada soa.iti.es :

- *Instances for distributed assembly permutation flowshops with sequence dependent setup times*
- *Instances for cyclic and noncyclic single machine scheduling problem with highly perishable products and deadlines*
- *Instances for cyclic scheduling of perishable products in parallel machines with release dates, due dates and deadlines*
- *Instances for mixed no-idle flowshop problems*
- *Instances for distributed assembly permutation flowshops*
- *Instances for hybrid flowshops with identical machines per stage and due windows*

Data kasus "*Instances for hybrid flowshops with identical machines per stage and due windows*" dapat digunakan untuk pengujian kasus *hybrid flow shop*. Data kasus ini merupakan data kasus penjadwalan *flow shop* dengan beberapa perangkat mesin yang dapat digunakan pada setiap tahap proses. Pada data kasus ini perangkat mesin yang dapat digunakan pada setiap tahap proses jumlahnya sama. Berikut ini adalah contoh data masukan dari tipe kasus ini.

```
1  HFSDDW
2  10  4   2
3  0   72  1   81
4  0   77  1   56
5  0   21  1   56
6  0   90  1   44
7  0   97  1   28
8  0   29  1   47
9  0   63  1   47
10 0   98  1   4
11 0   86  1   99
12 0   68  1   32
13 LBCmax: 367
```

Gambar 2.6: Contoh data masukan kasus *hybrid flow shop* milik soa.iti.es

Berdasarkan gambar 2.6, data-data yang diberikan dari data kasus ini adalah :

- Banyaknya pekerjaan
- Banyaknya mesin
- Banyaknya proses
- Detil waktu pengerjaan proses dari masing-masing pekerjaan
- Nilai *makespan* yang paling optimal

Baris kedua dari data kasus, secara terurut menjabarkan banyaknya pekerjaan, banyaknya mesin, dan banyaknya proses. Perlu diketahui, bahwa banyaknya mesin pada data kasus menjabarkan jumlah mesin dari seluruh proses yang ada. Oleh karena itu jika banyak mesin adalah 4 dan banyak proses adalah 2, maka dapat disimpulkan bahwa banyak mesin yang ada pada setiap tahap proses berjumlah 2.

Baris-baris selanjutnya akan menjabarkan detil waktu proses dari masing-masing pekerjaan. Sebuah baris akan menjabarkan waktu-waktu proses dari sebuah pekerjaan. Penjabaran waktu proses tersebut didahului dengan nomor proses terlebih dahulu. Banyaknya pekerjaan, proses, mesin, dan detil waktu masing-masing proses merupakan data masukan untuk suatu kasus penjadwalan *hybrid flow shop*.

Baris terakhir dari data menunjukkan nilai *makespan* paling optimal yang mungkin dihasilkan. Nilai *makespan* optimal yang dicantumkan pada LBCmax tersebut berguna untuk pengujian tingkat keoptimalan proses optimisasi oleh suatu algoritma. Jika nilai *makespan* yang dihasilkan dengan optimisasi oleh suatu algoritma semakin mendekati nilai LBCmax tersebut, maka algoritma tersebut dapat dianggap baik untuk optimisasi kasus *hybrid flow shop* tersebut. Data kasus ini juga menyediakan rentang waktu pengerjaan dari masing-masing pekerjaan untuk mempermudah proses pencarian solusi optimal.

Data kasus yang disediakan oleh soa.iti.es ini dibagi menjadi 2 jenis, kasus yang besar dan kasus yang kecil. Jenis tersebut ditentukan berdasarkan banyaknya pekerjaan / tingkat kerumitan dari kasus tersebut. Jika data kasus melibatkan pekerjaan yang banyaknya kurang dari 50, maka kasus tersebut termasuk ke dalam kasus yang kecil. Jika data kasus melibatkan 50 pekerjaan atau lebih, maka kasus tersebut termasuk ke dalam kasus yang besar.

Data kasus yang disediakan bervariasi berdasarkan banyaknya pekerjaan mesin dan proses pada kasus tersebut. Beberapa contoh data kasus yang disediakan oleh soa.iti.es di antaranya :

- 10 pekerjaan, 2 proses, dan 2 mesin untuk setiap tahap proses
- 10 pekerjaan, 4 proses, dan 3 mesin
- 20 pekerjaan, 2 proses, dan 2 mesin
- 50 pekerjaan, 5 proses, dan 3 mesin
- 50 pekerjaan, 10 proses, dan 5 mesin
- 100 pekerjaan, 5 proses, dan 5 mesin

BAB 3

ANALISIS MASALAH

Pada bab ini akan dibahas mengenai hasil analisa permasalahan *hybrid flow shop* dan cara pengaplikasian algoritma *ant colony* untuk optimisasi solusi permasalahan tersebut. Pada bab ini juga akan dibahas mengenai rancangan awal dari perangkat lunak yang memungkinkan diaplikasikannya algoritma *ant colony* pada permasalahan *hybrid flow shop*.

3.1 Analisis Kasus

Proses penjadwalan *hybrid flow shop* adalah proses pengaturan urutan pengerjaan sekumpulan pekerjaan dengan jumlah proses tertentu pada beberapa rangkaian mesin. 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 *hybrid flow shop* adalah:

- Banyaknya pekerjaan
- Banyaknya mesin / proses
- Detil waktu pengerjaan masing-masing proses untuk setiap pekerjaan
- Banyaknya mesin yang terdapat pada masing-masing tahap proses

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 *hybrid flow shop* adalah algoritma *ant colony*. Tingkat keoptimalan setiap algoritma berbeda-beda, oleh karena itu perlu dibuat sebuah perangkat lunak untuk optimisasi penjadwalan *hybrid flow shop* dengan menggunakan algoritma *ant colony*. Algoritma *ant colony* akan menerima data masukan dari suatu kasus *hybrid flow shop*, kemudian akan mencari urutan pengerjaan / solusi yang paling optimal dari kasus tersebut.

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 Konfigurasi Algoritma *Ant Colony* pada Penjadwalan *Hybrid Flow Shop*

3.2.1 Representasi Urutan Pengerjaan Sebagai Jalur

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 Panduan Pemilihan Urutan Pengerjaan / Jalur

Makespan yang dihasilkan oleh masing-masing urutan pengerjaan biasanya akan bernilai berbeda. Penentuan urutan pengerjaan yang optimal akan ditentukan oleh nilai feromon yang akan diberikan oleh semut-semut yang telah disebar. Nilai feromon tersebut dapat disimpan dengan cara yang beragam.

Dalam kasus *hybrid flow shop* juga terdapat beragam cara dalam menyimpan feromon dan penentuan urutan yang lebih optimal. Salah satunya adalah dengan menyimpan nilai feromon sebagai panduan pemilihan pekerjaan selanjutnya. Nilai feromon akan menyimpan kecenderungan dipilihnya suatu pekerjaan setelah dipilihnya suatu pekerjaan yang lain. Nilai feromon tersebut akan disimpan dalam bentuk matriks 2 dimensi.

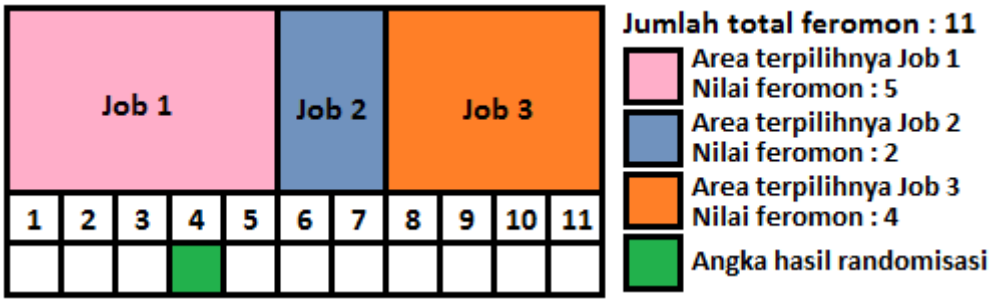
Indeks pertama dari matriks akan merepresentasikan nomor pekerjaan yang sebelumnya dipilih. Indeks kedua akan merepresentasikan pekerjaan yang selanjutnya akan dipilih. Sebagai contoh, pada matriks indeks [1][2] menunjukkan kecenderungan dipilihnya pekerjaan 2 setelah dipilihnya pekerjaan 1. Nilai feromon pada matriks dengan 2 indeks yang sama akan selalu bernilai 0.

Perlu diingat pula bahwa algoritma *ant colony* akan selalu memberikan peluang dipilihnya sebuah solusi. Meskipun solusi tersebut sudah dapat dianggap sebagai solusi yang kurang optimal, solusi tersebut harus tetap memiliki kemungkinan untuk dipilih. Oleh karena itu, indeks nilai feromon harus tetap dapat mengacu pada terpilihnya suatu solusi-solusi yang ada. Nilai feromon masing-masing indeks matriks (kecuali matriks dengan dua indeks yang sama) tidak akan lebih kecil dari 1. Hal ini dilakukan agar setiap solusi masih memiliki kemungkinan untuk dipilih, meskipun kemungkinannya sangat kecil.

3.2.3 Proses Pemilihan Urutan Pengerjaan / Jalur

Pada proses pemilihan jalur, nilai feromon yang disimpan oleh suatu indeks matriks mempengaruhi kemungkinan dipilihnya suatu pekerjaan. Semakin besar nilai feromon pada suatu indeks, semakin besar kemungkinan dipilihnya pekerjaan yang dirujuk oleh indeks matriks tersebut. Berdasarkan rumus untuk pemilihan jalur pada algoritma *ant colony*, diperlukan nilai objektif dan nilai feromon untuk penentuan pemilihan suatu jalur. Pada proses pemilihan pekerjaan ini, nilai objektif yang digunakan adalah 1 untuk semua indeks matriks. Proses pemilihan pekerjaan ini hanya akan dipengaruhi oleh nilai feromon yang disimpan pada indeks matriks yang ada.

Penentuan pekerjaan mana yang akan dipilih dapat dimulai dengan melakukan proses randomisasi suatu angka. Angka tersebut akan berada di kisaran nilai 1 hingga jumlah nilai feromon dari indeks-indeks yang mungkin dilibatkan. Penentuan pekerjaan dilakukan dengan menambahkan secara satu per satu nilai feromon dari indeks yang terlibat dimulai dari nilai 0. Jika setelah ditambahkan nilai dari suatu indeks, jumlah nilai telah melebihi angka hasil randomisasi, maka pekerjaan yang dirujuk indeks tersebut akan dipilih sebagai pekerjaan yang diambil.



Gambar 3.1: Ilustrasi proses pemilihan pekerjaan selanjutnya

Berdasarkan Gambar 3.1, Dapat dilihat bahwa masing-masing dari pekerjaan tersebut memiliki kemungkinan untuk dipilih sesuai dengan nilai feromonnya masing-masing. Sebagai contoh, pekerjaan 1 memiliki rentang 5 buah angka dari 11 buah angka untuk dipilih sebagai pekerjaan yang akan diambil. Proses penentuan pekerjaan ini tentu saja memperhatikan rumus awal untuk penentuan jalur dalam algoritma *ant colony*. Mengingat nilai objektif yang akan digunakan adalah 1, proses pemilihan jalur hanya akan ditentukan berdasarkan nilai feromon dari jalur-jalur yang terlibat. Berdasarkan rumus tersebut, dapat disimpulkan bahwa kemungkinan dipilihnya suatu pekerjaan adalah sebanyak jumlah feromon yang disimpan untuk pekerjaan tersebut dari jumlah total feromon dari semua pekerjaan yang mungkin dipilih.

Proses pemilihan jalur dilakukan secara bertahap, dimulai dari pemilihan jalur pertama. Untuk pemilihan jalur pertama, nilai feromon yang mempengaruhi terpilihnya suatu jalur merupakan jumlah nilai feromon pada matriks dengan indeks kedua yang merujuk pada jalur tersebut. Sebagai contoh, jika dimisalkan terdapat 2 buah pekerjaan, maka jumlah nilai matriks pada indeks [1][1] dan indeks [2][1] merupakan nilai feromon untuk terpilihnya pekerjaan 1 untuk dikerjakan pertama kali.

Untuk pemilihan pekerjaan selanjutnya dilakukan dengan representasi normal dari matriks tersebut. Indeks pertama berfungsi sebagai penunjuk pekerjaan yang diambil sebelumnya dan indeks kedua merupakan penunjuk pekerjaan yang akan diambil selanjutnya. Proses pemilihan ini akan memperhatikan pekerjaan-pekerjaan yang sebelumnya telah dipilih. Sebagai contoh, jika dimisalkan terdapat 4 buah pekerjaan dan urutan pekerjaan yang sudah dipilih adalah [1,2], maka indeks matriks yang akan diperhatikan dalam proses pemilihan hanya indeks matriks [2][3] dan [2][4]. Indeks matriks [2][1] tidak diperhatikan karena sudah diambil untuk dikerjakan pertama kali.

3.2.4 Proses Penambahan Feromon

Setiap solusi yang dicoba akan menghasilkan *makespan* yang berbeda-beda. Nilai *makespan* yang dihasilkan mempengaruhi tingkat keoptimalan dari solusi tersebut. Semakin kecil nilai *makespan* yang dihasilkan, semakin besar tingkat keoptimalan dari suatu solusi.

Nilai feromon yang disimpan harus mampu mempengaruhi semut-semut untuk lebih sering memilih jalur yang lebih optimal. Oleh karena itu, nilai feromon pada solusi optimal harus lebih cepat bertambah. Hal tersebut dapat dilakukan dengan memanipulasi banyak feromon yang ditambahkan. Semakin kecil nilai *makespan* yang dihasilkan, semakin besar nilai feromon yang akan disimpan / ditambahkan.

Nilai feromon yang akan ditambahkan merupakan hasil proses pengurangan antara nilai *makespan* yang dihasilkan dengan suatu nilai tetap. Nilai tetap yang akan digunakan merupakan salah satu nilai *makespan* yang dihasilkan dari solusi yang akan dianggap sebagai nilai tengah dari *makespan* yang mungkin dihasilkan. Solusi yang akan dijadikan nilai tengah ini merupakan solusi yang menggunakan urutan nama / nomor pekerjaan sebagai urutan pengerjaan. Sebagai contoh, jika terdapat 3 buah pekerjaan, maka solusi yang dijadikan nilai tengah adalah solusi dengan urutan pengerjaan [1,2,3].

Banyaknya feromon yang ditambahkan tidak akan bernilai negatif. Jika feromon yang ditambahkan bernilai negatif, maka penambahan feromon dari solusi tersebut tidak akan dilakukan. Penambahan feromon akan dilakukan pada indeks-indeks matriks tertentu berdasarkan solusi yang diambil. Sebagai contoh, jika solusi yang diambil adalah [2,1,3], maka indeks-indeks matriks yang akan ditambahkan feromonnya adalah indeks matriks [2,1] dan [1,3]. Dengan ditambahkannya matriks pada indeks-indeks tersebut, diharapkan indeks matriks tersebut akan lebih sering diacu dalam proses pengambilan jalur. Dengan ditambahkannya feromon pada indeks-indeks tersebut juga diharapkan solusi dengan urutan pengerjaan [2,1,3] akan lebih sering diambil.

3.3 Proses Optimisasi Penjadwalan *Hybrid Flow Shop* Dengan Menggunakan Algoritma *Ant Colony*

Pada saat melakukan proses optimisasi, algoritma *ant colony* akan melakukan proses pelatihan secara berulang-ulang. Pada setiap proses pelatihan, algoritma *ant colony* menyebarkan sejumlah semut yang akan mengerjakan kasus *hybrid flow shop* sesuai solusi yang dipilih masing-masing semut. Solusi-solusi tersebut dipilih secara acak berdasarkan nilai feromon yang disimpan. Solusi-solusi terbaik yang telah dipilih oleh suatu semut akan disimpan dan dibandingkan dengan solusi terbaik dari proses pelatihan sebelumnya.

Penambahan dan evaporasi nilai feromon yang disimpan akan dilakukan setiap suatu fase pelatihan dari algoritma *ant colony* selesai dilakukan. Nilai feromon yang ditambahkan akan disesuaikan dengan solusi dan hasil yang didapat oleh masing-masing semut. Semakin baik hasil yang diberikan suatu solusi, semakin banyak nilai feromon yang ditambahkan pada indeks feromon yang mengacu pada solusi tersebut. Proses pelatihan akan terus dilakukan hingga suatu kondisi berhenti dicapai. Kondisi berhenti tersebut merupakan penanda bahwa hasil solusi yang dipilih sudah dapat dianggap sebagai solusi yang optimal.

Algoritma *ant colony* merupakan algoritma optimisasi yang cukup terkenal. Keunggulan-keunggulan dari algoritma ini diantaranya adalah :

- Mampu dikerjakan secara paralel.
- Mampu diaplikasikan pada berbagai macam permasalahan.
- Mampu memberikan salah satu solusi yang dapat dianggap baik secara cepat.

Akan tetapi algoritma ini tentu saja memiliki kelemahan, diantaranya :

- Solusi terbaik pasti ditemukan, akan tetapi tidak ada lama waktu yang pasti untuk ditemukannya solusi optimal tersebut.
- Analisa secara teori tidak mungkin dilakukan, karena menggunakan pengambilan solusi secara acak dan tidak tetap.

Untuk menangani kelemahan-kelemahan tersebut, algoritma *ant colony* perlu menjabarkan jumlah pelatihan yang akan dilakukan dan banyaknya semut yang akan disebar pada satu kali proses pelatihan. Hal-hal tersebut perlu ditentukan agar proses pelatihan mampu menemukan solusi yang optimal dalam waktu yang relatif cepat. Semakin banyak proses pelatihan dan jumlah semut yang disebar, semakin besar kemungkinan ditemukannya solusi yang optimal. Akan tetapi pada saat yang bersamaan akan meningkatkan waktu pelaksanaan proses optimisasi. Baik atau tidaknya suatu proses optimisasi ditentukan oleh tingkat keoptimalan solusi yang dihasilkan dan waktu yang diperlukan untuk proses optimisasi.

3.3.1 Banyaknya Semut

Jumlah semut dalam proses optimisasi algoritma *ant colony* sangat berpengaruh pada tingkat keoptimalan dari solusi yang dihasilkan. Semakin banyak jumlah semut yang disebar, maka tingkat keoptimalan solusi yang diberikan akan cenderung lebih baik. Akan tetapi semakin banyak semut yang disebar akan menambah tingkat kompleksitas algoritma ini dan menambah waktu yang diperlukan untuk proses optimisasi. Jumlah semut yang disebar perlu ditentukan agar algoritma mampu memberikan solusi yang dapat dianggap cukup baik dalam waktu yang relatif cepat.

Pencarian solusi optimal untuk suatu kasus *hybrid flow shop* dapat dilakukan secara *brute force*. Pencarian solusi secara *brute force* dapat dianggap sebagai cara yang paling tidak efisien. *Brute force* akan menjabarkan solusi-solusi yang ada dan mencoba penyelesaian dengan menggunakan masing-masing solusi tersebut. Jika proses optimisasi diaplikasikan pada sebuah permasalahan, sebaiknya proses optimisasi tersebut dapat berjalan lebih cepat dibandingkan dengan cara *brute force*.

Jika permasalahan *hybrid flow shop* akan diselesaikan secara *brute force*, maka setiap solusi yang ada akan dijabarkan. Solusi-solusi pada *hybrid flow shop* dapat dicari dengan melakukan proses fungsi kombinasi pada banyaknya pekerjaan yang ada. Dilihat dari sisi kompleksitas, cara *brute force* memiliki tingkat kompleksitas n!, di mana n merupakan banyaknya pekerjaan yang ada.

Jumlah semut yang akan disebar sebaiknya kurang dari tingkat kompleksitas dengan cara *brute force* tersebut. Hal ini dilakukan agar optimisasi algoritma *ant colony* ini menjadi relatif lebih cepat dibandingkan dengan cara *brute force*. Jumlah semut yang optimal masih belum diketahui, oleh karena itu sekurang-kurangnya perlu diketahui pengaruh dari jumlah semut yang disebar terhadap hasil optimisasi algoritma ini.

3.3.2 Aturan Berhenti

Ada berbagai cara yang dapat dipakai dalam menentukan waktu berhenti dari proses optimisasi algoritma *ant colony*. Beberapa parameter yang dapat digunakan untuk penentuan waktu berhenti di antaranya :

- Nilai data masukan yang diberikan : algoritma akan terus melakukan pelatihan sebanyak n kali, di mana n memiliki hubungan aritmatik dengan salah satu data masukan kasus.
- Nilai data keluaran yang dihasilkan : algoritma akan terus melakukan pelatihan hingga data keluaran yang dihasilkan bernilai sama dengan data-data keluaran sebelumnya.
- Sesuai dengan keinginan pengguna : algoritma akan terus melakukan pelatihan hingga pengguna merasa hasil akhir yang diberikan sudah cukup optimal.

Perlu diingat pula bahwa salah satu dari kekurangan algoritma *ant colony* adalah tidak adanya lama waktu yang pasti untuk ditemukannya solusi yang paling optimal. Algoritma *ant colony* menggunakan proses randomisasi untuk penentuan solusi-solusi optimal. Meskipun proses randomisasi tetap dipengaruhi oleh feromon-feromon pada indeks matriks, tidak berarti bahwa solusi yang lebih optimal akan selalu terpilih. Proses randomisasi pada algoritma tetap memiliki kemungkinan terpilihnya solusi yang kurang optimal secara terus menerus.

Berdasarkan hal-hal tersebut, aturan berhenti algoritma ini akan disesuaikan dengan keinginan pengguna. Jika aturan berhenti algoritma dilakukan berdasarkan nilai data masukan, hasil yang akan diberikan belum tentu dapat dianggap sebagai hasil yang baik. Tidak ada jaminan bahwa hasil akhir yang baik jika dilakukan proses pelatihan sebanyak n kali. Jika aturan berhenti disesuaikan dengan data keluaran yang dihasilkan, belum tentu data keluaran yang diberikan merupakan solusi yang baik. Algoritma *ant colony* memiliki kemungkinan memilih solusi yang kurang optimal secara terus menerus, meskipun kemungkinan tersebut sangatlah kecil.

3.3.3 Penggunaan *Thread*

Perlu diingat pula bahwa salah satu keunggulan dari algoritma *ant colony* ini adalah mampu dikerjakan secara paralel. Sifat paralel ini dapat diartikan bahwa algoritma ini mampu melakukan pembagian tugas dalam melakukan proses perhitungan dalam optimisasi. Hal ini membuat algoritma ini mampu melakukan lebih dari satu proses perhitungan dalam satu waktu dan membuat lama waktu optimisasi menjadi relatif lebih cepat. Besarnya pengaruh sifat paralel terhadap kecepatan optimisasi dapat menjadi cukup besar. Semakin banyak proses perhitungan yang dapat dilakukan pada satu waktu, maka semakin cepat pula proses optimisasi berlangsung.

Sifat paralel dari algoritma ini dapat diaplikasikan dengan menggunakan kelas Thread dari bahasa pemrograman Java. Kelas Thread ini mampu membentuk suatu ruang baru untuk melakukan proses perhitungan lainnya. Sebuah Thread mampu mengatur berlangsungnya satu buah proses perhitungan. Jika pada proses optimisasi digunakan beberapa buah objek dari kelas Thread, maka sifat paralel dari algoritma *ant colony* dapat diaplikasikan.

Perlu diketahui pula bahwa sebuah objek dari kelas Thread sulit untuk membentuk keterhubungan dengan kelas Thread yang lain, oleh karena itu diperlukan sebuah kelas pusat yang mampu mengatur jalannya seluruh Thread yang ada. Proses perhitungan yang dikerjakan oleh masing-masing objek Thread juga sebaiknya tidak berhubungan satu sama lain. Pada proses optimisasi dengan menggunakan algoritma *ant colony*, proses perhitungan yang dapat diparalelkan adalah proses pembentukkan solusi acak dan penilaian/perhitungan tingkat keoptimalan dari solusi tersebut. Objek yang dapat dijalankan pada sebuah Thread dapat dibentuk dengan mengimplementasi *interface* Runnable.

3.4 Aturan-Aturan Dalam Proses Penjadwalan *Hybrid Flow Shop*

Selama proses optimisasi dengan menggunakan algoritma *ant colony*, masing-masing semut akan menjalankan kasus *hybrid flow shop* berdasarkan solusi yang dipilihnya. *Hybrid flow shop* memiliki banyak perangkat mesin yang berbeda untuk menyelesaikan suatu proses. Jumlah mesin yang berbeda-beda mampu mengakibatkan antrian dan memerlukan prioritas untuk penentuan mesin / pekerjaan yang akan digunakan / dikerjakan. Diperlukan aturan-aturan khusus untuk menjalankan proses *hybrid flow shop* tersebut.

Mesin-mesin pada *hybrid flow shop* akan selalu siap untuk menerima pekerjaan bila mesin sedang menunggu dan tidak ada pekerjaan di dalamnya. Jika terdapat sebuah proses yang sudah selesai dikerjakan dan terdapat mesin untuk proses selanjutnya yang sedang menunggu, maka pekerjaan tersebut akan langsung dioper ke mesin tersebut. Mesin yang akan menerima pekerjaan dipriori-

taskan berdasarkan nomor mesin yang dimilikinya. Sebagai contoh, mesin 3.2 memiliki prioritas lebih besar untuk menerima pekerjaan dibandingkan dengan mesin 3.4.

Jika terdapat lebih dari satu proses yang sudah selesai dikerjakan, maka urutan awal pengambilan pekerjaan akan menjadi prioritas mengenai proses mana yang dioper dahulu. Sebagai contoh, jika proses 1 dari pekerjaan 1 dan 2 sudah siap dioper dan urutan pengambilan awal pekerjaan adalah [2,3,1], maka proses 1 dari pekerjaan 2 akan siap untuk dioper terlebih dahulu. Aturan-aturan ini memungkinkan berbedanya urutan selesainya pekerjaan dengan urutan masuknya pekerjaan. Jika terdapat suatu pekerjaan dengan lama waktu proses yang sangat tinggi, maka pekerjaan dengan waktu proses yang lebih rendah mampu mendahului pekerjaan tersebut dan lebih cepat memulai proses selanjutnya. Hal ini dapat terjadi jika terdapat lebih dari 1 mesin yang mampu mengerjakan suatu proses.

Selain peraturan-peraturan di atas, aturan dasar dalam pengerjaan proses *hybrid flow shop* juga akan diaplikasikan. Aturan-aturan tersebut merupakan:

- Semua penyelesaian pekerjaan mengikuti alur proses yang sama dan sistematis.
- Sebuah proses dianggap telah selesai jika terdapat mesin yang telah mengerjakannya.

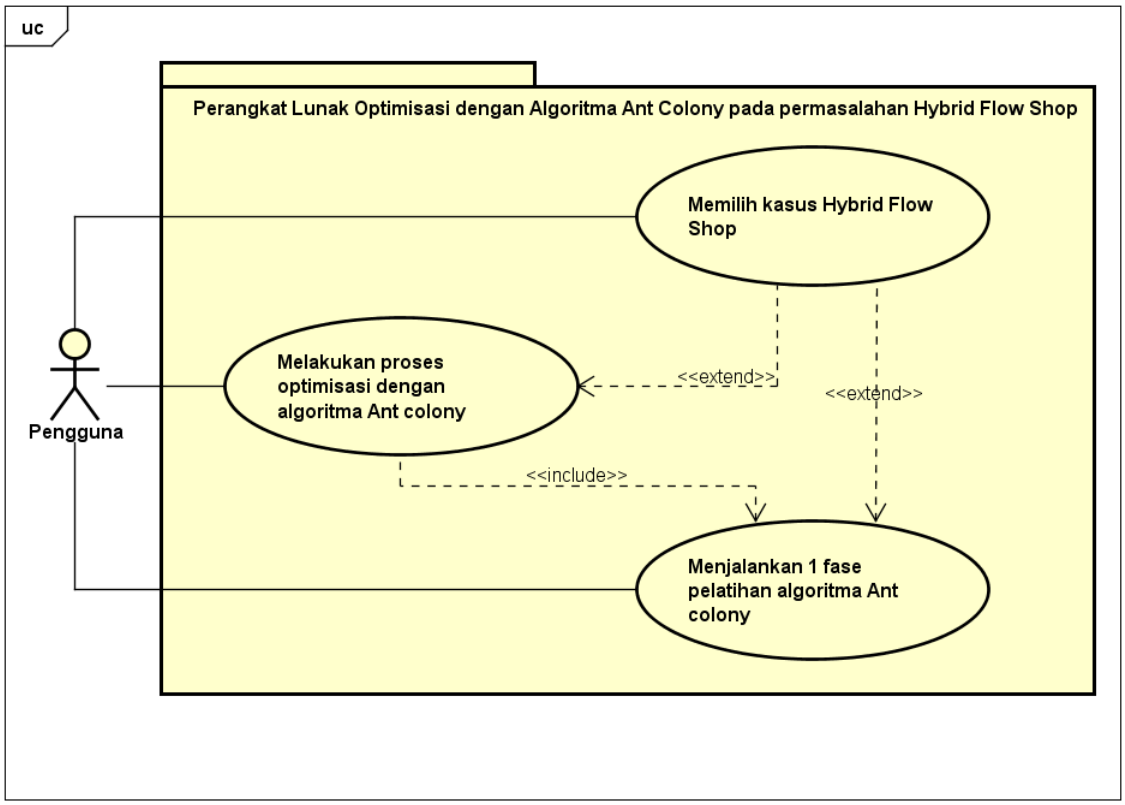
Masing-masing proses dari sebuah pekerjaan pasti akan dikerjakan dengan urutan-urutan yang sama dan pengerjaan suatu proses tidak dapat dilakukan jika proses-proses sebelumnya belum selesai dikerjakan.

3.5 Analisis Perangkat Lunak

Perangkat lunak akan melakukan optimisasi penjadwalan *hybrid flow shop* dengan menggunakan algoritma *ant colony*. Optimisasi dilakukan agar urutan pengerjaan yang dipilih mampu menghasilkan *makespan* yang paling minimum. Algoritma *ant colony* akan memilih urutan pengerjaan yang paling baik dari suatu kasus berdasarkan solusi-solusi yang dipilih oleh semut-semut dalam setiap proses pelatihan. Perangkat lunak ini kemudian akan menampilkan hasil urutan pengerjaan / solusi optimal dari kasus *hybrid flow shop* yang dipilih. Solusi tersebut akan ditampilkan dengan nilai *makespan* yang dihasilkan dari solusi tersebut.

3.5.1 *Use Case Diagram*

Perangat Lunak yang dibuat akan mampu berinteraksi dengan pengguna. Pengguna akan mampu menjabarkan kasus *hybrid flow shop* yang ingin dicari hasil optimalnya. Pengguna juga akan dapat mampu mengatur kapan dan bagaimana berlangsungnya suatu proses optimisasi. Untuk mewujudkan interaksi tersebut, berikut adalah rancangan *use case diagram* yang menjabarkan hal-hal yang mampu dilakukan pengguna melalui perangkat lunak ini.



Gambar 3.2: *Use Case Diagram*

Keterangan :

- Memilih kasus *hybrid flow shop*.
  Pengguna dapat memilih kasus *hybrid flow shop* yang ingin dicari hasil optimalnya. Kasus tersebut akan dijabarkan dalam sebuah *file* teks dengan format penulisan tertentu. Pengguna dapat memilih salah satu dari *file* teks tersebut sebagai kasus *hybrid flow shop* yang ingin dicari hasil optimalnya. Pengguna hanya dapat melakukan proses optimisasi dengan menggunakan algoritma *ant colony* setelah pengguna memilih kasus *hybrid flow shop* yang ingin dioptimisasi.

- Menjalankan 1 fase pelatihan algoritma *ant colony*.
  Pengguna dapat menjalankan 1 kali fase pelatihan dari algoritma *ant colony*. Proses pelatihan ini akan mengembalikan informasi mengenai hasil optimal yang didapat pada fase pelatihan ini.

- Melakukan proses optimisasi dengan menggunakan algoritma *ant colony*.
  Proses optimisasi ini akan menjalankan fase pelatihan algoritma *ant colony* secara terus-

menerus hingga kondisi berhenti yang ditentukan tercapai. Informasi hasil optimal yang didapatkan pada setiap fase pelatihan akan ditampilkan.

3.5.2 *Input* dan *Output*

Perangkat lunak akan mampu menerima data masukan dari sebuah *file* teks. *File* teks tersebut akan berisi detil mengenai kasus *hybrid flow shop* yang ingin diselesaikan. *File* teks tersebut akan berisi detil mengenai banyaknya pekerjaan, banyaknya proses, banyaknya mesin untuk setiap proses, dan detil waktu proses untuk setiap pekerjaan. *File* teks tersebut akan memiliki format penulisan tertentu, agar *file* teks tersebut dapat dikonversi menjadi suatu kasus *hybrid flow shop*.

Baris pertama dari teks akan berisi sebuah angka yang mendefinisikan banyaknya pekerjaan yang ada. Baris kedua akan berisi sebuah angka yang mendefinisikan banyaknya proses pada pekerjaan. Baris selanjutnya akan menjelaskan banyaknya mesin yang mampu mengerjakan masing-masing proses. Baris-baris selanjutnya akan menjabarkan detil waktu proses untuk masing-masing pekerjaan. Sebuah baris akan menjelaskan detil waktu sebuah pekerjaan secara terurut berdasarkan proses. Setelah penjabaran detil waktu proses selesai, 2 baris terakhir dari *file* teks akan menjabarkan nilai *upper bound* dan *lower bound* dari kasus tersebut.

```
1  5
2  5
3   1   1   1   1   1
4  10  11  29  10   4
5  24  14  28  59  32
6  34  75  31  12  83
7  24  84   7  42   3
8  34  93  38   4   5
9  357
10 340
```

Gambar 3.3: Contoh format data masukan untuk suatu kasus *hybrid flow shop*

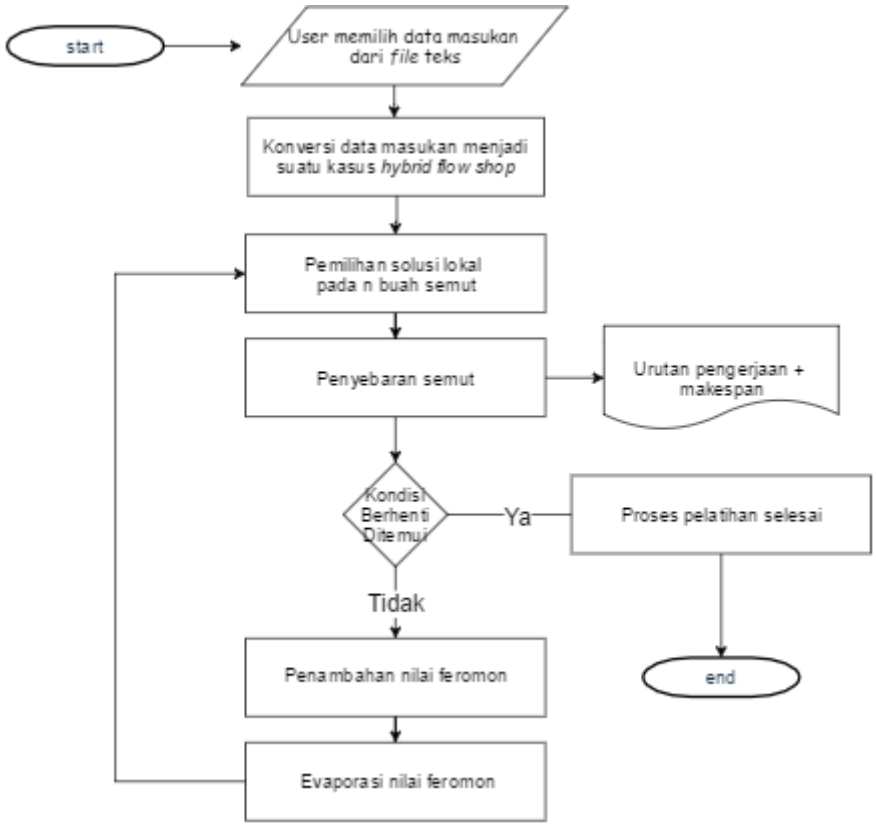
Contoh data masukan dari gambar 3.3 merupakan contoh kasus *hybrid flow shop* dengan 20 buah pekerjaan dan masing-masing pekerjaan memiliki 5 proses. Baris ke-3 menjabarkan detil mengenai banyaknya perangkat mesin yang mampu mengerjakan masing-masing proses. Detil waktu proses dari pekerjaan-pekerjaan tersebut dijabarkan mulai dari baris ke-4. Baris ke-4 menjabarkan detil waktu proses dari pekerjaan 1 dengan waktu pengerjaan proses 1 selama 10 satuan waktu, proses 2 selama 11 satuan waktu, dan seterusnya. Dua baris terakhir dari *file* teks akan menjabarkan nilai *upper bound* (357) dan *lower bound* (340) dari kasus tersebut.

Nilai *upper bound* akan dianggap sebagai nilai *makespan* optimal / nilai *makespan* target yang ingin dicapai oleh proses optimisasi. Nilai *lower bound* akan dianggap sebagai nilai *makespan* minimal yang mungkin dicapai dari suatu kasus. Nilai *lower bound* ini akan dijadikan batas nilai *makespan* yang mungkin dihasilkan dari suatu kasus. Nilai *lower bound* ini akan digunakan untuk pengujian perangkat lunak. Jika nilai *makespan* yang dihasilkan lebih kecil dari nilai *lower bound*, maka kemungkinan terdapat kesalahan pada perangkat lunak.

Setelah mengkonversi data masukan menjadi suatu kasus *hybrid flow shop*, perangkat lunak akan melakukan proses optimisasi. Proses optimisasi akan menghasilkan suatu solusi berupa urutan pengerjaan yang menghasilkan *makespan* yang dianggap optimal. Perangkat lunak kemudian akan menampilkan di layar pengguna mengenai urutan pengerjaan yang dianggap optimal beserta nilai *makespan* yang dihasilkan. Perangkat lunak juga akan menampilkan banyaknya pelatihan yang telah dilakukan untuk mendapat urutan pengerjaan tersebut.

3.5.3 Proses Optimisasi

Proses optimisasi dengan algoritma *ant colony* dilakukan setelah mengkonversi data masukan menjadi sebuah kasus *hybrid flow shop*. Algoritma *ant colony* akan menyesuaikan diri dengan kasus *hybrid flow shop* yang ingin dicari solusi optimalnya. Algoritma *ant colony* akan melakukan proses optimisasi dengan cara melakukan proses pelatihan secara berulang-ulang hingga suatu kondisi berhenti tercapai. Berikut adalah *flow chart* mengenai bagaimana perangkat lunak ini akan melakukan proses optimisasi dengan menggunakan algoritma *ant colony*.



Gambar 3.4: *Flow chart* proses optimisasi yang dilakukan oleh perangkat lunak

- **Konfigurasi data masukan**
  Membaca data masukan, lalu kemudian menyimpannya menjadi suatu kasus *hybrid flow shop*. Pada tahap ini, konfigurasi pada algoritma *ant colony* juga dilakukan. Konfigurasi tersebut di antaranya adalah membuat matriks penyimpanan feromon sesuai dengan banyaknya pekerjaan pada kasus, membuat mesin *hybrid flow shop* sesuai jumlah proses dan perangkat mesin yang ditentukan.

- **Pemilihan solusi lokal**
  Pada tahap ini, masing-masing semut akan diberikan salah satu solusi untuk penyelesaian kasus *hybrid flow shop* yang dimiliki. Solusi yang diberikan pada semut-semut ditentukan secara acak berdasarkan nilai feromon yang disimpan. Pembentukan solusi berdasarkan nilai feromon dilakukan secara bertahap sesuai dengan tata cara yang telah ditentukan.

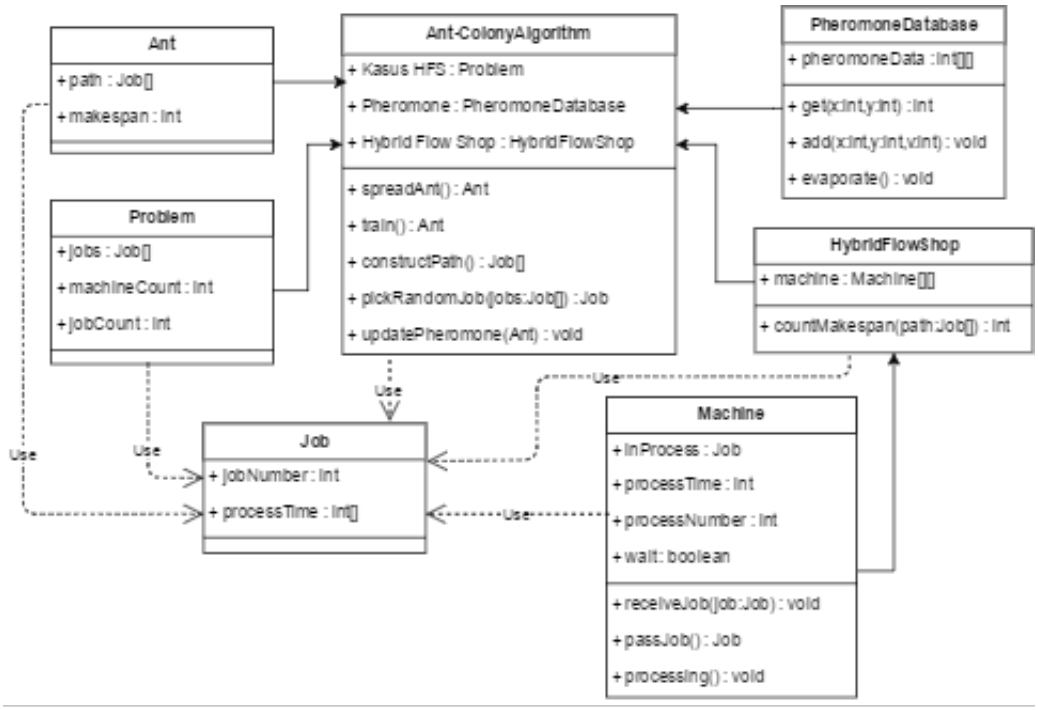
- **Penyebaran semut**
  Tahap ini juga dapat disebut sebagai tahap pelatihan. Pada tahap ini, semut-semut yang telah memiliki solusinya masing-masing akan menjalankan kasus *hybrid flow shop* dengan solusi yang dimilikinya. Hasil nilai *makespan* yang dihasilkan dari solusi tersebut akan disimpan dan dibandingkan dengan hasil *makespan* dari solusi yang dianggap paling optimal hingga saat ini. Jika hasil *makespan* dari suatu solusi bernilai lebih baik, maka solusi yang dipilih semut tersebut akan menggantikan solusi yang dianggap paling optimal hingga saat ini. Pada tahap ini, perangkat lunak akan menampilkan solusi dan *makespan* yang dianggap paling optimal hingga proses pelatihan saat ini.

- **Penambahan feromon**
  Pada tahap ini, perubahan nilai feromon akan dilakukan. Penambahan nilai feromon akan disesuaikan dengan solusi-solusi dan hasil yang dimiliki oleh semut pada tahap pelatihan terakhir. Masing-masing semut akan menambahkan nilai feromon berdasarkan solusi dan hasil yang dia miliki. Jumlah nilai feromon yang ditambahkan disesuaikan dengan tata cara yang ditentukan.

- **Evaporasi feromon**
  Pada tahap ini, nilai-nilai feromon yang disimpan akan dikurangi berdasarkan persentase tertentu. Tingkat evaporasi yang dipakai adalah 30%. Nilai feromon yang disimpan pada setiap indeks matriks akan dikurangi sebanyak 30%. Evaporasi feromon ini dilakukan agar nilai feromon yang disimpan tidak terlampau besar dan sulit untuk disimpan. Evaporasi feromon ini juga dilakukan dengan harapan agar solusi yang kurang optimal akan semakin sulit untuk dipilih.

3.5.4 Struktur Kelas

Perangkat lunak yang akan dibangun akan menjadikan algoritma *ant colony* sebagai kelas utama. Kelas tersebut bertugas untuk menentukan urutan pengerjaan yang akan dijalankan oleh *hybrid flow shop* dan mencari urutan pengerjaan yang paling optimal. Kelas algoritma *ant colony* akan dibantu beberapa kelas lainnya dalam menentukan urutan pengerjaan. Gambaran mengenai struktur kelas yang akan dibangun dapat dilihat pada Gambar 3.5.



Gambar 3.5: Struktur kelas untuk optimisasi *hybrid flow shop* dengan menggunakan algoritma *ant colony*

- Job
  Kelas Job akan merepresentasikan sebuah pekerjaan yang akan dikerjakan dalam sebuah kasus *hybrid flow shop*. Kelas ini akan menyimpan informasi mengenai sebuah pekerjaan sebagai atribut-atributnya. Atribut-atribut tersebut di antaranya adalah :

  – Atribut jobNumber
    Atribut ini berfungsi sebagai nomor penunjuk identitas dari sebuah pekerjaan. Atribut ini berfungsi sebagai pembeda sebuah pekerjaan dari pekerjaan lainnya dalam suatu proses penjadwalan.

  – Atribut processTime
    Atribut ini berfungsi untuk menyimpan informasi mengenai lama waktu pengerjaan dari proses-proses dalam sebuah pekerjaan. Informasi ini akan disimpan dalam bentuk *array*. Indeks *array* akan menunjukkan nomor proses dari sebuah pekerjaan. Sebagai contoh, indeks *array* pertama akan menunjukkan waktu pengerjaan proses 1 dari sebuah pekerjaan.

- Problem
  Kelas ini merepresentasikan sebuah kasus *hybrid flow shop*. Kelas ini akan menyimpan informasi mengenai suatu kasus *hybrid flow shop* sebagai atribut. Secara detil, berikut adalah atribut-atribut yang disimpan.

  – Atribut jobs
    Berisi informasi mengenai pekerjaan-pekerjaan yang ada.

  – Atribut machineCount

Berisi informasi mengenai banyaknya mesin / proses.

- Atribut jobCount
  Berisi informasi mengenai banyaknya pekerjaan.

- Machine
  Kelas Machine akan merepresentasikan sebuah mesin pada *hybrid flow shop*. Masing-masing kelas mesin akan mampu mengerjakan sebuah pekerjaan pada satu waktu. Kelas mesin juga akan mampu mengoper pekerjaan yang telah selesai dia kerjakan dan juga menerima pekerjaan baru jika mesin tidak sedang mengerjakan pekerjaan apapun. Berikut adalah detil mengenai atribut dan operasi yang dimiliki oleh sebuah kelas Machine.

  - Atribut inProcess
    Atribut ini berfungsi sebagai penunjuk mengenai pekerjaan yang saat ini sedang dikerjakan oleh sebuah mesin. Jika mesin tidak sedang mengerjakan pekerjaan apapun, maka atribut ini akan bernilai *null*.
  - Atribut processTime
    Atribut ini berfungsi sebagai penunjuk lama waktu yang telah ditempuh dalam mengerjakan proses dari sebuah pekerjaan.
  - Atribut processNumber
    Atribut ini berfungsi sebagai penunjuk mengenai nomor proses yang harus dikerjakan oleh kelas Machine tersebut.
  - Atribut wait
    Atribut ini berfungsi sebagai penanda bahwa mesin sedang senggang dan tidak mengerjakan pekerjaan apapun. Jika atribut ini bernilai *true*, maka mesin sedang senggang dan dianggap mampu menerima pekerjaan baru. Sebuah mesin dapat dianggap sedang senggang jika atribut inProcess dari mesin bernilai *null*. Jika mesin tidak sedang mengerjakan / menyimpan sebuah pekerjaan, maka atribut wait ini akan bernilai *true*.
  - Operasi receiveJob
    Operasi ini berfungsi untuk memberikan mesin sebuah pekerjaan untuk dikerjakan. Sebuah mesin hanya mampu menerima pekerjaan baru, hanya jika atribut wait dari mesin tersebut bernilai *true*.
  - Operasi passJob
    Operasi ini berfungsi untuk mengoper pekerjaan yang telah selesai dikerjakan oleh sebuah mesin. Sebuah mesin hanya mampu mengoper pekerjaan jika atribut processTime telah mencapai lama waktu pengerjaan dari sebuah proses pada pekerjaan di atribut inProcess.
  - Operasi processing
    Operasi ini berfungsi menambah nilai dari atribut processTime pada sebuah mesin. Operasi ini tidak akan menambah nilai atribut processTime jika mesin tidak sedang mengerjakan pekerjaan apapun / pekerjaan pada mesin telah selesai dikerjakan.

- HybridFlowShop
  Kelas HybridFlowShop berfungsi sebagai pihak yang mengatur keterhubungan antar kelas Machine. Kelas ini merupakan kelas yang mengatur kapan sebuah mesin diperbolehkan mengoper

dan menerima sebuah pekerjaan. Kelas ini juga mampu mengukur lama waktu yang diperlukan (*makespan*) untuk pengerjaan pekerjaan-pekerjaan dalam urutan tertentu. Atribut dan operasi yang mampu dilakukan kelas ini di antaranya :

- Atribut machine
  Atribut ini merupakan kumpulan mesin yang terdapat dalam sebuah sistem *hybrid flow shop*. Mesin-mesin tersebut disimpan dalam *array* 2 dimensi, karena *hybrid flow shop* mampu memiliki lebih dari 1 perangkat mesin untuk pengerjaan sebuah proses. Indeks pertama dari *array* akan menjadi penunjuk nomor proses, sedangkan indeks kedua merupakan penunjuk nomor mesin. Sebagai contoh, indeks *array* [3][2] merupakan penunjuk pada mesin kedua yang mampu mengerjakan proses nomor 3.
- Operasi countMakespan
  Operasi ini menghitung nilai *makespan* yang dihasilkan dari pengerjaan pekerjaan dengan urutan tertentu. Operasi ini akan menganggap terdapat sekumpulan pekerjaan yang harus dikerjakan oleh sekumpulan mesin yang dimiliki kelas ini dalam urutan tertentu. Operasi ini akan mengatur waktu pengoperan pekerjaan antara mesin-mesin yang dimiliki sesuai dengan ketentuan dan urutan yang diinginkan. Operasi ini menerima data masukan berupa *array* kelas Job. Indeks *array* berfungsi sebagai urutan pengerjaan dari pekerjaan-pekerjaan pada *array*.

- Ant
  Kelas ini berfungsi sebagai representasi seekor semut pada algoritma *ant colony*. Kelas ini akan menyimpan sebuah solusi dari hasil randomisasi algoritma *ant colony*. Nilai-nilai yang disimpan dari seekor semut akan disesuaikan dengan permasalahan yang ingin diselesaikan. Dalam permasalahan *hybrid flow shop*, nilai-nilai yang disimpan adalah :
  - Atribut path Berisi informasi mengenai urutan pengerjaan / solusi untuk *hybrid flow shop*. Informasi ini diperoleh dari proses pemilihan jalur secara acak milik algoritma *ant colony*.
  - Atribut makespan Berisi informasi mengenai lama waktu pengerjaan dengan menggunakan urutan pengerjaan pada atribut path. Informasi ini diperoleh setelah solusi dikerjakan pada kelas HybridFlowShop.

- PheromoneDatabase
  Kelas ini merepresentasikan tempat penyimpanan nilai feromon untuk panduan proses pemilihan jalur secara acak milik algoritma *ant colony*. Cara penyimpanan nilai feromon disesuaikan dengan permasalahan yang ingin diselesaikan. Untuk permasalahan *hybrid flow shop*, berikut adalah nilai yang disimpan dan hal-hal yang mampu dilakukan kelas PheromoneDatabase.
  - Atribut pheromoneData
    Berisi informasi nilai-nilai feromon yang disimpan. Untuk permasalahan *hybrid flow shop*, nilai disimpan dalam bentuk *array* 2 dimensi. Indeks pertama dari *array* merupakan penunjuk nomor pekerjaan sebelumnya, sedangkan indeks kedua menunjukkan nomor pekerjaan selanjutnya.

  - Operasi add
    Menambahkan nilai feromon pada *array* indeks [x][y] dengan nilai v.
  - Operasi get
    Mengembalikan nilai feromon pada *array* indeks [x][y].
  - Operasi evaporate
    Melakukan proses penguapan feromon. Nilai feromon pada setiap indeks matriks akan dikurangi sebanyak 30%. Hasil penguapan nilai feromon tidak akan lebih kecil dari 1.

- Ant-ColonyAlgorithm Kelas ini merupakan kelas yang akan melakukan optimisasi dengan menggunakan algoritma *ant colony*. Dalam permasalahan *hybrid flow shop*, kelas ini akan mencari urutan pengerjaan yang menghasilkan *makespan* yang optimal. Dalam pengaplikasian sifat paralel dari algoritma *ant colony*, kelas ini kemungkinan besar akan dipecah menjadi 2 bagian. Satu kelas akan digunakan untuk kelas pusat yang memantau berjalannya Thread yang ada dan satu kelas lagi akan mengimplementasi *interface* Runnable dan menjalankan proses perhitungan tingkat keoptimalan secara paralel. Dalam mencari hasil optimal tersebut, kelas ini akan dibantu oleh kelas PheromoneDatabase. Operasi dan atribut yang dimiliki oleh kelas ini adalah :

  - Atribut kasusHFS
    Berisi informasi mengenai kasus *hybrid flow shop* yang akan dicari hasil optimalnya. Informasi disimpan dalam bentuk kelas Problem.
  - Atribut pheromone
    Berisi informasi mengenai nilai-nilai feromon yang berguna untuk membantu proses optimisasi dan pencarian solusi secara acak. Atribut ini dibentuk berdasarkan nilai yang disimpan pada atribut kasusHFS.
  - Atribut hybridFlowShop
    Merupakan kelas HybridFlowShop yang berguna untuk perhitungan nilai *makespan* dari solusi acak yang dibuat oleh algoritma *ant colony*. Atribut ini dibentuk dan disesuaikan dengan permasalahan *hybrid flow shop* pada atribut kasusHFS.
  - Operasi spreadAnt
    Melakukan proses pelatihan pada algoritma *ant colony*. Proses ini dilakukan untuk mengubah nilai pada atribut pheromone, sehingga proses pencarian solusi secara acak semakin optimal. Diharapkan dengan operasi ini, proses pencarian solusi secara acak semakin sering memberikan solusi yang lebih optimal. Proses-proses pada operasi ini mencakup pembuatan solusi secara acak pada semut, penyebaran semut, perubahan nilai feromon berdasarkan solusi acak milik semut yang disebar, dan evaporasi feromon. Operasi ini mengembalikan solusi yang optimal hingga saat ini dalam bentuk kelas Ant.

    Dalam pengaplikasian sifat paralel dari algoritma *ant colony*, operasi ini akan dibagi menjadi 2 bagian. Satu bagian akan bertugas memantau dan melakukan *update* nilai feromon juga hasil solusi optimal. Satu bagian lagi akan melakukan pembentukan solusi acak dan menghitung tingkat keoptimalan dari suatu solusi. Pembentukan dan perhitungan tingkat keoptimalan dari solusi tersebut dikerjakan secara paralel.

- Operasi train

  Melakukan operasi spreadAnt hingga suatu kondisi berhenti tercapai. Operasi ini akan mengembalikan solusi yang dianggap paling optimal untuk penyelesaian kasus *hybrid flow shop*. Solusi optimal tersebut dikembalikan dalam bentuk kelas Ant.

- Operasi constructPath

  Membangun solusi / urutan pengerjaan untuk kasus *hybrid flow shop*. Solusi tersebut dibangun secara acak berdasarkan nilai yang disimpan pada atribut pheromone. Operasi ini mengembalikan data berupa *array* kelas Job. Indeks *array* berfungsi sebagai urutan pengerjaan dari pekerjaan-pekerjaan pada *array*.

- Operasi pickRandomJob

  Memilih salah satu pekerjaan dari data masukan *array* kelas Job. Pemilihan dilakukan secara acak berdasarkan nilai yang disimpan pada atribut pheromone. Operasi ini mengembalikan salah satu pekerjaan yang terpilih di antara kumpulan pekerjaan dari data masukan.

- Operasi updatePheromone

  Melakukan perubahan nilai pada atribut pheromone berdasarkan data masukan kelas Ant. Operasi ini akan melakukan penambahan nilai feromon sesuai dengan nilai *makespan* yang dimiliki oleh data masukan. Indeks feromon yang ditambahkan disesuaikan dengan solusi / urutan pengerjaan yang dimiliki oleh data masukan.

BAB 4

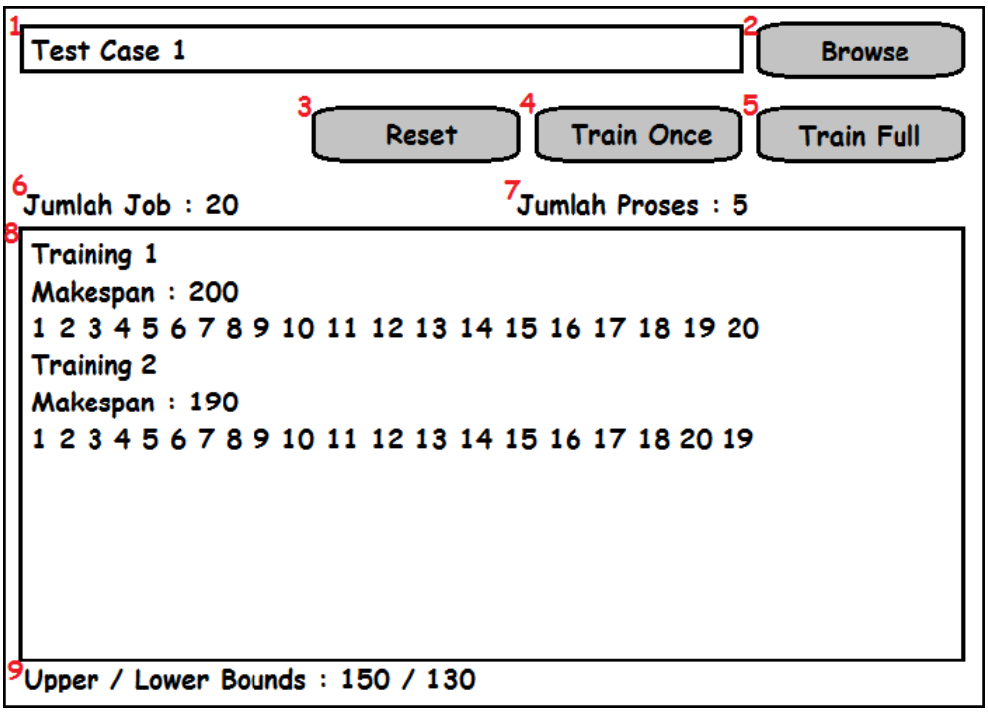
PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dibahas mengenai detil-detil dari perangkat lunak optimisasi algoritma *ant colony* untuk kasus *hybrid flow shop*. Tampilan *interface* dari perangkat lunak dan hal-hal apa saja yang mampu dilakukan pengguna melalui perangkat lunak ini akan ditampilkan. Struktur kelas dan hal-hal yang mampu dilakukan / diinformasikan melalui perangkat lunak ini juga akan dijabarkan.

4.1 Perancangan Antarmuka Grafis

Untuk mempermudah penggunaan, perangkat lunak akan memiliki sebuah *interface*. *Interface* ini dibuat agar pengguna perangkat lunak dapat dengan mudah menjabarkan kasus *hybrid flow shop* yang ingin dioptimisasi. *Interface* ini juga dibuat agar proses optimisasi yang dilakukan dapat lebih mudah dipantau. Melalui *interface* ini, pengguna dapat memberikan data masukan kasus *hybrid flow shop* melalui sebuah *file* teks. Pengguna juga dapat mengetahui hasil optimisasi yang diberikan pada setiap fase pelatihan.

Berikut adalah rancangan *interface* yang akan dibuat.



Gambar 4.1: Rancangan *interface* perangkat lunak

Adapun hal-hal yang dapat dilakukan oleh *interface* ini adalah :

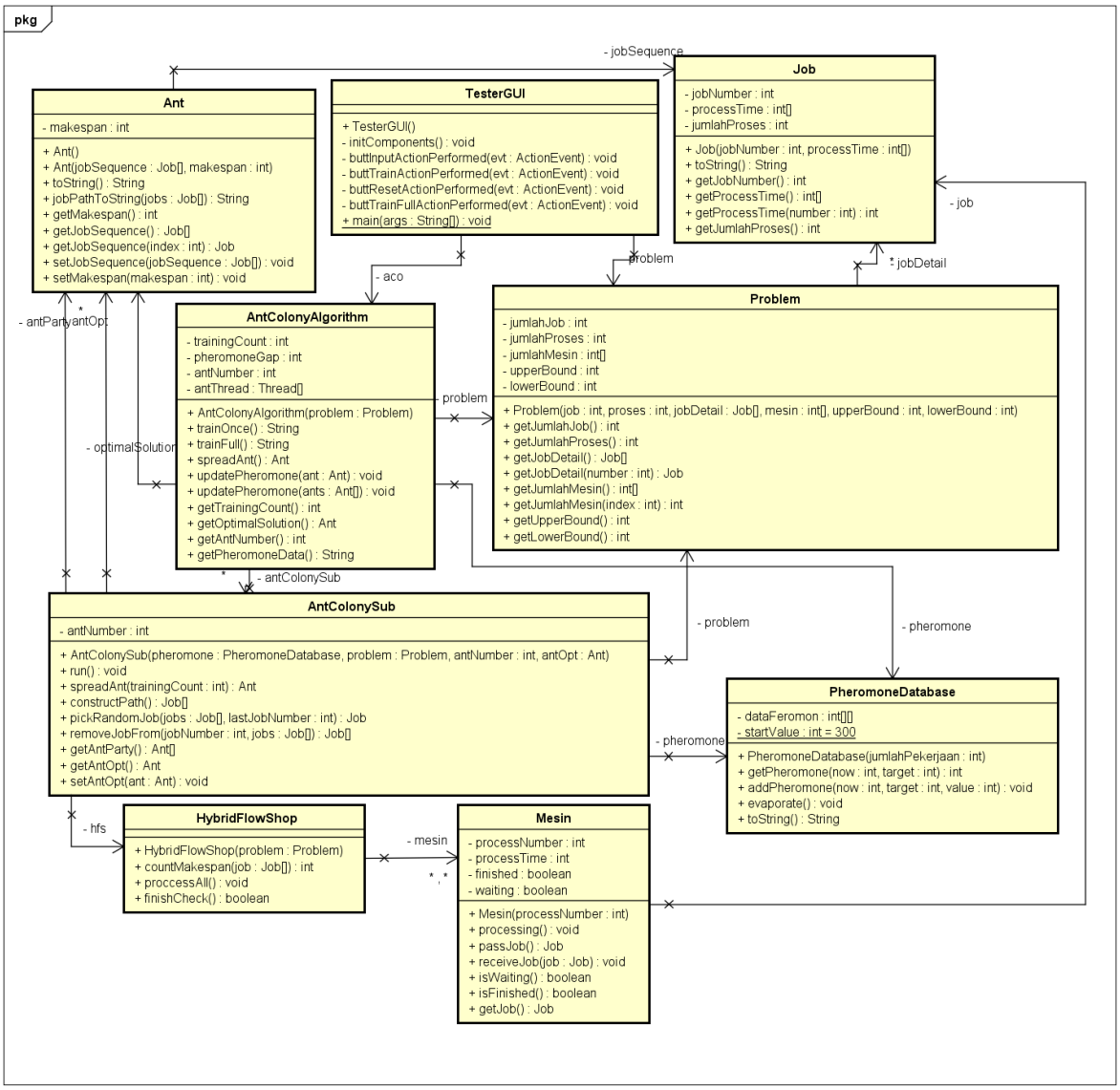
1. Memberitahu informasi nama *file* teks yang telah dipilih pengguna untuk dioptimisasi.
2. Memilih *file* teks yang berisi satu kasus *hybrid flow shop*. *File* yang dipilih harus berupa *file* .txt. Agar *file* dapat dikonversi menjadi sebuah kasus *hybrid flow shop*, data di dalam *file* tersebut juga harus sesuai dengan format penulisan yang ditentukan.
3. Mengulang kembali proses optimisasi dari fase pelatihan pertama.
4. Melakukan proses pelatihan dengan menggunakan algoritma *ant colony* sebanyak 1 kali pada kasus *hybrid flow shop* yang dipilih.
5. Melakukan proses pelatihan dengan menggunakan algoritma *ant colony* pada kasus *hybrid flow shop* yang dipilih hingga kondisi berhenti dicapai. Kondisi berhenti yang digunakan adalah diberikannya solusi optimal yang sama pada 10 proses pelatihan terakhir.
6. Menampilkan banyaknya pekerjaan yang ada pada kasus *hybrid flow shop* yang dipilih.
7. Menampilkan banyaknya mesin / proses yang harus dilalui pada kasus *hybrid flow shop* yang dipilih.
8. Menampilkan informasi mengenai solusi optimal yang diberikan pada setiap proses pelatihan. Informasi yang diberikan berupa urutan pengerjaan yang dilakukan, nilai *makespan* yang dihasilkan, dan pada proses pelatihan ke berapa solusi tersebut didapatkan.
9. Menampilkan informasi nilai *upper bound* dan *lower bound* dari kasus *hybrid flow shop* yang dipilih.

4.2 Diagram Kelas Lengkap

Perangkat lunak akan dibuat secara *Object Oriented*. Kelas-kelas yang dibuat akan memperhatikan kebutuhan dan keterhubungan dari masing-masing objek yang terlibat. *Interface* yang akan dibuat akan langsung memiliki keterhubungan dengan kelas yang mampu melakukan proses optimisasi. Keterhubungan tersebut tidak diperantara oleh kelas Controller seperti pada topologi *Model*, *View*, dan *Controller*.

Perlu diingat pula bahwa proses optimisasi algoritma *ant colony* mampu dikerjakan secara paralel. Untuk pengaplikasian proses optimisasi secara paralel, kelas yang melakukan proses optimisasi akan dipecah menjadi 2 kelas. Satu kelas akan bertugas sebagai kelas pusat yang mengatur pelaksanaan proses optimisasi dan mengatur proses perubahan nilai feromon serta mencari hasil yang paling optimal. Kelas yang satu lagi akan mengimplementasi *interface* "Runnable" agar mampu melakukan proses perhitungan secara serentak. Kelas ini bertugas untuk membentuk solusi dan mencari nilai *makespan* dari solusi tersebut.

Berikut adalah diagram kelas dari perangkat lunak yang dibuat.



Gambar 4.2: Diagram kelas dari perangkat lunak

Keterangan :

- **Kelas TesterGUI**
  Kelas ini merupakan kelas yang membentuk *interface* dan mengaplikasikan fungsi-fungsi aplikasi pada komponen-komponen *interface*. *Interface* yang dibentuk oleh kelas ini berguna untuk proses interaksi dengan pengguna selama perangkat lunak dijalankan. Atribut dan *method* yang dimiliki oleh kelas ini adalah :

  – Atribut

  ∗ problem
  Kasus *hybrid flow shop* yang telah dikonversi dari *file* teks dan ingin dicari hasil optimalnya.

    * aco
      Algoritma *ant colony* yang digunakan untuk pencarian hasil optimal dari kasus *hybrid flow shop* yang disimpan pada atribut problem.
  - *Method*
    * TesterGUI
      Constructor dari kelas ini.
    * initComponents
      Menginisialisasikan komponen-komponen seperti Tombol, *Text Box*, dan *Text Area* pada *interface*.
    * buttInputActionPerformed
      Membuat kotak dialog baru untuk memilih *file* teks berisi kasus *hybrid flow shop* yang ingin dicari hasil optimalnya. *Method* ini akan mengkonversi *file* teks tersebut menjadi kasus *hybrid flow shop* dan mengkonfigurasi algoritma *ant colony* yang akan digunakan untuk pencarian hasil optimal dari kasus tersebut. *Method* ini akan dijalankan saat pengguna menekan tombol "Browse".
    * buttTrainActionPerformed
      Melakukan proses pelatihan dengan menggunakan algoritma *ant colony* sebanyak 1 kali pada kasus *hybrid flow shop* yang dipilih. Hasil dari proses optimisasi akan ditampilkan pada pengguna. *Method* ini akan dijalankan saat pengguna menekan tombol "Train Once".
    * buttResetActionPerformed
      Mengulang kembali proses optimisasi dari kondisi awal. *Method* ini akan dijalankan saat pengguna menekan tombol "Reset".
    * buttTrainFullActionPerformed
      Melakukan proses pelatihan dengan menggunakan algoritma *ant colony* pada kasus *hybrid flow shop* yang dipilih hingga kondisi berhenti dicapai. Kondisi berhenti yang digunakan adalah diberikannya solusi optimal yang sama pada 10 proses pelatihan terakhir. Hasil dari proses optimisasi hingga kondisi tercapainya kondisi berhenti akan ditampilkan pada pengguna. *Method* ini akan dijalankan saat pengguna menekan tombol "Train Full".
    * main
      Method untuk menginisialisasi dan menjalankan perangkat lunak.

- **Kelas Job**
  Kelas ini merepresentasikan sebuah pekerjaan yang akan dikerjakan pada sebuah kasus *hybrid flow shop*. Kelas ini menyimpan informasi-informasi mengenai sebuah pekerjaan pada kasus *hybrid flow shop*. Atribut dan *method* yang dimiliki oleh kelas ini adalah :
  - Atribut
    * jobNumber
      Menyimpan nomor petunjuk identitas sebuah pekerjaan. Atribut ini merupakan nomor pembeda sebuah pekerjaan dari pekerjaan lainnya dalam sebuah kasus *hybrid flow shop*.

* processTime
  Menyimpan detil waktu pengerjaan proses-proses pada sebuah pekerjaan. Detil-detil waktu proses tersebut disimpan dalam bentuk *array* integer. Indeks *array* berfungsi sebagai penunjuk nomor proses dari pekerjaan tersebut.
* jumlahProses
  Menyimpan detil mengenai banyaknya proses dalam sebuah pekerjaan. Nilai atribut ini diberikan pada saat pembentukkan kelas. Nilai dari atribut ini sesuai dengan panjang *array* integer dari atribut processTime.

– *Method*

* Job
  Contructor dari kelas ini. Pembuatan kelas ini dilakukan saat mengkonversi file teks menjadi suatu kasus *hybrid flow shop*. Adapun parameter-parameter yang diperlukan untuk pembentukkan kelas ini adalah :
  · jobNumber : nomor / nama pekerjaan
  · processTime : detil waktu proses pekerjaan, dalam bentuk *array* integer
* toString
  Method ini akan mengembalikan detil waktu-waktu proses pada sebuah pekerjaan. Detil waktu tersebut akan ditampilkan dalam sebuah String yang akan diberikan kepada pengguna perangkat lunak.
* getJobNumber
  Method ini akan mengembalikan nilai atribut jobNumber yang berisi nomor identitas / nama dari pekerjaan ini.
* getProcessTime
  Method ini akan mengembalikan nilai atribut processTime yang berisi detil waktu-waktu proses dari pekerjaan ini. Atribut ini akan diberikan dalam bentuk *array* integer. *Method* ini juga mampu menerima parameter sebuah angka integer sebagai penunjuk nomor proses. Jika *method* ini menerima parameter nomor proses tersebut, maka *method* ini akan mengembalikan sebuah integer lama waktu pengerjaan proses ke-x dari pekerjaan tersebut, di mana x tersebut merupakan parameter nomor pekerjaan yang dimasukkan.
* getJumlahProses
  Method ini akan mengembalikan nilai atribut jumlahProses yang berisi banyaknya proses pada pekerjaan ini.

- **Kelas Problem**
  Kelas ini merepresentasikan sebuah kasus *hybrid flow shop*. Kelas ini menyimpan informasi-informasi mengenai sebuah kasus *hybrid flow shop*. Atribut dan *method* yang dimiliki oleh kelas ini adalah :

  – Atribut

  * jumlahJob
    Menyimpan informasi mengenai banyaknya pekerjaan pada sebuah kasus *hybrid flow shop*.

  * jumlahProses
    Menyimpan informasi mengenai banyaknya mesin / proses pada sebuah kasus *hybrid flow shop*.
  * jobDetail
    Menyimpan informasi mengenai setiap pekerjaan yang terdapat pada kasus *hybrid flow shop*. Informasi disimpan dalam bentuk *array* kelas Job. Indeks *array* berfungsi sebagai penunjuk nomor pekerjaan.
  * jumlahMesin
    Menyimpan informasi mengenai banyaknya perangkat mesin yang mampu mengerjakan masing-masing proses. Informasi disimpan dalam bentuk *array* integer. Indeks *array* berfungsi sebagai penunjuk nomor proses untuk mesin.
  * upperBound
    Menyimpan informasi nilai *upper bound* dari kasus *hybrid flow shop* ini.
  * lowerBound
    Menyimpan informasi nilai *lower bound* dari kasus *hybrid flow shop* ini.

- *Method*
  * Problem
    Constructor dari kelas ini. Pembuatan kelas ini merupakan hasil konversi *file* teks menjadi sebuah kasus *hybrid flow shop*. Parameter-parameter yang dibutuhkan untuk pembentukkan kelas ini adalah :
    · job : banyaknya pekerjaan
    · proses : banyaknya mesin / proses
    · jobDetail : informasi pekerjaan-pekerjaan yang ada, dalam bentuk *array* kelas Job
    · mesin : banyaknya perangkat mesin untuk setiap proses, dalam bentuk *array* integer
    · upperBound : nilai *upper bound*
    · lowerBound : nilai *lower bound*
  * getJumlahJob
    Mengembalikan nilai atribut jumlahJob yang berisi informasi mengenai banyaknya pekerjaan pada kasus *hybrid flow shop*.
  * getJumlahProses
    Mengembalikan nilai atribut jumlahProses yang berisi informasi mengenai banyaknya mesin / proses pada kasus *hybrid flow shop*.
  * getJobDetail
    Mengembalikan detil informasi mengenai pekerjaan yang ada pada kasus *hybrid flow shop*. *Method* ini akan mengembalikan informasi detil pekerjaan dalam bentuk *array* kelas Job. *Method* ini juga mampu menerima sebuah parameter number, di mana number merupakan penunjuk nomor / nama pekerjaan. Jika *method* ini menerima parameter tersebut, maka *method* ini akan mengembalikan informasi pekerjaan dengan nomor / nama pekerjaan tersebut. Informasi tersebut diberikan dalam bentuk

sebuah objek dari kelas Job.

* getJumlahMesin
  Mengembalikan detil informasi mengenai banyaknya perangkat mesin yang ada pada kasus *hybrid flow shop*. *Method* ini akan mengembalikan informasi detil banyaknya perangkat mesin dalam bentuk *array* integer. *Method* ini juga mampu menerima sebuah parameter index, di mana index merupakan penunjuk nomor proses dari mesin tersebut. Jika *method* ini menerima parameter tersebut, maka *method* ini akan mengembalikan informasi banyaknya perangkat mesin pada nomor proses tersebut. Informasi tersebut diberikan dalam bentuk sebuah angka integer.
* getUpperBound
  Mengembalikan nilai *upper bound*pada kasus *hybrid flow shop* yang disimpan pada atribut upperBound.
* getLowerBound
  Mengembalikan nilai *lower bound*pada kasus *hybrid flow shop* yang disimpan pada atribut lowerBound.

- **Kelas Mesin**
  Kelas ini merepresentasikan sebuah mesin pada sistem penjadwalan *hybrid flow shop*. Kelas ini menjabarkan kondisi mesin yang sedang mengerjakan / menunggu sebuah pekerjaan. Atribut dan *method* yang dimiliki oleh kelas ini adalah :
  - Atribut
    * job
      Menyimpan informasi mengenai pekerjaan yang sedang dikerjakan / disimpan pada mesin ini. Informasi disimpan dalam bentuk sebuah objek dari kelas Job.
    * processNumber
      Menyimpan informasi mengenai nomor dari proses yang akan dikerjakan oleh mesin.
    * processTime
      Menyimpan informasi mengenai lama waktu proses yang telah dilalui dalam pengerjaan proses dari sebuah pekerjaan. Jika pekerjaan telah selesai dikerjakan / tidak ada pekerjaan pada mesin, maka atribut ini akan bernilai 0.
    * finished
      Menyimpan informasi mengenai status mesin. Atribut ini mengidentifikasikan sebuah mesin telah menyelesaikan pekerjaannya.
    * waiting
      Menyimpan informasi mengenai status mesin. Atribut ini mengidentifikasikan sebuah mesin sedang kosong dan menunggu pekerjaan baru untuk dikerjakan.
  - *Method*
    * Mesin
      Constructor dari kelas ini. Pembuatan kelas ini dilakukan pada saat konfigurasi mesin-mesin pada sistem *hybrid flow shop* agar sesuai dengan kasus *hybrid flow shop* yang ingin dikerjakan / dioptimisasi.

Parameter yang diperlukan dalam pembentukkan kelas ini adalah :

· processNumber : nomor proses yang akan dikerjakan oleh mesin

Pada saat pembentukkan objek dari kelas ini, mesin akan dianggap sebagai sebuah mesin baru tanpa ada pekerjaan apapun yang dikerjakan di dalamnya. Oleh karena itu, nilai dari atribut job akan bernilai null. Nilai dari atribut processTime akan bernilai 0. Nilai dari atribut waiting dan finished akan bernilai *true*.

* processing

  Menambah 1 nilai pada atribut processTime. Jika nilai dari atribut processTime telah sama dengan waktu pengerjaan dari proses pekerjaan yang ditugaskan pada mesin, maka *method* ini akan mengubah status mesin menjadi sudah selesai mengerjakan tugasnya. Status mesin yang sudah selesai mengerjakan tugasnya ditandai dengan atribut finished yang bernilai *true*. Jika mesin telah menyelesaikan tugasnya, maka nilai dari atribut processTime juga akan diatur ulang menjadi bernilai 0. *Method* ini hanya akan berjalan pada mesin dengan nilai atribut finished *false*.

* passJob

  Mengoper pekerjaan yang telah diselesaikan oleh mesin. *Method* ini akan mengembalikan pekerjaan yang telah diselesaikan oleh mesin dalam bentuk sebuah objek dari kelas Job. Pada saat yang sama, *method* ini juga akan mengubah status mesin menjadi kosong / menunggu pekerjaan baru. Mesin yang sedang menunggu pekerjaan baru ditandai dengan berubahnya nilai dari atribut waiting menjadi *true*. *Method* ini juga akan mengatur ulang nilai dari atribut job menjadi *null*. *Method* ini hanya bisa dijalankan pada mesin dengan nilai atribut finished *true* dan nilai atribut waiting *false*. Nilai atribut tersebut dapat diartikan sebagai mesin yang sudah menyelesaikan pekerjaannya tetapi masih menyimpan pekerjaannya.

* receiveJob

  Menerima pekerjaan baru untuk dikerjakan mesin. *Method* ini menerima sebuah parameter berupa objek dari kelas Job, di mana parameter tersebut merupakan pekerjaan baru yang harus dikerjakan oleh mesin. Parameter tersebut akan disimpan sebagai atribut job dari mesin. *Method* ini akan mengubah status mesin menjadi sedang mengerjakan sebuah pekerjaan. Mesin yang sedang mengerjakan sebuah pekerjaan ditandai dengan atribut finished dan waiting yang bernilai *false*. *Method* ini hanya dapat dijalankan pada mesin yang sedang menunggu pekerjaan baru(atribut waiting bernilai *true*).

* isWaiting

  Mengembalikan informasi mengenai apakah mesin sedang kosong atau tidak. Informasi didapat dari atribut waiting. Jika mesin sedang kosong dan sedang menunggu diberikannya pekerjaan baru, maka *method* ini akan mengembalikan nilai boolean *true*. Jika mesin sedang mengerjakan / menyimpan sebuah pekerjaan, maka *method* akan mengembalikan nilai boolean *false*.

* isFinished

  Mengembalikan informasi mengenai apakah mesin sudah / belum menyelesaikan pekerjaannya. Informasi tersebut didapatkan dari atribut finished. Jika mesin telah

menyelesaikan proses dari pekerjaan di dalamnya, maka *method* ini akan mengembalikan nilai boolean *true*. Jika mesin belum menyelesaikan pekerjaannya, maka *method* akan mengembalikan nilai boolean *false*.

* getJob
  Mengembalikan informasi mengenai pekerjaan yang sedang dikerjakan / disimpan oleh mesin. Informasi diberikan dalam bentuk sebuah objek dari kelas Job. Jika mesin sedang tidak mengerjakan / menyimpan sebuah pekerjaan, maka *method* ini akan mengembalikan nilai *null*.

- **Kelas HybridFlowShop**
  Kelas ini merepresentasikan sebuah sistem penjadwalan *hybrid flow shop*. Kelas ini bertugas mengatur kapan sebuah mesin harus menerima dan mengoper sebuah pekerjaan. Kelas ini juga bertugas untuk menghitung lama waktu yang dibutuhkan untuk pengerjaan sekumpulan pekerjaan dengan urutan tertentu. Atribut dan *method* yang dimiliki oleh kelas ini adalah :

  – Atribut

  * mesin
    Informasi mengenai rangkaian mesin yang ada pada sistem *hybrid flow shop*. Informasi disimpan dalam bentuk *array* 2 dimensi dari kelas Mesin. Indeks pertama dari *array* menunjukkan nomor proses yang dikerjakan oleh mesin dan indeks *array* kedua menunjukkan nomor mesin. Sebagai contoh indeks *array* [3][2] dari atribut ini menunjuk pada mesin kedua yang mampu mengerjakan proses 3.

  – *Method*

  * HybridFlowShop
    Constructor dari kelas ini. Pembentukkan kelas ini dilakukan pada saat konfigurasi sistem *hybrid flow shop* agar sesuai dengan kasus *hybrid flow shop* yang ingin dikerjakan / dioptimisasi. Parameter yang dibutuhkan untuk pembentukkan kelas ini adalah :
    · problem : kasus *hybrid flow shop* yang ingin dikerjakan / dioptimisasi.
    Pada saat pembentukkan sebuah sistem *hybrid flow shop*, konfigurasi rangkaian mesin pada sistem *hybrid flow shop* akan dilakukan. Banyaknya perangkat mesin untuk setiap proses akan diatur berdasarkan nilai-nilai yang ada pada parameter kasus *hybrid flow shop*. Nomor proses yang harus dikerjakan masing-masing mesin juga akan diatur secara otomatis.

  * countMakespan
    Method ini akan menghitung nilai *makespan* yang dihasilkan oleh parameter *array* kelas Job yang dimasukkan. Parameter tersebut dianggap sebagai sekumpulan pekerjaan yang akan dimasukkan / dikerjakan oleh sistem *hybrid flow shop* ini. Pekerjaan-pekerjaan tersebut akan dikerjakan secara terurut berdasarkan indeks *array* kelas Job tersebut. *Method* ini akan menghitung nilai *makespan* yang dihasilkan dari suatu solusi / urutan pengerjaan berdasarkan berapa kali *method* processAll dijalankan.

Method ini akan mengatur kapan proses pengoperan dan penerimaan pekerjaan antar mesin terjadi. *Method* ini juga akan mengatur mesin mana yang berhak melakukan pengoperan / menerima sebuah pekerjaan. Proses pengoperan dimulai dengan pengecekan mesin-mesin yang mengerjakan proses terakhir dari suatu pekerjaan. Jika mesin-mesin tersebut telah menyelesaikan pekerjaannya, maka mesin tersebut akan langsung mengoper pekerjaannya tanpa harus ada mesin lain yang menerima pekerjaan yang dioper. Pekerjaan yang telah dioper oleh mesin terakhir ini dianggap telah selesai dikerjakan.

Untuk proses pengoperan mesin-mesin lainnya, pengecekan akan dilakukan secara terurut dimulai dari mesin-mesin dengan nomor proses terbesar. Untuk setiap mesin-mesin pengerja proses yang sama, jika mesin tersebut telah menyelesaikan pekerjaannya, mesin tersebut akan diingat oleh sistem *hybrid flow shop* agar dapat dengan segera dioper pekerjaannya. Jika terdapat mesin yang mampu mengoper pekerjaannya, maka pengecekan mesin untuk proses selanjutnya akan dilakukan.

Jika terdapat mesin untuk proses selanjutnya yang sedang menunggu pekerjaan baru, maka pekerjaan dapat langsung dioper pada mesin tersebut. Jika tidak ada mesin untuk proses selanjutnya yang sedang menunggu, maka proses pengoperan belum akan dilakukan. Prioritas mengenai mesin yang berhak melakukan pengoperan terlebih dahulu ditentukan berdasarkan nomor / nama dari pekerjaan yang akan dioper oleh mesin. Jika suatu pekerjaan dikerjakan terlebih dahulu, maka pekerjaan tersebut akan memiliki prioritas lebih besar untuk dioper terlebih dahulu.

Setelah proses pengecekan dilakukan pada setiap mesin, pekerjaan baru akan dimasukkan pada mesin-mesin dengan nomor proses 0 (mesin yang mengerjakan proses pertama). Jika terdapat mesin dengan nomor proses 0 yang sedang menunggu pekerjaan baru, maka sebuah pekerjaan baru akan diberikan pada mesin tersebut. Pekerjaan baru tersebut akan diberikan secara terurut berdasarkan indeks *array* dari parameter *array* kelas Job yang dimasukkan.

Setelah proses-proses pengoperan tersebut dilakukan, *method* processAll akan dijalankan. *Method* processAll tersebut akan melakukan pemrosesan pada pekerjaan yang ada di setiap mesin. *Method* countMakespan ini akan terus berjalan selama masih ada objek dari parameter *array* kelas Job yang belum dimasukkan ke dalam sistem *hybrid flow shop*. *Method* ini juga akan terus berjalan selama *method* finishCheck mengembalikan nilai *true*. Dari tata cara tersebut, dibentuklah *pseudocode* berikut ini.

```
1  while(proses belum selesai)
2      untuk setiap mesin pada tahap proses terakhir
3          if(mesin m selesai) -> keluarkan pekerjaan
4      untuk setiap mesin lainnya pada setiap tahap proses
5          mesinProsesX <- ingat-ingat setiap mesin yang mampu mengoper pekerjaan pada satu
               tahap proses
6          untuk setiap mesin pada tahap proses berikutnya
7              if(mesinX+1 dapat menerima pekerjaan) ->
8                  periksa pekerjaan yang disimpan di kumpulan mesinProsesX sesuai dengan
                       urutan pengerjaan pekerjaan yang dipilih
9                  if(mesinX dari kumpulan mesinProsesX mengerjakan pekerjaan yang sedang
                         diperiksa) -> oper pekerjaan dari kumpulan mesinX ke mesinX+1
10     untuk setiap pekerjaan baru yang belum mulai dikerjakan (jika ada)
11         untuk setiap mesin pada tahap proses pertama
12             if(mesin1 dapat menerima pekerjaan) -> masukkan pekerjaan ke mesin1 untuk
                   dikerjakan
```

```
13      jalankanProses()
14  output : berapa kali jalankanProses() dilakukan
```

* processAll
  Menjalankan *method* processing pada setiap mesin yang dimiliki oleh *hybrid flow shop*. *Method* ini akan menambahkan waktu pemrosesan (atribut processTime kelas Mesin) setiap mesin sebanyak 1 satuan waktu. Jika mesin telah menyelesaikan pekerjaannya, maka mesin tersebut akan dilewat dan tidak ditambahkan waktu pemrosesannya. *Method* ini berguna untuk *method* countMakespan. Dalam menghitung nilai *makespan* yang dihasilkan oleh suatu urutan pengerjaan / solusi, dijalankannya *method* ini akan menambahkan nilai makespan yang dihasilkan *method* countMakespan sebanyak 1.
* finishCheck
  Method ini berguna sebagai penanda berhentinya *method* countMakespan. *Method* ini akan memeriksa kondisi setiap mesin pada sistem *hybrid flow shop*. Jika setiap mesin sedang dalam kondisi menunggu pekerjaan baru (atribut waiting dari kelas Mesin bernilai *true*), maka seluruh pekerjaan yang diberikan dari *method* countMakespan dapat dianggap telah selesai dikerjakan. Jika seluruh pekerjaan telah selesai dikerjakan, maka *method* ini akan mengembalikan nilai *true*. Sebaliknya jika terdapat salah satu mesin yang tidak dalam kondisi menunggu pekerjaan baru, maka *method* ini akan mengembalikan nilai *false*.

- **Kelas Ant**
  Kelas ini merepresentasikan seekor semut pada algoritma *ant colony*. Kelas ini akan menyimpan sebuah solusi dari hasil proses pembentukan solusi milik algoritma *ant colony*. Nilai-nilai yang disimpan dari seekor semut akan disesuaikan dengan permasalahan yang ingin diselesaikan. Dalam kasus ini, nilai yang akan disimpan oleh semut sebagai solusi akan disesuaikan dengan permasalahan *hybrid flow shop*. Atribut dan *method* dari kelas ini adalah :

  – Atribut

  * jobSequence
    Menyimpan informasi mengenai salah satu solusi / urutan pengerjaan dari kasus *hybrid flow shop*. Informasi tersebut disimpan dalam bentuk *array* kelas Job. Indeks *array* menunjukkan nomor urutan pengerjaan.
  * makespan
    Menyimpan informasi mengenai nilai *makespan* yang dihasilkan dari urutan pengerjaan / solusi yang disimpan pada atribut jobSequence.

  – *Method*

  * Ant
    Constructor dari kelas ini. Pembentukkan kelas ini dilakukan setiap dijalankannya fase pelatihan dari algoritma *ant colony*. Objek dari kelas ini mampu dibentuk langsung dengan parameter urutan pengerjaan dan nilai *makespan* yang ingin disimpan. Objek dari kelas ini juga mampu dibentuk tanpa dengan memberikan parameter

apapun. Jika objek dibentuk tanpa dengan memasukkan parameter, maka atribut jobSequence akan bernilai *null* dan atribut makespan akan bernilai -1.

* toString
  Mengembalikan informasi mengenai solusi yang disimpan oleh semut ini secara lengkap. Informasi akan diberikan dalam bentuk 2 baris string. Baris pertama menunjukkan nilai *makespan* yang dimiliki dan baris kesua menunjukkan solusi / urutan pengerjaan yang menghasilkan nilai *makespan* tersebut.
* jobPathToString
  Mengkonversi *array* kelas Job menjadi sebuah string urutan pengerjaan. Pengerjaan sebuah pekerjaan ditandai oleh atribut jobNumber dari kelas Job. Perlu diketahui pula bahwa perangkat lunak ini membentuk atribut jobNumber pada kelas Job secara otomatis dimulai dari nilai 0. Untuk mempermudah penggunaan perangkat lunak, *method* ini akan mengkonversi nilai jobNumber dalam string yang akan diberikan. Untuk setiap jobNumber pada string tersebut akan ditambahkan 1. Hal ini dilakukan agar pengerjaan pekerjaan pertama didefinisikan dengan jobNumber 1 bukan dengan jobNumber 0.
* getMakespan
  Mengembalikan informasi nilai *makespan* yang dihasilkan oleh urutan pengerjaan / solusi pada atribut jobSequence.
* getJobSequence
  Mengembalikan informasi urutan pengerjaan yang disimpan oleh semut. Informasi diberikan dalam bentuk *array* kelas Job, dengan indeks *array* sebagai nomor urutan pengerjaannya. *Method* ini juga mampu menerima sebuah parameter index. Jika *method* dijalankan dengan memberikan parameter tersebut, maka *method* akan mengembalikan informasi mengenai pekerjaan yang dikerjakan di urutan ke index. Informasi tersebut diberikan dalam bentuk seuah objek dari kelas Job.
* setMakespan
  Mengubah nilai dari atribut makespan sesuai dengan nilai parameter yang dimasukkan.
* setJobSequence
  Mengubah nilai dari atribut jobSequence sesuai dengan parameter *array* kelas Job yang dimasukkan.

- **Kelas PheromoneDatabase**
  Kelas ini merepresentasikan tempat penyimpanan nilai feromon untuk panduan proses pembentukan solusi secara acak milik algoritma *ant colony*. Cara penyimpanan nilai feromon disesuaikan dengan permasalahan yang ingin diselesaikan. Dalam kasus ini, nilai feromon yang disimpan akan disesuaikan dengan permasalahan *hybrid flow shop* dan tata cara yang digunakan dalam proses pembentukan solusi secara acak milik algoritma *ant colony*. Atribut dan *method* yang dimiliki oleh kelas ini adalah :

  – Atribut

    * dataFeromon

Menyimpan informasi mengenai nilai-nilai feromon yang disimpan. Informasi tersebut disesuaikan dengan tata cara pembentukkan solusi acak dari algoritma *ant colony*. Informasi disimpan dalam bentuk *array* 2 dimensi. Indeks pertama dari *array* menunjukkan nomor pekerjaan sebelumnya dan indeks kedua menunjukkan nomor pekerjaan selanjuntya.

* startValue
  Menyimpan informasi mengenai nilai awal yang diberikan pada masing-masing indeks atribut dataFeromon. Nilai awal tersebut hanya akan diberikan pada saat pembentukkan objek dari kelas ini.

– *Method*

* PheromoneDatabase
  Constructor dari kelas ini. Pembentukkan kelas ini dilakukan saat pembentukkan kelas AntColonyAlgorithm untuk disesuaikan dengan kasus *hybrid flow shop* yang ingin dicari hasil optimalnya. Parameter yang diperlukan untuk pembentukkan kelas ini adalah :
  · jumlahPekerjaan : banyaknya pekerjaan pada kasus *hybrid flow shop* yang ingin dioptimisasi.
  Atribut dataFeromon dari kelas ini akan dibentuk berdasarkan nilai parameter jumlahPekerjaan yang dimasukkan. *Array* yang disimpan pada atribut dataFeromon akan berukuran n x n, di mana n merupakan nilai parameter jumlahPekerjaan yang dimasukkan. Masing-masing indeks dari atribut dataFeromon akan diberikan nilai awal sesuai dengan nilai dari atribut startValue.
* getPheromone
  Mengembalikan nilai pada suatu indeks dari atribut dataFeromon. *Method* ini menerima 2 parameter: now dan target. *Method* ini akan mengembalikan nilai dari indeks *array* [now][target] dari atribut dataFeromon.
* addPheromone
  Menambahkan nilai pada suatu indeks dari atribut dataFeromon. *Method* ini menerima 3 parameter: now, target, dan value. *Method* ini akan menambahkan nilai pada indeks *array* [now][target] dari atribut dataFeromon sejumlah nilai value.
* evaporate
  Melakukan proses evaporasi feromon pada nilai-nilai feromon yang disimpan. *Method* ini akan mengurangi 30% nilai dari masing-masing indeks pada atribut dataFeromon. Nilai hasil evaporasi tidak akan lebih rendah dari 1.
* toString
  Mengembalikan informasi mengenai nilai-nilai feromon yang disimpan pada atribut dataFeromon dalam bentuk string matirks 2 dimensi.

- **Kelas AntColonyAlgorithm**
  Kelas ini akan melakukan optimisasi dengan menggunakan algoritma *ant colony*. Kelas ini akan mencari urutan pengerjaan yang menghasilkan *makespan* yang optimal dalam suatu kasus *hybrid flow shop*. Dalam mencari hasil optimal tersebut, kelas ini akan dibantu oleh

kelas PheromoneDatabase. Atribut dan *method* yang dimiliki oleh kelas ini adalah :

- Atribut
  - problem
    Menyimpan kasus *hybrid flow shop* yang ingin dicari solusi optimalnya.
  - pheromone
    Menyimpan kelas pheromoneDatabase yang akan membantu dalam pembentukkan solusi acak algoritma *ant colony*.
  - optimalSolution
    Menyimpan solusi optimal dalam bentuk objek dari kelas Ant. Solusi optimal yang disimpan merupakan solusi terbaik dari fase-fase pelatihan yang telah dilakukan.
  - trainingCount
    Menyimpan informasi mengenai banyaknya fase pelatihan yang telah dijalankan.
  - pheromoneGap
    Menyimpan informasi mengenai sebuah nilai *makespan* yang dianggap sebagai nilai tengah dari nilai-nilai *makespan* lainnya yang mungkin dihasilkan. Nilai ini berguna dalam menentukan jumlah feromon yang akan ditambahkan dari sebuah solusi acak yang telah dibentuk.
  - antColonySub
    Menyimpan informasi mengenai kelompok-kelompok semut yang ada dalam sistem algoritma *ant colony* ini. Masing-masing kelompok akan dijalankan secara bersamaan pada saat proses pencarian solusi-solusi lokal dari algoritma *ant colony*.
  - antNumber
    Menyimpan informasi mengenai banyaknya semut pada setiap kelompok semut.
  - antThread
    Menyimpan informasi thread yang dapat dijadikan penanda selesainya proses pencarian solusi / pelatihan dari salah satu kelompok semut.
- *Method*
  - AntColonyAlgorithm
    Constructor dari kelas ini. Pembentukkan kelas ini dilakukan setelah selesai mengkonversi *file* teks menjadi suatu kasus *hybrid flow shop*. Parameter untuk pembentukkan kelas ini adalah :
    - problem : kasus *hybrid flow shop* yang ingin dicari hasil optimalnya.

    Pembentukkan kelas ini akan mengkonfigurasi atribut pheromone dan hfs agar sesuai dengan kasus *hybrid flow shop* yang ingin dicari hasil optimalnya. Nilai dari atribut antNumber akan dibentuk berdasarkan atribut jumlahPekerjaan pada parameter kelas Problem yang dimasukkan. Nilai dari atribut pheromoneGap akan disamakan dengan nilai *makespan* yang dihasilkan dari salah satu solusi kasus *hybrid flow shop*. Solusi / urutan pengerjaan yang akan dipakai nilai *makespan*-nya adalah solusi yang urutan pengerjaannya terurut berdasarkan nilai atribut jobNumber.
  - trainOnce
    Method ini akan menjalankan proses pelatihan pada algoritma *ant colony* sebanyak

satu kali. *Method* ini akan menjalankan *method* spreadAnt dengan atribut antNumber sebagai parameter. *Method* ini akan mengembalikan hasil dari proses optimisasi tersebut dalam bentuk 3 baris string. Baris pertama menginformasikan banyaknya fase pelatihan yang telah dilakukan, baris kedua menunjukkan nilai *makespan* optimal yang diketahui hingga proses pelatihan ini, dan baris ketiga menunjukkan urutan pengerjaan yang menghasilkan nilai *makespan* tersebut.

* trainFull
  Method ini akan menjalankan proses pelatihan pada algoritma *ant colony* hingga suatu kondisi berhenti tercapai. *Method* ini akan menjalankan *method* spreadAnt hingga suatu kondisi berhenti tercapai. *Method* spreadAnt dijalankan dengan atribut antNumber sebagai parameter. Kondisi berhenti yang digunakan adalah diberikannya hasil optimisasi yang sama pada 10 fase pelatihan terakhir. *Method* ini akan mengembalikan hasil dari proses optimisasi tersebut dalam bentuk sebuah string. String tersebut akan berisi informasi mengenai hasil optimisasi yang diberikan pada setiap fase pelatihan hingga kondisi berhenti tersebut tercapai. Masing-masing hasil optimisasi pada tiap fase pelatihan dijabarkan dalam 3 baris string. Baris pertama menginformasikan nomor fase pelatihan, baris kedua menunjukkan nilai *makespan* optimal yang diketahui hingga proses pelatihan tersebut, dan baris ketiga menunjukkan urutan pengerjaan yang menghasilkan nilai *makespan* tersebut.

* spreadAnt
  Method ini akan melakukan fase pelatihan pada algoritma *ant colony*. *Method* ini akan menginstruksikan masing-masing objek kelas AntColonySub pada atribut untuk mulai mencari solusi-solusi lokal. Secara terurut hal-hal yang dilakukan oleh *method* ini adalah :
  - Menginstruksikan setiap kelompok semut (atribut antColonySub) untuk memulai mencari solusi-solusi dari permasalahan pada atribut problem.
  - Masing-masing kelompok semut akan :
    - Membentuk solusi-solusi - Mencari nilai *makespan* dari solusi-solusi tersebut - Menyimpan solusi terbaik dan solusi-solusi lainnya yang dibentuk.
  - Menunggu setiap kelompok semut untuk menyelesaikan proses pencarian solusinya.
  - Melakukan *update* nilai feromon berdasarkan solusi-solusi yang dibentuk pada setiap kelompok semut.
  - Membandingkan solusi terbaik saat ini dengan solusi terbaik pada setiap kelompok semut.
  - Jika terdapat solusi yang lebih baik, maka solusi tersebut akan disimpan.
  - Menginformasikan solusi terbaik pada fase pelatihan ini pada setiap kelompok semut
  - Melakukan evaporasi nilai feromon.
  - Menambah jumlah hitungan fase pelatihan.
  - Memberikan solusi optimal pada fase pelatihan ini sebagai data keluaran.

Berdasarkan urutan tersebut, berikut ini adalah *pseudocode* untuk *method* ini.

```
1    untuk setiap kelompok semut
2        instruksikan kelompok semut untuk mulai menyebarkan semut
3    untuk setiap kelompok semut
4        tunggu hingga setiap semut selesai menyebarkan semut
5    ambil solusi terbaik dari setiap kelompok semut dan bandingkan
6    informasikan solusi terbaik pada setiap kelompok semut
7    penambahan nilai feromon
8    evaporasi nilai feromon
```

* updatePheromone

  Method ini akan melakukan proses penambahan nilai feromon berdasarkan solusi yang diberikan sebagai parameter. *Method* ini menerima parameter dari kelas Ant yang dapat dianggap sebagai suatu solusi untuk kasus *hybrid flow shop*. Indeks-indeks matriks feromon yang akan ditambahkan ditentukan berdasarkan atribut jobSequence milik parameter objek Ant.

  Jumlah nilai feromon yang akan ditambahkan pada setiap target indeks matriks ditentukan berdasarkan nilai *makespan* yang dihasilkan oleh solusi tersebut. Nilai feromon yang akan ditambahkan sejumlah selisih dari nilai *makespan* dari solusi ini dengan *makespan* dari solusi yang dianggap sebagai nilai tengah. Nilai *makespan* yang dianggap sebagai nilai tengah disimpan pada atribut pheromoneGap.

  Method ini juga mampu menerima *array* kelas Ant sebagai parameter. Jika *method* ini menerima *array* kelas Ant sebagai parameter, maka *method* ini akan melakukan penambahan nilai feromon untuk setiap solusi yang dimiliki oleh setiap objek Ant pada *array* tersebut.

* getTrainingCount

  Mengembalikan nilai dari atribut trainingCount yang merupakan banyaknya proses pelatihan yang telah dilakukan.

* getOptimalSolution

  Mengembalikan nilai dari atribut optimalSolution yang merupakan informasi solusi optimal dari kasus *hybrid flow shop* pada atribut problem. Solusi optimal tersebut merupakan solusi terbaik dari fase-fase pelatihan yang telah dilakukan. Informasi solusi optimal tersebut diberikan dalam bentuk sebuah objek dari kelas Ant.

* getAntNumber

  Mengembalikan nilai dari atribut antNumber yang merupakan informasi mengenai banyaknya semut yang dibentuk pada setiap fase pelatihan.

* getPheromoneData

  Mengembalikan informasi dari nilai-nilai feromon yang digunakan untuk pembentukkan solusi acak dari algoritma *ant colony*. Informasi tersebut diolah dan dikembalikan dalam bentuk sebuah string matriks 2 dimensi.

- **Kelas AntColonySub**

  Kelas ini merepresentasikan sebuah kelompok semut pada algoritma *ant colony*. Kelas ini mengimplementasikan kelas *interface* Runnable agar mampu dijalankan secara bersamaan. Kelas ini akan dijalankan pada saat proses pelatihan algoritma *ant colony* dilakukan. Kelas ini akan mencari solusi-solusi lokal berdasarkan nilai-nilai feromon yang ada pada algoritma *ant colony* dan mencari nilai *makespan* yang dihasilkan berdasarkan solusi tersebut. Atribut

dan *method* yang dimiliki oleh kelas ini adalah :

- Atribut
  - problem
    Menyimpan kasus *hybrid flow shop* yang ingin dicari solusi optimalnya.
  - pheromone
    Menyimpan kelas pheromoneDatabase yang dioper dari kelas AntColonyAlgorithm yang akan membantu dalam pembentukkan solusi acak algoritma *ant colony*.
  - hfs
    Sistem *hybrid flow shop* yang dibentuk berdasarkan atribut problem dan digunakan untuk menghitung nilai *makespan* yang dihasilkan oleh suatu solusi.
  - antOpt
    Menyimpan solusi optimal dari kelompok semut ini dalam bentuk objek dari kelas Ant.
  - antParty
    Menyimpan solusi-solusi lokal yang didapatkan oleh kelompok semut ini pada fase pelatihan / pencarian solusi terakhir.
  - antNumber
    Menyimpan informasi mengenai banyaknya semut (yang akan disebar saat proses pelatihan) pada kelompok semut ini.
- *Method*
  - run
    Method ini merupakan *method* turunan dari interface Runnable. *Method* ini akan menjalankan *method* spreadAnt dengan atribut antNumber sebagai parameter yang dimasukkan. Setelah proses pencarian solusi selesai, method ini akan diberhentikan hingga proses pencarian solusi berikutnya dijalankan.
  - spreadAnt
    Method ini akan melakukan proses pelatihan / pencarian solusi berdasarkan nilai feromon pada atribut pheromone. *Method* ini akan membentuk solusi-solusi sebanyak nilai dari atribut integer yang dimasukkan. Solusi-solusi tersebut akan dibentuk dengan *method* constructPath. Setiap solusi yang dibentuk akan dicari nilai *makespan*-nya dengan menggunakan kelas HybridFlowShop yang disimpan pada atribut dan dibandingkan dengan solusi yang paling optimal pada saat ini. Solusi yang lebih baik akan disimpan pada atribut antOpt. *Pseudocode* dari *method* ini adalah sebagai berikut.

```
1    untuk semut 1 sampai semut n
2        urutanAcak <- bentuk urutan pekerjaan secara acak
3        makespan <- hitung nilai makespan yang dihasilkan dari urutan pekerjaan tersebut
4        if(nilai makespan lebih kecil dari nilai makespan optimal) -> urutanOptimal <-
             urutanAcak
```

  - constructPath
    Method ini akan membentuk salah satu urutan pengerjaan / solusi dari kasus *hybrid flow shop* yang ada pada atribut problem. Urutan pengerjaan / solusi tersebut di-

bentuk secara acak berdasarkan nilai-nilai feromon yang disimpan pada atribut pheromone. *Method* ini akan mengembalikan urutan pengerjaan / solusi dalam bentuk *array* kelas Job dengan indeks *array* sebagai nomor urutan pengerjaannya.
Proses pembentukkan jalur dimulai dengan membuat duplikat *array* pekerjaan yang ada pada permasalahan *hybrid flow shop*. Proses selanjutnya akan menjalankan *method* pickRandomJob dengan memasukkan duplikat *array* pekerjaan tersebut dan angka -1. Angka -1 tersebut akan menunjukkan bahwa belum ada pekerjaan yang dipilih untuk dimasukkan ke dalam urutan pengerjaan.
Method pickRandomJob akan memilih sebuah pekerjaan dari duplikat *array* pekerjaan secara acak berdasarkan nilai feromon yang disimpan. Pekerjaan yang dipilih dari oleh *method* pickRandomJob akan dimasukkan ke dalam urutan pengerjaan yang akan dihasilkan. Nomor / nama dari pekerjaan yang dipilih akan disimpan sebagai nomor pekerjaan terakhir dan pekerjaan dengan nomor / nama tersebut akan disingkirkan dari duplikat *array* pekerjaan dengan menggunakan *method* removeJob. Pemilihan urutan pekerjaan selanjutnya tetap dilakukan dengan *method* pickRandomJob. Parameter yang dimasukkan adalah *array* duplikat hasil dari *method* removeJob dan nomor pekerjaan terakhir. Hasil *method* pickRandomJob ditambahkan pada hasil urutan pengerjaan. Nomor pekerjaan terakhir diubah sesuai dengan nomor pekerjaan hasil pickRandomJob dan pekerjaan dengan nomor tersebut disisihkan dari *array* duplikat. Proses ini dilakukan hingga seluruh pekerjaan pada *array* duplikat dikeluarkan. Dari tata cara tersebut, dibentuklah *pseudocode* berikut :

```
1  pekerjaanInput  <- pekerjaan-pekerjaan yang ada
2  pekerjaanOutput <- -
3  selama masih ada pekerjaan pada pekerjaanInput
4      pekerjaanX      <- pilih satu pekerjaan secara acak dari pekerjaanInput
5      pekerjaanOutput <- tambahkan pekerjaanX pada hasil urutan pekerjaan yang akan
           dipilih
6      keluarkan pekerjaanX dari pekerjaanInput
7  output : pekerjaanOutput
```

* pickRandomJob

Method ini akan memilih sebuah Job secara acak dari parameter *array* kelas Job yang dimasukkan. Pemilihan Job secara acak tersebut dipengaruhi oleh nilai feromon yang disimpan pada atribut pheromone. *Method* ini menerima 2 buah parameter : *array* kelas Job dan nomor penunjuk pekerjaan terakhir. Indeks matriks feromon yang mempengaruhi proses pemilihan pekerjaan ditentukan oleh nomor / nama pekerjaan yang terdapat pada parameter *array* dan nomor pekerjaan terakhir yang dimasukkan.
Sebagai contoh, jika terdapat pekerjaan 2 pada parameter *array* kelas Job dan nomor pekerjaan terakhir adalah 3, maka indeks matriks [3][2] pada feromon akan mempengaruhi kemungkinan terpilihnya pekerjaan 2 di antara pekerjaan-pekerjaan pada parameter *array* kelas Job tersebut. Jika parameter nomor pekerjaan terakhir memiliki nilai -1, maka proses pemilihan akan menganggap belum ada nomor penunjuk pekerjaan terakhir (belum ada pekerjaan lain yang telah dikerjakan). Jika belum ada nomor penunjuk pekerjaan terakhir, maka proses pemilihan akan melibatkan jumlah dari nilai indeks matriks untuk masing-masing pekerjaan pada *array* kelas Job.

Proses pemilihan pekerjaan dimulai dengan menjumlahkan nilai feromon dari indeks-indeks matriks yang mampu mempengaruhi pemilihan sebuah pekerjaan. Sebuah angka akan dibentuk secara acak di antara nilai 0 hingga jumlah nilai feromon tersebut. Proses selanjutnya akan melakukan proses pejumlahan nilai feromon dari indeks-indeks matriks yang mampu mempengaruhi pemilihan sebuah pekerjaan secara satu per satu. Jika penambahan nilai feromon dari suatu indeks matriks mengakibatkan nilai penjumlahan dari proses ini melebihi nilai angka acak dari proses sebelumnya, maka pekerjaan yang ditunjuk oleh indeks matriks tersebut merupakan pekerjaan yang terpilih.

Pekerjaan tersebut kemudian akan diberikan sebagai *return value* dari *method* ini. Pekerjaan tersebut akan diberikan dalam bentuk sebuah objek dari kelas Job. Berikut adalah *pseudocode* yang dapat digunakan untuk pembentukkan *method* ini :

```
1    feromonMax <- 0
2    untuk setiap pekerjaan pada indeks x
3        tambahkan nilai feromon dari pekerjaan tersebut pada feromonMax
4    feromonRandom <- random 1 nilai antara nilai 0 sampai feromonMax
5    feromonCount <- 0
6    untuk setiap pekerjaan pada indeks x
7        tambahkan nilai feromon dari pekerjaan tersebut pada feromonCount
8        if(nilai feromonCount lebih besar atau sama dengan nilai feromonRandom) -> hasil
             <- pekerjaan pada indeks x
9    output : hasil
```

* removeJobFrom

  Method ini menerima parameter sebuah *array* kelas Job dan sebuah angka penunjuk nomor / nama pekerjaan. *Method* ini akan menyisihkan pekerjaan dengan nomor / nama pekerjaan yang dimasukkan dari parameter *array* kelas Job. *Method* ini akan mengembalikan parameter *array* kelas Job tanpa objek Job dengan nomor / nama pekerjaan yang dimasukkan.

* getAntParty

  Method ini akan mengembalikan informasi mengenai solusi-solusi yang dibentuk oleh kelompok semut ini pada fase pelatihan / pencarian solusi terakhir.

* getAntOpt

  Method ini akan mengembalikan informasi mengenai solusi yang dianggap paling optimal hingga saat ini oleh kelompok semut ini.

* setAntOpt

  Method ini akan menginformasikan dan mengubah nilai dari solusi yang saat ini dianggap paling optimal oleh kelompok semut ini. Nilai dari solusi optimal tersebut akan diubah menjadi nilai solusi baru yang lebih optimal.

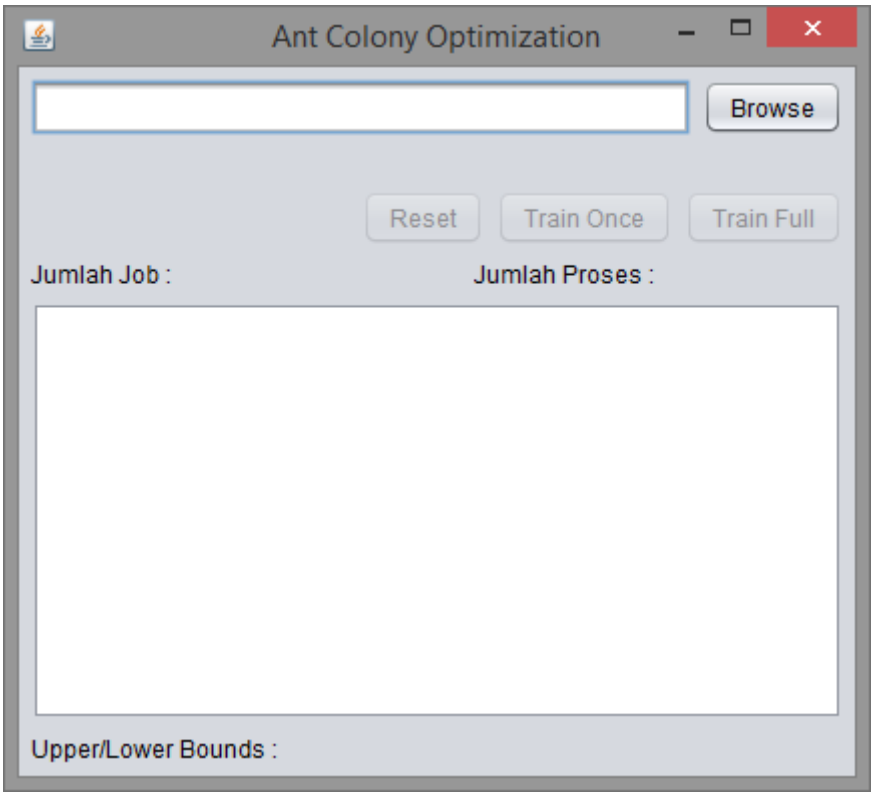
BAB 5

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini, akan dijelaskan hasil implementasi algoritma pada perangkat lunak ini. Cara dan hasil pengujian perangkat lunak juga akan dijabarkan pada bab ini. Hasil pengujian tersebut akan digunakan pengukuran tingkat keoptimalan dari proses optimisasi yang dilakukan algoritma *ant colony* pada perangkat lunak.

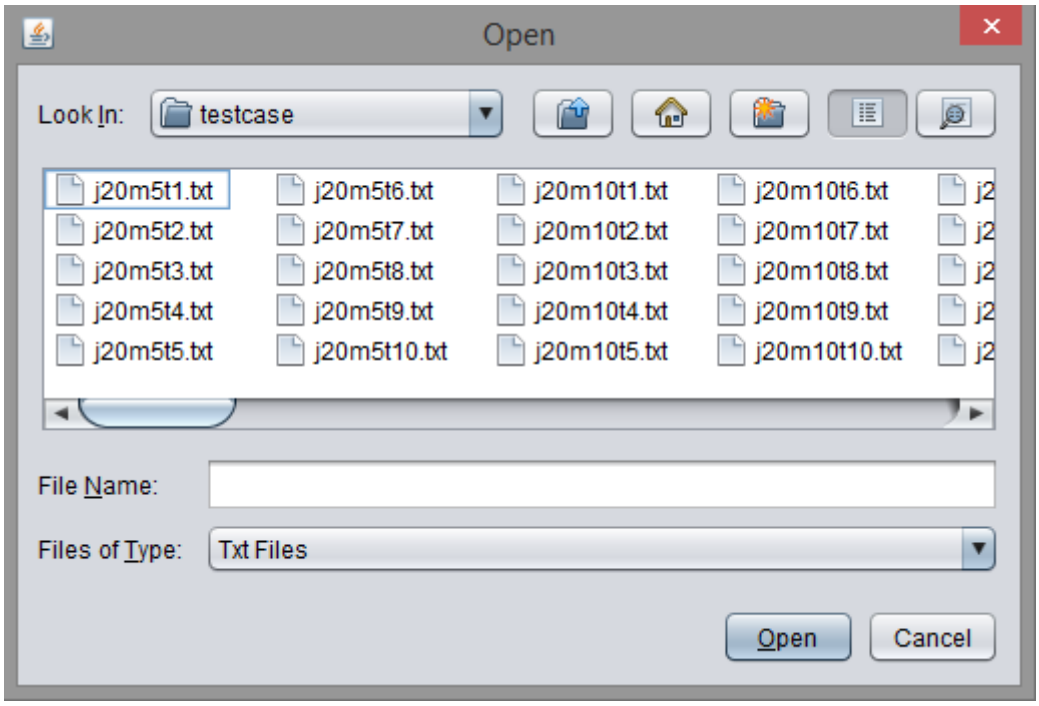
5.1 Implementasi Algoritma

Perangkat lunak ini akan mampu membentuk objek dari kelas AntColonyAlgorithm yang bertugas untuk mencari urutan pengerjaan / solusi yang optimal dari suatu kasus *hybrid flow shop*. Perangkat lunak ini akan mampu menerima data masukan sebuah kasus *hybrid flow shop* dari sebuah *file* teks dan membentuk objek AntColonyAlgorithm yang disesuaikan dengan data-data dari kasus yang diberikan. Tampilan awal dari perangkat lunak akan tampak seperti pada Gambar 5.1.



Gambar 5.1: Tampilan awal dari *interface* perangkat lunak

Pada tampilan awal ini, pengguna belum dapat menjalankan proses optimisasi. Untuk dapat menjalankan proses optimisasi, pengguna perlu memasukkan kasus *hybrid flow shop* terlebih dahulu melalui tombol *"Browse"*. Tombol *"Browse"* tersebut akan membuka *interface* baru seperti pada gambar 5.2. *Interface* tersebut berguna untuk memilih *file* teks yang berisi kasus *hybrid flow shop*.

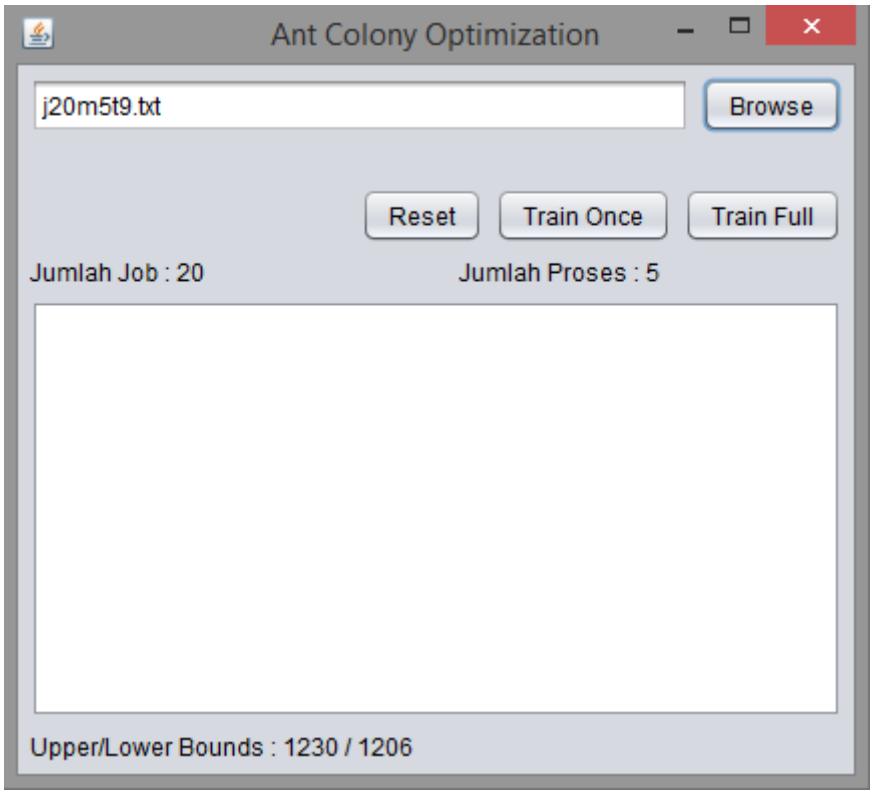


Gambar 5.2: Tampilan *interface* pemilihan kasus *hybrid flow shop*

Setelah kasus *hybrid flow shop* dipilih, perangkat lunak akan mengkonversi kasus tersebut menjadi objek dari kelas Problem dan membentuk objek AntColonyAlgorithm berdasarkan data-data kasus tersebut. Objek AntColonyAlgorithm tersebut akan membentuk objek dari kelas PheromoneDatabase dan AntColonySub yang akan membantu proses pencarian solusi optimal. Pembentukan objek dari kedua kelas tersebut akan diatur oleh objek AntColonyAlgorithm agar sesuai dengan kasus *hybrid flow shop* yang ingin dicari hasil optimalnya.

Objek dari kelas AntColonySub merepresentasikan sebuah kelompok semut yang akan disebar pada saat proses pelatihan algoritma *ant colony*. Objek-objek AntColonySub tersebut akan mampu melakukan pencarian solusi lokal secara bersama-sama. Banyaknya objek AntColonySub yang dibentuk akan disesuaikan dengan banyaknya pekerjaan pada kasus *hybrid flow shop* yang dimasukkan. Semakin banyak pekerjaan pada kasus *hybrid flow shop* yang diberikan, semakin banyak pula objek AntColonySub yang dibentuk.

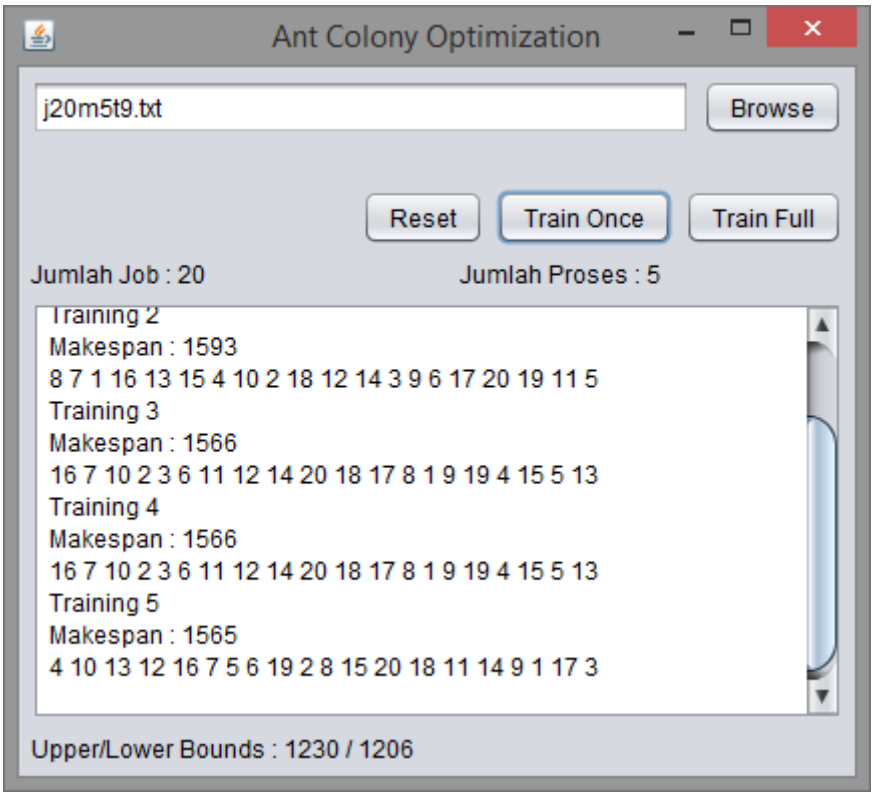
Setelah pengguna memasukkan kasus *hybrid flow shop* dan proses konversi kasus tersebut selesai, pengguna dapat mulai melakukan proses optimisasi. Informasi-informasi penting mengenai kasus *hybrid flow shop* yang ingin dicari hasil optimalnya akan ditampilkan pada *interface* perangkat lunak ini. Informasi mengenai banyaknya pekerjaan dan banyaknya proses / mesin pada kasus *hybrid flow shop* akan ditampilkan di atas *text area* pada *interface*. Nilai *upper* dan *lower bound* juga akan ditampilkan di bawah *text area*. Tampilan *interface* tersebut akan tampak seperti gambar 5.3.



Gambar 5.3: Tampilan *interface* setelah pemilihan kasus *hybrid flow shop*

Pengguna dapat menggunakan beberapa cara dalam melakukan proses optimisasi. Untuk melakukan proses optimisasi dengan menggunakan 1 kali fase pelatihan, pengguna dapat menekan tombol *"Train Once"*. Hasil dari proses optimisasi pada fase pelatihan ini akan ditampilkan pada *text area*. Nilai-nilai pada fase pelatihan ini akan dicatat oleh objek AntColonyAlgorithm dalam bentuk feromon dan akan mempengaruhi proses-proses optimisasi selanjutnya. Dengan dicatatnya fase-fase pelatihan yang dilakukan, hasil proses optimisasi akan cenderung lebih baik dan semakin mendekati solusi yang paling optimal.

Pengguna juga dapat melakukan proses optimisasi dengan melakukan fase pelatihan secara terus menerus hingga suatu kondisi berhenti tercapai. Proses optimisasi tersebut dapat dilakukan dengan menekan tombol *"Train Full"*. Setiap fase pelatihan yang terjadi pada proses ini akan mempengaruhi dengan fase pelatihan selanjutnya. Sama seperti fungsi pada tombol *"Train Once"*, hasil dari proses optimisasi pada setiap fase pelatihan yang terjadi akan ditampilkan pada *text area*.



Gambar 5.4: Tampilan *interface* setelah beberapa proses optimisasi

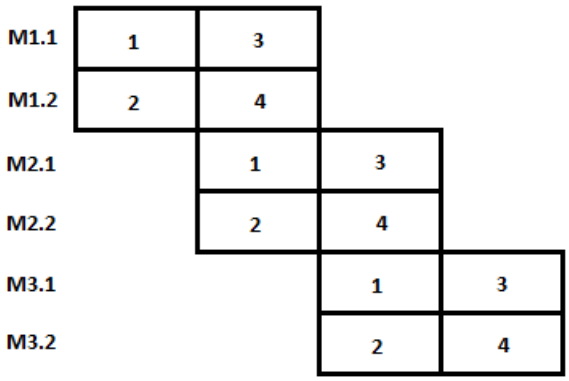
Tampilan *interface* setelah melakukan beberapa kali proses pelatihan / optimisasi akan terlihat seperti gambar 5.4. Pengguna juga mampu mengulang proses optimisasi dari fase pelatihan pertama dengan menekan tombol *"Reset"*. Pengguna juga dapat memasukkan kasus *hybrid flow shop* baru yang ingin dicari hasil optimalnya dengan menekan kembali tombol *"Browse"*. Jika kasus *hybrid flow shop* yang baru telah dipilih, perangkat lunak akan membentuk ulang objek AntColonyAlgorithm agar sesuai dengan kasus baru yang dipilih dan memulai kembali proses optimisasi dari fase pelatihan pertama.

5.2 Pengujian Fungsional

Perangkat lunak ini akan memiliki beberapa fungsi, di antaranya :

- Mengkonversi *file* teks menjadi kasus *hybrid flow shop*.
- Membentuk jalur secara acak.
- Menghitung nilai *makespan* dari solusi kasus *hybrid flow shop*.
- Menambah nilai feromon.
- Melakukan 1 fase pelatihan dari algoritma *ant colony*.
- Melakukan proses optimisasi dengan algoritma *ant colony*.

Proses konversi *file* teks sudah dapat dianggap berhasil setelah perangkat lunak mampu menampilkan informasi banyaknya pekerjaan dan banyaknya proses / mesin pada kasus *hybrid flow shop*. Pengujian proses perhitungan nilai *makespan* akan dilakukan dengan menggunakan kasus *hybrid flow shop* yang sederhana. Kasus *hybrid flow shop* yang digunakan melibatkan 4 buah pekerjaan dan 3 buah proses. Masing-masing pekerjaan memiliki waktu pengerjaan 1 untuk setiap proses. Kasus ini akan memiliki 2 rangkaian mesin untuk pengerjaan setiap proses.



Gambar 5.5: Ilustrasi solusi kasus *hybrid flow shop*

Gambar 5.5 merupakan ilustrasi hasil solusi yang akan dihasilkan dari kasus yang digunakan untuk pengujian perhitungan nilai *makespan*. Berdasarkan ilustrasi tersebut, nilai *makespan* yang dihasilkan akan bernilai 4 untuk setiap urutan pengerjaan. Perangkat lunak yang dibuat sudah mampu menghasilkan nilai *makespan* 4 dari kasus tersebut.

Proses pembentukan jalur secara acak dilakukan untuk membentuk urutan pengerjaan yang berbeda-beda dengan mempertimbangkan nilai feromon yang ada. Berikut ini adalah beberapa hasil proses pembentukan jalur secara acak dari 5 buah pekerjaan dan masing-masing pekerjaan masih memiliki nilai feromon yang sama.

1	5	4	3	1	2
2	4	2	1	5	3
3	2	1	4	5	3
4	5	4	2	3	1
5	3	1	4	2	5
6	3	1	5	4	2
7	4	2	5	1	3
8	1	4	2	5	3
9	3	4	5	2	1
10	5	3	4	1	2
11	1	4	3	5	2
12	4	1	3	2	5
13	5	1	4	2	3
14	2	4	3	1	5
15	2	4	3	5	1
16	5	1	4	3	2
17	5	4	3	2	1
18	5	4	3	2	1
19	2	1	4	5	3
20	1	4	3	2	5
21	2	4	5	1	3
22	5	4	1	3	2
23	2	4	3	1	5
24	2	1	5	4	3
25	5	2	3	4	1

Pembentukan jalur secara acak tersebut sudah baik dan mampu menghasilkan urutan pengerjaan yang berbeda-beda. Proses pelatihan dari algoritma *ant colony* berguna untuk mencari solusi baru yang lebih optimal. Proses pelatihan ini sudah mampu membedakan solusi yang lebih baik dan menampilkan solusi lain yang lebih optimal jika solusi tersebut ditemukan. Setelah proses pelatihan selesai, penambahan nilai feromon dan evaporasi nilai feromon dilakukan.

Tabel 5.1: Tabel nilai feromon

0	14	1	1	1
1	0	1	14	1
1	1	0	1	14
1	1	14	0	1
1	1	1	1	0

Tabel 5.1 merupakan tabel nilai feromon yang dihasilkan dari kasus *hybrid flow shop* dengan 5 buah pekerjaan. Setelah menjalankan beberapa kali proses pelatihan, nilai tabel feromon yang pada awalnya sama berubah menjadi seperti pada tabel. Pada tabel tersebut dapat dilihat bahwa terdapat beberapa indeks dengan nilai yang jauh lebih besar dibandingkan dengan indeks-indeks yang lain. Indeks-indeks feromon tersebut merupakan indeks-indeks feromon yang membantu pembentukan suatu solusi dan solusi tersebut merupakan solusi yang dianggap paling optimal pada saat ini.

Pemantauan berlangsungnya proses optimisasi dari algoritma *ant colony* sulit untuk dilakukan. Pembuktian mengenai bekerjanya proses optimisasi dari algoritma *ant colony* tidak dapat dilakukan secara teoritis / dengan menggunakan perhitungan tertentu. Algoritma *ant colony* menggunakan proses randomisasi untuk membentuk solusi dan solusi tersebut tidak selalu sama. Nilai feromon yang mempengaruhi proses randomisasi tersebut juga selalu berubah-ubah pada setiap fase pelatihannya. Pembuktian bekerjanya proses optimisasi hanya dapat dilakukan dengan eksperimen secara terus menerus.

5.3 Eksperimen

Pada bagian ini akan dijelaskan mengenai tata cara yang digunakan untuk melakukan eksperimen dan informasi apa saja yang diambil dari eksperimen tersebut. Hasil data dan analisa dari eksperimen tersebut juga akan dijelaskan.

5.3.1 Cara Eksperimen

Proses eksperimen akan dilakukan untuk mengukur tingkat performa dari proses optimisasi dengan menggunakan algoritma *ant colony*. Tingkat performa algoritma *ant colony* ini akan diukur berdasarkan perbandingan nilai *makespan* yang dihasilkan dengan nilai *upper bound* dari suatu kasus. Eksperimen ini akan dilakukan pada *laptop* dengan spesifikasi :

- *Operating system* Windows 8.1
- Prosessor Intel Core i5-4210U 1.7GHz
- Memori 8GB DDR3 L

- Bahasa pemrograman Java (jdk) versi 1.7
- Aplikasi untuk menjalankan bahasa pemrograman java (jre) versi 7

Eksperimen ini akan mencari tahu pengaruh jumlah semut, banyaknya pekerjaan, dan banyaknya mesin terhadap hasil optimisasi yang diberikan oleh algoritma *ant colony*. Eksperimen ini juga akan mencari tahu apakah algoritma *ant colony* ini sudah berjalan dengan benar. Data masukan yang akan digunakan berasal dari *framework* soa.iti.es. Jenis data kasus yang akan digunakan tergolong pada data kasus "*Instances for hybrid flowshops with identical machines per stage and due windows*".

Proses eksperimen akan dilakukan dengan memasukkan *file* teks dari beberapa jenis kasus *hybrid flow shop* yang disediakan oleh soa.iti.es. Masing-masing jenis kasus akan dibedakan berdasarkan banyaknya pekerjaan, proses, dan mesin pada setiap tahap proses. Eksperimen akan melakukan optimisasi pada 10 kasus yang berbeda untuk setiap jenis kasus.

Proses eksperimen akan melakukan proses optimisasi untuk setiap data kasus sebanyak 5 kali. Hal ini dilakukan untuk mendapatkan rata-rata nilai *makespan* yang didapatkan dari optimisasi sebuah data kasus. Pada eksperimen akan dilakukan optimisasi dengan menggunakan tombol *"Train Full"* secara berulang hingga fase pelatihan telah dilakukan lebih dari 200 kali.

Batas 200 kali fase pelatihan tersebut digunakan berdasarkan pengujian pada kasus-kasus acak yang sebelumnya telah dilakukan. Pada 100 pelatihan awal cenderung sering ditemukan solusi baru yang lebih optimal pada setiap fase pelatihan, akan tetapi pada 100 pelatihan selanjutnya cenderung hanya 1 atau 2 kali ditemukan solusi baru yang lebih optimal. Pada 100 fase pelatihan selanjutnya solusi baru yang lebih optimal akan semakin jarang didapatkan atau bahkan sama sekali tidak ditemukan solusi baru yang lebih optimal. Berdasarkan hal-hal tersebut dapat diasumsikan bahwa algoritma *ant colony* sudah mencapai solusi yang dianggap optimal dengan 200 kali fase pelatihan dan untuk menemukan solusi lain yang lebih optimal akan dibutuhkan waktu yang cukup lama.

Pada setiap kasus, eksperimen akan melakukan optimisasi dengan 2 jumlah semut yang berbeda. Proses optimisasi ini akan melakukan proses pelatihan algoritma *ant colony* dengan melibatkan n^3 dan n^2 semut, di mana n adalah banyaknya pekerjaan pada kasus *hybrid flow shop* yang dipilih. Jumlah semut tersebut digunakan dengan asumsi bahwa tingkat kompleksitas dari algoritma ini sebaiknya lebih baik dibandingkan dengan cara *brute force* dengan tingkat kompleksitas $n!$ dan juga dengan asumsi bahwa jumlah semut yang disebar pada setiap fase pelatihan mempengaruhi tingkat kompleksitas dari algoritma *ant colony*.

Proses optimisasi ini juga akan melibatkan n buah objek dari kelas AntColonySub, di mana n adalah banyaknya pekerjaan pada kasus *hybrid flow shop* yang dipilih. Semut-semut yang melakukan proses pelatihan akan dibagi secara merata pada masing-masing objek AntColonySub. Jumlah kelas AntColonySub tersebut digunakan dengan asumsi bahwa semakin banyak pekerjaan pada kasus *hybrid flow shop*, maka akan semakin rumit suatu kasus.

Semakin rumit kasus, maka semakin lama waktu yang dibutuhkan untuk proses optimisasi. Banyaknya kelas AntColonySub mampu mengurangi waktu optimisasi. Jika banyaknya kelas AntColonySub disesuaikan dengan banyaknya pekerjaan, maka proses optimisasi akan berlangsung dengan waktu yang relatif lebih cepat dan sesuai dengan kerumitan dari suatu kasus.

Hasil eksperimen yang akan dicatat adalah nilai *makespan* yang dihasilkan, urutan pengerjaan / solusi yang menghasilkan nilai *makespan* tersebut, dan pada fase pelatihan ke berapa solusi tersebut didapatkan. Hasil dari nilai *makespan* yang diberikan akan dibandingkan dengan nilai LBCmax yang telah disediakan oleh soa.iti.es. LBCmax itu sendiri akan dianggap sebagai nilai *upper bound* sebagai nilai *makespan* target. Jika nilai *makespan* yang dihasilkan semakin mendekati nilai *upper bound*, maka semakin baik pula penilaian terhadap tingkat optimisasi algoritma ini.

Eksperimen ini akan menggunakan kasus *hybrid flow shop* dengan jumlah mesin yang sama pada setiap tahap proses. Data kasus yang digunakan memiliki data tambahan berupa rentang waktu pengerjaan dari suatu pekerjaan, akan tetapi data tersebut tidak akan digunakan dalam eksperimen ini.

5.3.2 Hasil Eksperimen

	Jumlah semut n^2						Jumlah semut n^3					
	Makespan - Upper Bound			Fase Pelatihan			*Makespan - Upper Bound*			Fase Pelatihan		
Kasus	Rata-rata	Min	Max	Rata-rata	Min	Max	Rata-rata	Min	Max	Rata-rata	Min	Max
10 job, 2 proses, 2 mesin	11.18	0	21	91	44	144	8.26	0	15	53	10	94
10 job, 2 proses, 3 mesin	6.54	0	12.4	59	11	125	5.6	0	11	38	3	94
10 job, 4 proses, 2 mesin	33.26	16	45.4	83	27	137	28.98	12.2	40.8	41	13	88
10 job, 4 proses, 3 mesin	8.44	0	19	61	13	171	6.5	0	16.6	41	9	124
20 job, 2 proses, 2 mesin	28.18	3.2	50.4	114	86	164	21.38	0.8	39	115	71	160
20 job, 2 proses, 3 mesin	20.62	10.2	37.8	116	83	151	14.74	5	30.2	104	52	149
20 job, 4 proses, 2 mesin	56.48	3	98.4	122	75	151	46.14	0.6	81.8	116	67	144
20 job, 4 proses, 3 mesin	55.86	31	74.4	119	93	138	47	26	67.4	104	76	132
50 job, 5 proses, 3 mesin	194.02	155.4	219.6	127	75	153	173.24	133.6	196.6	124	79	165
50 job, 5 proses, 5 mesin	124.5	100	146.6	110	77	151	110.2	84.4	129.2	126	99	154
50 job, 10 proses, 3 mesin	296.42	216.8	351	120	79	149	271.64	202	320.8	134	104	165
50 job, 10 proses, 5 mesin	193.1	129.8	230.6	113	70	141	178.04	114.2	210.2	125	80	157

Gambar 5.6: Tabel hasil eksperimen

Gambar 5.6 di atas berisi tabel data hasil eksperimen yang telah dilakukan. Untuk setiap kasus telah dilakukan perhitungan rata-rata selisih nilai *makespan* dengan nilai *upper bound* dari 5 kali pengulangan proses optimisasi. Nilai-nilai tersebut kemudian akan dikelompokkan berdasarkan banyaknya semut yang disebar dan jenis kasus *hybrid flow shop* yang menghasilkannya. Jenis dari suatu kasus dapat didefinisikan berdasarkan banyaknya pekerjaan, proses, dan mesin pada setiap tahap proses dari suatu kasus.

Dari nilai-nilai tersebut kemudian dicari rata-rata nilai selisih yang dihasilkan dari suatu jenis kasus. Dengan dilakukannya hal ini, kita dapat mengetahui ruang lingkup nilai selisih yang mungkin dihasilkan dari suatu jenis kasus. Nilai rata-rata selisih yang minimal dan maksimal juga dicari untuk mempermudah menganalisa besar nilai selisih yang mungkin dihasilkan.

5.3.3 Analisa Hasil Eksperimen

Berdasarkan hasil eksperimen, dapat dilihat bahwa algoritma *ant colony* mampu menghasilkan nilai *makespan* yang paling optimal. Hal ini dapat dilihat dari data minimal dari nilai minimal pada selisih *makespan* dengan *upper bound*. Nilai selisih 0 menunjukkan bahwa algoritma *ant colony* berhasil selalu mendapatkan nilai *makespan* yang paling optimal dari 5 kali proses optimisasi suatu kasus yang sama. Hal ini dapat disimpulkan mengingat bahwa nilai *makespan* yang dihasilkan tidak akan lebih kecil dari nilai *makespan* yang paling optimal (*upper bound*).

Nilai *makespan* optimal tersebut akan semakin sulit didapat jika jumlah pekerjaan dari suatu kasus semakin besar. Nilai selisih *makespan* dengan *upper bound* akan semakin besar jika kasus yang ingin dioptimisasi semakin rumit. Pada optimisasi kasus *hybrid flow shop* dengan 50 buah pekerjaan, nilai selisih yang dihasilkan cenderung sangat besar. Jika dibandingkan dengan nilai selisih pada optimisasi kasus *hybrid flow shop* dengan 20 buah pekerjaan, nilai selisih pada kasus dengan 20 pekerjaan hanya 1/4 dari nilai selisih pada kasus dengan 50 buah pekerjaan. Semakin banyak pekerjaan yang ada pada kasus *hybrid flow shop*, maka tingkat keoptimalan dari hasil optimisasi algoritma *ant colony* ini akan semakin rendah.

Cara untuk menambah tingkat keoptimalan hasil optimisasi dapat dilakukan dengan menambah banyak semut yang disebar pada fase pelatihan. Hal ini dapat dilihat pada gambar 5.6, pada hasil optimisasi kasus-kasus dengan jumlah semut yang berbeda. Hasil nilai selisih yang dihasilkan dengan n^3 semut pada setiap fase pelatihan selalu bernilai lebih kecil jika dibandingkan dengan yang dihasilkan oleh n^2 semut. Dari hal tersebut dapat dilihat bahwa semakin banyak semut yang disebar, maka hasil optimisasi yang dihasilkan juga akan cenderung lebih mendekati nilai yang paling optimal.

Banyaknya semut yang disebar juga mempengaruhi banyaknya fase pelatihan yang dibutuhkan untuk mencapai hasil optimal. Banyaknya fase pelatihan yang dilalui relatif lebih sedikit jika proses optimisasi dilakukan dengan jumlah semut yang lebih banyak. Di sisi lain, semakin rumit suatu kasus yang ingin dioptimisasi, maka semakin banyak pula fase pelatihan yang perlu dilalui untuk mencapai hasil yang optimal.

Perlu diingat pula bahwa semakin banyaknya semut bukan merupakan jaminan ditemukannya solusi yang lebih optimal. Ada sebagian kecil dari proses pengujian yang menggunakan n^2 semut mampu menghasilkan solusi yang lebih baik dibandingkan dengan hasil solusi dari proses pengujian menggunakan n^3 semut. Semakin banyaknya semut yang disebar hanya akan menambah kemungkinan ditemukannya solusi yang lebih baik.

Banyaknya fase pelatihan yang terjadi tentu saja sangat berpengaruh pada hasil optimisasi yang dihasilkan. Akan tetapi, banyaknya fase pelatihan juga bukan merupakan jaminan dihasilkannya solusi yang lebih optimal. Banyaknya mesin pada kasus *hybrid flow shop* tidak mempengaruhi hasil optimisasi oleh algoritma *ant colony*. Hal ini dapat dilihat dari variasi nilai selisih yang dihasilkan pada eksperimen.

Pada data hasil eksperimen dari beberapa kasus, terdapat solusi yang relatif lebih baik dan banyak fase pelatihan yang dibutuhkan untuk mencapai solusi tersebut juga lebih sedikit. Setiap fase pelatihan yang terjadi terkadang kurang efektif dan tidak menghasilkan solusi lain yang lebih optimal. Untuk menjamin tingkat efektifitas dari suatu fase pelatihan, banyaknya semut yang disebar pada setiap fase pelatihan harus ditambah. Dengan memperbanyak jumlah semut pada

setiap fase pelatihan, kemungkinan ditemukannya solusi baru yang lebih optimal pada suatu fase pelatihan akan semakin besar.

Cukup sulit untuk membuktikan bahwa algoritma ini telah bekerja sesuai dengan yang diinginkan. Algoritma *ant colony* sulit untuk dianalisa secara teoritis, karena membentuk solusi secara acak. Nilai-nilai feromon yang mempengaruhi pembentukan solusi tersebut juga selalu berubah-ubah. Cara untuk menganalisa algoritma ini hanya dengan menggunakan cara eksperimental dengan mengambil beberapa sampel hasil solusi.

Tabel 5.2: Tabel hasil optimisasi satu kasus acak

Makespan	*training*	Urutan Pekerjaan
2559	109	20 4 2 16 15 10 13 5 9 18 3 12 14 19 17 6 1 8 11 7
2555	72	20 14 4 17 16 15 10 13 5 9 18 11 12 1 6 3 2 19 8 7
2551	134	20 16 4 1 13 5 9 18 10 14 11 12 15 17 8 19 6 3 2 7
2553	54	1 4 20 14 16 15 10 13 5 9 18 19 17 2 11 6 8 3 12 7
2526	95	4 2 16 15 10 13 5 9 18 3 14 20 11 12 17 6 1 8 19 7

Tabel 5.2 di atas merupakan hasil pengujian salah satu kasus *hybrid flow shop* dengan 20 buah pekerjaan. Hasil optimisasi di atas dapat dijadikan data sampel untuk membuktikan bahwa algoritma *ant colony* yang diaplikasikan sudah bekerja dengan benar. Hal ini dapat dilihat dari dipilihnya pekerjaan 7 sebagai pekerjaan untuk dikerjakan terakhir pada setiap solusi yang didapatkan. Pekerjaan 20 juga cenderung untuk dikerjakan pertama kali.

Dalam satu kasus *hybrid flow shop*, algoritma *ant colony* akan cenderung mengarahkan dirinya untuk membentuk sebuah solusi yang sama. Solusi yang dijadikan tujuan tentu saja merupakan solusi yang paling optimal dari kasus *hybrid flow shop* tersebut. Pada perangkat lunak ini, algoritma *ant colony* akan berusaha membentuk urutan-urutan pekerjaan yang terbaik berdasarkan prioritas diambilnya suatu pekerjaan setelah diambilnya pekerjaan yang lain.

Jika diperhatikan dengan seksama, pada Tabel 5.2 terdapat kelompok pekerjaan yang selalu diprioritaskan untuk dikerjakan secara terurut. Hampir pada setiap hasil optimisasi memiliki potongan urutan pengerjaan [16 15 10 13 5 9 18 3]. Hal ini membuktikan bahwa algoritma ini memang sedang mengarah pada solusi yang sama dan solusi optimal tersebut kemungkinan besar juga memiliki potongan urutan pengerjaan tersebut. Jika fase pelatihan dari hasil proses pengujian tersebut dilanjutkan, solusi yang dihasilkan kemungkinan besar akan semakin mirip bahkan dapat serupa.

BAB 6

KESIMPULAN DAN SARAN

Pada bab ini akan dijabarkan mengenai hal-hal apa saja yang dapat disimpulkan dari karya tulis ini. Beberapa saran untuk pengembangan selanjutnya dari perangkat lunak ini juga akan dijabarkan.

6.1 Kesimpulan

Hal-hal yang dapat disimpulkan dari karya tulis ini adalah :

- Proses penjadwalan *hybrid flow shop* adalah penentuan urutan pengerjaan sekumpulan pekerjaan yang akan dikerjakan pada suatu rangkaian mesin. Setiap proses dari pekerjaan-pekerjaan tersebut akan dikerjakan secara terurut dan sistematis. Urutan pengerjaan yang berbeda akan menghasilkan nilai *makespan* yang berbeda. Penentuan urutan pengerjaan yang baik sangat diperlukan agar nilai *makespan* yang dihasilkan merupakan nilai *makespan* yang paling optimal.

- Algoritma *ant colony* adalah algoritma optimisasi yang bekerja dengan mengikuti prinsip kerja dari koloni semut. Algoritma ini akan mencoba solusi-solusi yang dapat dianggap optimal. Penentuan solusi optimal tersebut dipengaruhi oleh nilai feromon. Semakin besar nilai feromon suatu solusi, semakin besar kecenderungan bagi solusi tersebut untuk dipilih sebagai solusi yang optimal.

- Algoritma *ant colony* dapat diaplikasikan untuk proses optimisasi penjadwalan *hybrid flow shop* dengan membantu proses pembentukan urutan pengerjaan. Proses pembentukan pekerjaan tersebut akan memperhatikan kecenderungan dipilihnya sebuah pekerjaan setelah suatu pekerjaan lain. Kecenderungan dipilihnya pekerjaan tersebut akan dipengaruhi oleh nilai feromon yang disimpan algoritma *ant colony*.

- Algoritma *ant colony* berhasil mendapatkan solusi dari permasalahan *hybrid flow shop*. Untuk setiap pengulangan proses optimisasi pada kasus yang sama, algoritma ini cenderung memberikan hasil-hasil yang berbeda. Akan tetapi terdapat beberapa potongan solusi yang mirip di antara solusi-solusi tersebut. Hal ini menunjukkan bahwa algoritma *ant colony* sedang berusaha untuk membentuk satu solusi optimal yang sama.

- Berdasarkan hasil perbandingan nilai *upper bound* dan rata-rata nilai *makespan* yang dihasilkan, algoritma *ant colony* mampu mencapai hasil yang paling optimal dari suatu kasus *hybrid flow shop*. Akan tetapi, semakin rumit suatu kasus yang ingin dioptimisasi, semakin sulit pula bagi algoritma *ant colony* untuk mencapai hasil yang paling optimal.

- Banyaknya semut yang dibentuk pada setiap fase pelatihan algoritma *ant colony* sangat mempengaruhi hasil solusi akhir yang diberikan. Semakin banyak jumlah semut, hasil solusi akhir yang diberikan juga akan cenderung semakin baik. Akan tetapi, jika semut yang dibentuk semakin banyak, maka hal tersebut akan meningkatkan tingkat kompleksitas dari algoritma *ant colony* dan memperlambat proses optimisasi.

- Agar proses optimisasi dapat berlangsung dengan baik dalam waktu yang relatif cepat, kombinasi yang baik dari 3 aspek penting dalam algoritma *ant colony* perlu ditentukan. Aspek-aspek tersebut merupakan banyaknya fase pelatihan, banyaknya semut yang disebar pada setiap fase pelatihan, dan banyaknya objek AntColonySub yang mampu melakukan proses perhitungan solusi optimal secara bersama-sama.

6.2 Saran

Berikut ini adalah beberapa saran dan ide yang didapat selama proses penelitian.

- Pengguna dapat menentukan banyaknya fase pelatihan dari algoritma *ant colony* yang akan dijalankan oleh perangkat lunak. Dengan diaplikasikannya ide ini, proses optimisasi diharapkan dapat berlangsung sesuai dengan kebutuhan pengguna.

- Banyaknya semut yang disebar pada setiap fase pelatihan dapat dibuat dinamis/berbeda-beda satu sama lain. Hal ini dapat dilakukan dengan mempertimbangkan banyaknya fase pelatihan yang telah terjadi dan hasil optimisasi dari fase-fase pelatihan sebelumnya. Diharapkan dengan pengaplikasian ide ini, proses optimisasi dapat berlangsung lebih cepat dan setiap fase pelatihan dapat menjadi lebih efektif. Ide ini dapat diaplikasikan dengan menggunakan jaringan syaraf tiruan.

- Nilai pada setiap indeks feromon sebaiknya mampu disimpan dalam bentuk presentase. Penyimpanan nilai feromon dalam bentuk suatu jumlah nilai akan sangat memakan memori dan suatu saat pasti akan melebihi batas dari penyimpanan memori yang dimiliki. Dengan menyimpan nilai feromon dalam bentuk persentase, hal tersebut tidak akan dimungkinkan terjadi dan proses pembentukan solusi juga akan cenderung lebih cepat.

- Nilai konstanta yang digunakan untuk menentukan nilai feromon yang ditambahkan dapat dibuat dinamis / berubah-ubah. Hal ini dilakukan agar solusi-solusi yang sudah dianggap buruk tidak akan memberikan dampak kepada nilai feromon yang disimpan. Dengan diaplikasikannya ide ini, solusi-solusi yang buruk akan cenderung semakin jarang dipilih.

- Proses evaporasi feromon yang dinamis. Proses evaporasi feromon dapat dilakukan secara bebas selama proses optimisasi. Hal ini bertujuan agar nilai feromon yang disimpan dapat dengan cepat berubah jika ditemukan suatu solusi baru yang lebih optimal. Dengan diaplikasikannya ide ini, solusi baru yang lebih optimal akan lebih sering dipilih dan semakin mempercepat penambahan nilai feromon dari solusi tersebut.

- Diaplikasikannya proses seleksi solusi dalam proses penambahan feromon. Proses seleksi ini bertujuan untuk mengambil beberapa solusi terbaik dari solusi-solusi yang dicoba pada satu fase pelatihan. Dengan diaplikasikannya ide ini, solusi-solusi yang dianggap buruk tidak akan memberikan pengaruh pada nilai feromon dan hanya solusi-solusi yang dianggap baik yang akan mempengaruhi nilai feromon.

- Penyimpanan hasil solusi yang telah dicoba. Penyimpanan tersebut menampung solusi dan nilai yang dihasilkan dari solusi tersebut. Dengan adanya penyimpanan hasil solusi, proses perhitungan dari solusi yang sama tidak akan dilakukan terus menerus. Dengan diaplikasikannya ide ini, diharapkan proses optimisasi dapat berlangsung dengan lebih cepat.

DAFTAR REFERENSI

[1] P. Michael, *Schedulling: Theory, Algorithms, and System.* Prentice Hall International, New Jersey, 2002.

[2] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 1–18, 2010.

[3] I. Osman and C. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.

[4] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 1061–1071, 1996.

[5] C. Rajendran and H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *European Journal of Operational Research*, vol. 155, no. 2, pp. 426–438, 2004.

[6] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930–1947, 2007.

[7] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical computer science*, vol. 344, no. 2, pp. 243–278, 2005.

[8] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, 2006.

[9] V. Selvi and R. Umarani, "An ant colony optimization for job scheduling to minimize makespan time," *International Journal of Computer and Communication Technology*, vol. 3, 2012.

LAMPIRAN A

DATA PENGUJIAN KASUS *HYBRID FLOW SHOP* DENGAN 10 PEKERJAAN

Tabel A.1: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 2 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	367	367	169	6 10 2 3 7 1 4 9 5 8
		371	85	2 3 7 10 4 6 1 9 5 8
		367	54	6 1 10 7 4 3 9 2 5 8
		375	39	2 3 1 9 7 4 6 10 5 8
		367	150	6 3 7 4 1 10 9 2 5 8
	Rata-rata	369.4	99.4	
2	311	330	157	4 5 3 9 6 10 2 7 8 1
		330	122	4 5 3 9 6 10 2 7 8 1
		329	71	4 9 10 2 7 6 1 5 3 8
		330	29	4 3 5 6 9 10 2 7 8 1
		326	36	3 6 4 1 9 10 2 7 5 8
	Rata-rata	329	83	
3	259	273	30	1 10 9 3 4 8 6 5 2 7
		271	69	1 8 3 9 5 4 10 2 6 7
		271	139	8 1 3 9 10 4 5 6 2 7
		271	160	1 8 3 9 10 4 5 2 6 7
		271	115	8 1 3 9 10 4 5 2 6 7
	Rata-rata	271.4	102.6	
4	349	350	68	1 4 2 6 8 9 10 7 5 3
		350	13	1 7 8 2 9 6 10 4 5 3
		351	178	8 1 9 6 2 4 7 10 5 3
		349	100	8 1 4 10 2 6 7 9 5 3
		352	109	4 6 2 7 8 1 10 9 5 3
	Rata-rata	350.4	93.6	
5	263	284	193	6 4 9 5 10 1 3 8 7 2
		287	139	5 4 10 6 1 8 3 9 7 2
		288	136	4 6 5 1 10 3 9 8 7 2
		283	63	4 6 1 3 8 10 7 9 5 2
		278	186	9 5 6 4 10 1 3 8 7 2
	Rata-rata	284	143.4	

Tabel A.2: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 2 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	260	280	81	4 10 3 5 2 7 1 8 9 6
		280	148	4 10 3 5 2 7 1 8 9 6
		278	173	3 7 4 2 1 5 10 8 6 9
		279	126	9 4 2 10 1 7 5 8 3 6
		273	72	1 9 4 2 10 7 5 8 3 6
	Rata-rata	278	120	
7	314	328	107	1 10 3 2 5 6 4 8 7 9
		326	42	10 1 6 2 3 8 5 7 9 4
		330	125	1 4 8 6 10 3 2 5 7 9
		326	111	1 4 8 6 3 2 10 5 7 9
		327	152	1 8 10 5 2 3 6 7 9 4
	Rata-rata	327.4	107.4	
8	366	366	37	8 5 10 4 6 3 7 9 2 1
		366	133	5 4 10 8 7 3 6 9 2 1
		366	60	5 4 7 3 10 8 6 9 2 1
		366	23	5 3 7 4 10 9 8 6 2 1
		366	14	5 9 10 4 7 8 3 6 2 1
	Rata-rata	366	53.4	
9	312	328	19	7 9 3 10 1 2 5 8 4 6
		334	97	1 9 3 8 10 5 2 4 6 7
		333	21	7 9 2 3 1 10 5 8 4 6
		325	36	9 3 5 2 1 10 8 4 6 7
		326	44	9 3 2 8 1 10 5 4 7 6
	Rata-rata	329.2	43.4	
10	328	339	52	5 3 8 7 1 4 10 9 2 6
		335	52	3 5 8 2 10 7 9 1 6 4
		336	153	8 2 5 7 9 10 1 6 4 3
		335	5	5 1 8 7 4 9 10 6 2 3
		335	42	5 2 8 7 9 10 1 6 4 3
	Rata-rata	336	60.8	

Tabel A.3: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 2 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	367	367	20	6 3 4 2 1 7 10 9 5 8
		367	9	10 6 2 3 7 1 4 9 5 8
		367	85	2 3 7 9 10 4 6 1 5 8
		367	4	6 3 7 4 1 2 9 10 5 8
		367	36	7 6 9 3 10 2 4 1 5 8
	Rata-rata	367	30.8	
2	311	327	15	10 4 5 6 3 7 2 9 8 1
		324	48	4 6 3 1 9 10 2 7 5 8
		326	39	10 2 3 6 7 5 9 4 8 1
		324	13	4 6 3 1 7 10 2 9 5 8
		325	9	3 10 6 5 7 4 2 9 1 8
	Rata-rata	325.2	24.8	
3	259	271	20	1 8 3 9 10 4 6 5 2 7
		271	74	8 1 3 9 10 4 5 2 6 7
		271	67	8 1 3 9 10 4 6 5 2 7
		271	17	1 8 3 9 10 4 5 2 6 7
		271	43	8 1 3 9 5 4 10 2 6 7
	Rata-rata	271	44.2	
4	349	349	186	2 1 9 10 8 6 7 4 5 3
		349	15	1 2 9 10 8 6 7 4 5 3
		349	102	8 1 4 10 2 6 7 9 5 3
		349	37	1 2 9 10 8 6 7 4 5 3
		349	109	2 1 9 10 8 6 7 4 5 3
	Rata-rata	349	89.8	
5	263	278	168	9 6 5 4 10 1 3 8 7 2
		278	69	5 9 6 4 10 1 3 8 7 2
		278	42	9 5 6 4 10 1 3 8 7 2
		278	82	9 5 6 4 10 1 3 8 7 2
		278	107	5 9 6 4 10 1 3 8 7 2
	Rata-rata	278	93.6	

Tabel A.4: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 2 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	260	273	40	1 9 4 2 10 7 5 8 3 6
		273	50	1 9 4 2 5 7 10 8 3 6
		273	97	7 9 4 2 5 1 10 8 3 6
		271	164	9 4 2 1 5 7 10 8 3 6
		271	12	4 5 9 2 7 1 10 8 3 6
	Rata-rata	272.2	72.6	
7	314	324	16	8 6 2 10 5 3 1 7 9 4
		323	41	8 6 2 10 1 3 5 7 9 4
		323	137	6 8 2 10 1 3 5 7 9 4
		325	111	4 5 8 2 10 1 3 6 7 9
		325	145	6 4 8 2 10 1 3 5 7 9
	Rata-rata	324	90	
8	366	366	9	5 4 10 8 7 3 6 9 2 1
		366	4	6 4 9 5 10 8 7 3 2 1
		366	5	4 7 3 5 10 8 6 9 2 1
		366	11	5 8 10 3 6 4 7 9 2 1
		366	19	10 3 9 6 4 5 7 8 2 1
	Rata-rata	366	9.6	
9	312	325	12	9 3 1 2 5 8 10 4 6 7
		325	59	3 9 2 1 10 8 5 4 6 7
		325	23	9 3 2 5 1 10 8 4 6 7
		323	82	9 3 1 10 8 5 2 4 6 7
		323	83	9 3 1 10 8 5 2 4 6 7
	Rata-rata	324.2	51.8	
10	328	335	21	3 5 8 7 2 9 10 1 6 4
		335	33	8 5 10 1 9 7 2 6 4 3
		335	21	5 2 8 10 4 7 9 6 1 3
		335	14	8 5 2 7 4 1 10 9 6 3
		335	7	5 3 8 7 2 4 9 10 6 1
	Rata-rata	335	19.2	

Tabel A.5: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 3 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	145	145	1	8 3 2 5 7 1 10 4 6 9
		145	2	2 3 8 7 1 5 10 4 6 9
		145	12	5 3 2 8 7 1 10 4 6 9
		145	35	3 5 2 7 10 4 1 9 8 6
		145	4	2 8 5 3 10 7 4 1 9 6
	Rata-rata	145	10.8	
2	204	207	94	9 1 7 10 8 3 5 4 2 6
		207	80	9 7 10 8 1 3 5 4 2 6
		209	120	9 2 7 10 8 1 3 5 6 4
		205	192	7 9 10 8 1 3 5 4 6 2
		207	138	9 8 7 10 1 3 5 4 2 6
	Rata-rata	207	124.8	
3	202	210	38	7 3 5 8 1 10 9 2 4 6
		210	10	8 5 7 1 3 10 9 2 4 6
		215	66	1 10 3 5 7 9 2 8 4 6
		212	11	8 1 5 3 7 10 6 9 2 4
		210	103	3 5 7 8 1 10 9 2 4 6
	Rata-rata	211.4	45.6	
4	166	172	53	7 6 9 1 5 4 3 10 2 8
		172	96	6 1 7 9 3 5 4 10 2 8
		171	160	6 8 7 9 4 5 1 3 10 2
		171	50	6 7 8 9 4 5 1 3 10 2
		171	122	4 7 6 8 9 5 1 3 10 2
	Rata-rata	171.4	96.2	
5	188	192	103	10 2 8 9 5 3 6 1 4 7
		192	42	6 2 10 3 5 8 9 1 4 7
		192	13	10 8 6 2 5 3 9 1 4 7
		193	122	10 2 6 3 8 5 9 1 7 4
		192	92	8 2 10 9 5 3 6 1 4 7
	Rata-rata	192.2	74.4	

Tabel A.6: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 3 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	197	206	72	6 8 3 5 10 1 4 9 2 7
		208	75	3 2 10 6 5 9 4 8 1 7
		207	78	1 3 10 5 6 4 9 8 2 7
		208	29	2 3 10 5 6 9 8 4 1 7
		206	43	1 3 6 5 10 8 4 9 2 7
	Rata-rata	207	59.4	
7	220	234	43	2 7 3 5 4 10 1 8 6 9
		236	16	3 5 2 7 8 10 1 4 6 9
		234	41	2 7 3 5 4 10 1 8 6 9
		229	148	3 7 2 4 5 9 8 1 6 10
		229	22	2 4 3 7 5 6 8 1 10 9
	Rata-rata	232.4	54	
8	199	203	8	7 9 8 6 5 3 10 1 2 4
		204	14	9 8 10 4 5 3 7 6 2 1
		203	39	9 7 10 6 1 3 5 8 2 4
		203	57	9 8 10 1 7 3 5 6 2 4
		203	46	8 10 9 1 7 3 5 6 2 4
	Rata-rata	203.2	32.8	
9	224	230	10	7 3 8 9 10 5 4 6 2 1
		232	14	6 7 8 9 3 10 5 4 2 1
		230	31	9 8 7 10 5 3 4 1 2 6
		228	51	8 7 5 10 9 3 4 2 1 6
		227	22	6 5 7 8 10 9 3 4 2 1
	Rata-rata	229.4	25.6	
10	167	180	14	5 6 10 2 7 9 8 3 4 1
		178	45	10 2 6 9 7 8 5 3 4 1
		178	154	6 10 5 4 9 2 7 8 3 1
		178	62	6 9 10 7 2 5 8 3 4 1
		178	43	10 4 7 5 2 9 8 3 6 1
	Rata-rata	178.4	63.6	

Tabel A.7: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 3 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	145	145	1	10 2 5 3 4 7 1 9 6 8
		145	1	8 3 2 5 10 7 4 1 9 6
		145	2	2 1 8 7 5 10 4 9 6 3
		145	4	7 2 4 8 5 10 1 6 3 9
		145	4	2 7 4 10 5 1 8 9 3 6
	Rata-rata	145	2.4	
2	204	206	59	2 9 7 10 8 1 3 5 4 6
		205	104	9 7 10 8 1 3 5 4 6 2
		206	43	7 9 1 10 8 3 5 4 6 2
		206	84	7 9 2 10 8 1 3 5 4 6
		205	178	9 7 10 8 3 1 5 4 6 2
	Rata-rata	205.6	93.6	
3	202	212	21	1 7 3 8 10 5 2 9 4 6
		211	105	5 8 1 7 10 6 9 2 4 3
		211	70	5 8 1 7 10 6 9 2 4 3
		210	24	7 5 3 8 1 10 9 2 4 6
		210	14	5 3 7 8 1 10 9 2 4 6
	Rata-rata	210.8	46.8	
4	166	171	18	6 8 7 5 9 4 1 3 10 2
		171	118	6 8 7 4 5 9 1 3 10 2
		171	78	6 8 7 5 9 4 1 3 10 2
		171	25	7 6 8 9 4 5 1 3 10 2
		171	35	9 6 7 1 3 5 4 10 2 8
	Rata-rata	171	54.8	
5	188	192	49	8 10 6 2 3 5 9 1 4 7
		192	43	10 8 6 2 3 5 9 1 4 7
		192	88	8 10 2 9 5 3 6 1 4 7
		192	52	10 2 8 9 5 3 6 4 1 7
		192	39	10 2 6 5 3 8 9 1 4 7
	Rata-rata	192	54.2	

Tabel A.8: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 2 proses, 3 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	197	206	13	1 10 3 5 6 8 4 9 2 7
		207	49	6 1 3 5 10 4 9 8 2 7
		206	36	6 1 3 5 10 8 4 9 2 7
		206	108	10 1 3 5 6 8 4 9 2 7
		206	60	8 3 6 5 10 1 4 9 2 7
	Rata-rata	206.2	53.2	
7	220	228	13	4 2 3 5 8 7 1 6 10 9
		227	45	8 2 3 7 5 6 4 1 9 10
		229	11	7 3 2 4 5 6 8 1 9 10
		229	19	2 4 3 5 8 7 6 1 9 10
		229	15	2 3 7 4 5 9 8 1 6 10
	Rata-rata	228.4	20.6	
8	199	203	1	9 8 7 3 10 1 5 6 2 4
		203	6	9 8 7 3 1 5 6 10 2 4
		203	3	8 7 9 3 1 4 5 6 2 10
		203	2	10 7 9 5 6 3 1 8 2 4
		203	3	8 7 9 5 1 3 10 6 2 4
	Rata-rata	203	3	
9	224	228	30	9 8 5 10 3 7 4 2 1 6
		227	58	6 7 5 8 9 10 3 4 2 1
		227	27	7 6 5 8 10 9 3 4 2 1
		228	22	8 5 7 9 10 3 4 2 1 6
		230	9	8 3 7 6 10 9 5 4 2 1
	Rata-rata	228	29.2	
10	167	178	13	6 3 7 5 4 9 2 8 10 1
		178	11	10 4 7 9 8 5 2 3 6 1
		178	25	5 6 10 4 9 2 7 8 3 1
		178	34	10 6 4 9 7 5 8 2 3 1
		178	18	10 7 6 5 8 9 2 3 4 1
	Rata-rata	178	20.2	

Tabel A.9: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 2 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	304	336	182	4 2 5 10 3 7 9 1 8 6
		336	128	4 2 5 10 3 7 9 1 8 6
		336	117	4 3 9 2 5 10 7 8 1 6
		332	127	2 9 4 5 10 3 8 7 1 6
		332	32	2 4 9 5 10 3 7 8 1 6
	Rata-rata	334.4	117.2	
2	357	400	182	5 2 10 1 7 9 3 6 8 4
		398	97	10 7 2 5 6 9 8 3 1 4
		401	32	10 7 2 3 5 9 1 6 8 4
		397	13	10 5 2 3 7 9 1 6 8 4
		397	144	10 2 7 1 5 9 3 8 6 4
	Rata-rata	398.6	93.6	
3	405	425	14	9 10 2 5 3 4 6 8 7 1
		426	48	3 6 5 10 9 4 2 8 7 1
		427	168	6 5 9 3 10 4 1 2 8 7
		427	151	5 6 9 3 10 4 1 2 8 7
		430	74	5 3 10 6 9 4 2 8 7 1
	Rata-rata	427	91	
4	412	457	12	4 8 6 3 1 5 9 10 7 2
		454	50	4 6 8 3 1 5 9 7 2 10
		455	30	4 6 8 3 1 5 9 10 7 2
		449	20	4 6 3 1 8 9 5 10 7 2
		469	21	8 1 3 6 4 9 5 7 2 10
	Rata-rata	456.8	26.6	
5	357	403	65	4 5 6 3 9 8 1 10 7 2
		404	165	5 10 9 3 6 4 8 1 2 7
		404	106	10 5 9 3 6 4 8 1 2 7
		402	73	4 6 3 9 8 1 5 7 2 10
		399	5	6 2 4 3 8 10 1 5 7 9
	Rata-rata	402.4	82.8	

Tabel A.10: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 2 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	361	393	23	1 7 4 10 3 9 8 2 5 6
		400	67	1 4 3 7 10 9 8 2 5 6
		392	38	4 3 9 10 1 7 2 5 8 6
		393	56	4 3 9 10 1 7 5 2 8 6
		402	48	4 3 7 1 2 5 9 8 6 10
	Rata-rata	396	46.4	
7	351	384	45	5 1 3 6 10 7 9 4 8 2
		389	80	7 10 4 3 9 5 6 1 8 2
		392	59	7 10 4 3 9 6 8 5 1 2
		377	81	7 10 4 3 6 9 5 8 1 2
		384	191	5 1 3 6 10 7 9 4 8 2
	Rata-rata	385.2	91.2	
8	420	440	94	5 9 2 1 10 6 8 3 7 4
		437	34	2 5 3 10 1 8 6 9 4 7
		434	160	2 5 4 3 10 8 1 6 9 7
		440	193	1 2 5 3 6 8 10 4 9 7
		429	63	5 2 3 8 10 9 1 6 4 7
	Rata-rata	436	108.8	
9	414	458	91	9 1 5 7 3 8 6 10 2 4
		457	6	6 1 2 7 3 5 4 9 8 10
		458	23	3 5 1 8 7 6 10 2 9 4
		453	34	3 1 5 8 6 4 7 9 2 10
		451	3	1 2 9 7 5 3 6 8 4 10
	Rata-rata	455.4	31.4	
10	308	324	89	8 6 1 10 9 4 7 5 3 2
		335	143	8 6 4 1 10 9 7 5 2 3
		331	195	1 8 10 9 4 7 5 6 3 2
		328	162	1 8 10 9 7 4 6 5 3 2
		331	95	1 8 10 9 4 7 5 6 3 2
	Rata-rata	329.8	136.8	

Tabel A.11: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 2 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	304	332	36	9 2 4 5 10 3 7 8 1 6
		332	25	2 9 4 5 10 3 7 8 1 6
		332	79	9 2 4 5 10 3 7 8 1 6
		332	6	2 9 4 5 10 3 7 8 1 6
		332	54	2 9 4 5 10 3 7 8 1 6
	Rata-rata	332	40	
2	357	399	14	10 2 5 3 7 9 1 6 8 4
		397	14	5 10 2 3 7 9 1 6 8 4
		397	14	5 10 2 3 7 9 1 6 4 8
		397	8	10 2 7 1 5 9 3 8 4 6
		397	11	10 3 5 2 7 9 1 6 8 4
	Rata-rata	397.4	12.2	
3	405	423	45	6 9 3 10 4 5 2 8 7 1
		423	54	3 6 9 4 2 10 5 8 7 1
		426	26	6 9 10 2 5 3 8 4 7 1
		423	54	3 6 9 4 2 10 5 8 7 1
		419	14	5 6 9 4 2 3 10 8 7 1
	Rata-rata	422.8	38.6	
4	412	449	40	6 4 3 1 8 9 5 10 7 2
		449	34	4 6 3 1 8 9 5 10 7 2
		451	34	6 4 3 1 8 9 5 7 10 2
		453	3	1 4 6 3 8 9 5 7 10 2
		449	73	6 4 3 1 8 9 5 10 7 2
	Rata-rata	450.2	36.8	
5	357	397	23	10 6 4 5 2 3 8 1 9 7
		399	54	5 6 4 3 8 10 1 2 7 9
		399	9	5 6 4 3 8 10 1 2 7 9
		399	11	9 6 4 5 2 3 8 1 10 7
		395	39	6 4 5 10 3 2 8 1 7 9
	Rata-rata	397.8	27.2	

Tabel A.12: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 2 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	361	392	60	4 3 9 10 1 7 2 5 8 6
		392	82	4 3 9 10 1 7 2 5 8 6
		391	51	4 3 10 9 1 7 2 5 8 6
		391	37	4 1 7 10 3 9 5 8 2 6
		391	26	4 1 7 10 3 9 5 8 2 6
	Rata-rata	391.4	51.2	
7	351	377	57	7 10 4 3 6 9 5 8 1 2
		383	105	7 10 4 3 9 5 6 8 1 2
		377	70	7 10 4 3 6 9 5 8 1 2
		377	66	7 10 4 3 6 9 5 8 1 2
		377	2	10 7 4 3 6 9 5 8 1 2
	Rata-rata	378.2	60	
8	420	429	2	2 5 3 8 10 9 1 6 7 4
		434	33	5 2 3 4 10 8 1 6 9 7
		434	22	5 2 4 3 10 1 8 6 9 7
		435	36	2 3 5 9 10 1 8 6 7 4
		429	69	5 2 3 8 10 9 1 6 7 4
	Rata-rata	432.2	32.4	
9	414	455	19	7 2 1 9 5 3 6 8 4 10
		452	39	3 1 7 6 9 5 4 2 8 10
		451	7	2 1 7 9 3 5 8 6 4 10
		453	36	3 1 5 8 6 7 4 9 2 10
		453	15	2 1 6 7 3 5 9 4 8 10
	Rata-rata	452.8	23.2	
10	308	324	125	8 1 6 10 9 4 7 5 3 2
		324	20	6 8 1 10 9 4 7 5 3 2
		324	83	8 6 1 10 9 4 7 5 3 2
		324	21	8 1 6 10 9 4 7 5 3 2
		324	189	8 6 1 10 9 4 7 5 3 2
	Rata-rata	324	87.6	

Tabel A.13: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 3 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	260	270	11	5 10 3 6 1 4 2 8 7 9
		269	34	6 5 3 4 1 2 8 10 9 7
		269	24	1 6 10 3 5 4 2 8 7 9
		270	189	1 6 3 10 4 8 2 7 5 9
		270	27	5 10 3 6 1 4 8 2 7 9
	Rata-rata	269.6	57	
2	383	402	33	7 6 3 4 8 5 1 2 10 9
		401	18	4 3 1 7 9 5 2 6 8 10
		403	10	4 3 7 5 9 1 2 6 8 10
		401	22	7 1 8 3 4 5 2 6 9 10
		403	38	4 3 6 7 1 5 2 8 9 10
	Rata-rata	402	24.2	
3	295	315	63	7 3 5 6 10 4 1 8 9 2
		310	27	4 6 5 1 7 10 3 8 9 2
		305	26	7 4 6 10 5 3 1 8 9 2
		306	30	5 6 10 3 7 4 1 9 8 2
		304	5	10 5 6 3 4 7 1 8 9 2
	Rata-rata	308	30.2	
4	357	365	4	2 1 5 4 9 6 10 7 8 3
		365	4	5 9 4 6 2 1 7 8 3 10
		364	10	3 5 1 6 4 7 10 9 2 8
		361	22	2 1 4 5 6 7 3 10 9 8
		363	23	1 5 2 6 3 4 7 9 8 10
	Rata-rata	363.6	12.6	
5	298	298	74	3 8 2 9 1 5 7 4 10 6
		298	62	5 9 8 3 4 2 10 7 1 6
		298	63	5 9 8 2 3 4 10 6 7 1
		298	27	3 9 8 4 5 2 7 10 1 6
		298	25	2 3 8 9 5 4 1 10 6 7
	Rata-rata	298	50.2	

Tabel A.14: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 3 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	352	357	187	1 10 9 3 8 4 2 5 6 7
		359	152	3 9 10 8 4 7 1 2 5 6
		359	167	3 10 9 8 4 1 5 2 6 7
		356	198	10 1 9 4 8 3 6 2 5 7
		356	148	10 1 9 4 8 3 6 2 5 7
	Rata-rata	357.4	170.4	
7	319	319	52	6 4 9 2 8 7 10 5 1 3
		319	28	9 1 6 4 2 7 8 10 3 5
		320	20	1 6 8 4 7 3 2 9 10 5
		319	16	9 4 6 2 8 7 5 10 1 3
		320	44	1 6 8 4 7 9 2 5 10 3
	Rata-rata	319.4	32	
8	301	316	34	4 7 6 10 1 8 9 3 5 2
		317	14	9 1 8 4 10 6 7 2 3 5
		317	28	1 9 8 4 7 6 10 3 5 2
		317	64	6 8 9 4 7 10 1 3 2 5
		316	35	6 4 7 10 1 8 9 5 3 2
	Rata-rata	316.6	35	
9	367	367	108	7 1 6 10 4 3 2 5 9 8
		369	98	7 1 10 6 4 3 2 9 5 8
		367	109	7 1 6 10 4 3 2 9 5 8
		367	92	7 1 6 10 4 3 2 5 9 8
		367	19	7 6 1 10 4 3 2 9 5 8
	Rata-rata	367.4	85.2	
10	329	342	162	3 10 6 5 8 7 2 1 4 9
		345	198	3 10 2 5 7 9 1 8 4 6
		346	59	5 8 3 10 1 7 2 4 9 6
		342	15	6 10 3 5 8 7 2 1 4 9
		342	125	5 3 6 8 7 2 10 1 4 9
	Rata-rata	343.4	111.8	

Tabel A.15: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 3 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	260	267	15	1 10 6 3 4 5 8 2 7 9
		269	15	6 3 5 4 1 2 8 10 9 7
		267	8	1 10 6 3 4 5 2 8 7 9
		269	11	10 3 6 1 5 4 2 8 7 9
		267	15	10 1 6 3 4 5 2 8 7 9
	Rata-rata	267.8	12.8	
2	383	400	32	7 1 4 6 8 5 2 3 9 10
		399	24	1 4 7 6 8 5 2 3 10 9
		399	8	7 1 4 6 8 5 2 3 10 9
		399	9	1 7 6 8 5 4 2 3 10 9
		401	11	9 7 1 3 5 4 2 6 10 8
	Rata-rata	399.6	16.8	
3	295	304	50	10 6 7 3 5 4 1 8 9 2
		304	13	6 5 10 3 7 4 1 8 9 2
		304	12	6 10 5 3 4 7 1 8 9 2
		305	12	4 7 6 10 5 3 1 8 9 2
		304	8	5 4 6 10 7 3 1 8 9 2
	Rata-rata	304.2	19	
4	357	359	9	3 1 9 5 2 6 4 7 8 10
		359	11	1 3 2 9 5 6 4 7 8 10
		359	5	3 1 9 2 5 6 4 7 8 10
		359	11	3 1 9 5 2 6 4 7 8 10
		359	5	1 2 3 9 5 6 4 7 8 10
	Rata-rata	359	8.2	
5	298	298	23	2 8 3 9 1 5 7 4 10 6
		298	2	8 1 9 2 3 5 4 6 10 7
		298	2	8 9 3 2 4 10 5 1 7 6
		298	14	9 1 8 3 5 2 4 10 7 6
		298	7	3 8 2 9 4 1 5 10 6 7
	Rata-rata	298	9.6	

Tabel A.16: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 10 pekerjaan, 4 proses, 3 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	352	356	28	10 9 1 4 8 3 6 2 5 7
		357	81	9 1 10 3 8 4 2 5 6 7
		356	76	9 10 1 4 8 3 6 2 5 7
		356	166	10 1 9 4 8 3 6 2 5 7
		356	171	10 1 9 4 8 3 6 2 5 7
	Rata-rata	356.2	104.4	
7	319	319	4	6 4 1 8 7 2 3 10 9 5
		319	9	4 1 6 8 2 7 10 9 3 5
		319	5	1 6 4 8 7 2 10 9 5 3
		319	22	1 6 9 4 7 8 2 5 10 3
		319	16	9 1 6 4 2 8 7 10 5 3
	Rata-rata	319	11.2	
8	301	317	4	8 6 9 4 7 10 3 2 5 1
		317	19	9 4 10 8 7 6 3 5 1 2
		316	5	4 6 8 7 10 1 5 9 3 2
		316	66	6 8 4 7 10 1 5 9 3 2
		316	122	4 7 6 10 1 8 9 5 3 2
	Rata-rata	316.4	43.2	
9	367	367	44	7 1 6 10 4 3 2 5 9 8
		367	39	7 1 6 10 4 3 2 9 5 8
		367	62	7 1 6 10 4 3 2 5 9 8
		367	60	1 6 7 10 4 3 2 9 5 8
		367	77	1 6 7 10 4 3 2 5 9 8
	Rata-rata	367	56.4	
10	329	338	36	3 10 5 2 8 7 1 9 4 6
		342	171	6 5 3 8 7 2 10 1 4 9
		338	95	2 5 3 10 8 7 1 9 4 6
		338	148	5 3 10 2 8 7 1 9 4 6
		338	170	5 2 3 10 8 7 1 9 4 6
	Rata-rata	338.8	124	

LAMPIRAN B

DATA PENGUJIAN KASUS *HYBRID FLOW SHOP* DENGAN 20 PEKERJAAN

Tabel B.1: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 2 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	556	586	162	19 18 10 3 11 2 14 1 20 7 16 13 8 17 4 9 5 15 6 12
		591	53	7 19 20 8 5 13 1 17 3 6 15 14 9 16 10 2 12 18 4 11
		589	52	12 14 13 8 19 3 16 4 5 15 7 18 1 9 11 17 20 10 2 6
		584	174	14 20 12 8 4 1 11 6 17 13 18 15 10 3 19 5 9 2 16 7
		592	132	2 19 7 18 5 9 11 20 1 13 8 10 3 6 4 16 17 15 12 14
	Rata-rata	588.4	114.6	
2	493	545	125	16 14 18 10 12 9 7 17 6 5 15 20 3 11 2 1 4 19 8 13
		540	118	11 2 1 6 9 15 20 5 16 3 4 17 19 10 12 18 13 8 14 7
		541	169	14 3 11 16 7 20 5 6 10 12 17 2 19 4 9 15 1 8 18 13
		547	50	11 9 18 17 10 7 12 4 2 20 13 5 16 3 6 15 14 19 1 8
		544	199	11 2 18 14 17 12 19 16 4 3 10 15 1 9 8 6 5 7 20 13
	Rata-rata	543.4	132.2	
3	516	566	146	13 1 3 10 18 12 7 4 6 14 11 15 9 2 17 8 5 19 16 20
		566	100	18 6 12 17 3 14 2 15 5 7 8 4 13 19 10 9 1 11 16 20
		560	87	4 6 11 17 12 2 10 7 8 20 1 13 9 15 5 3 19 16 14 18
		562	45	13 3 2 17 14 4 12 7 10 6 18 11 15 9 19 8 20 1 16 5
		561	55	13 14 1 12 9 10 17 4 19 2 15 20 11 8 16 6 7 18 5 3
	Rata-rata	563	86.6	
4	534	555	129	13 19 10 4 8 11 15 14 6 5 2 20 1 3 18 16 7 12 9 17
		553	32	12 14 20 10 13 7 1 18 16 17 4 5 19 2 6 3 11 8 15 9
		546	62	18 13 11 9 1 19 7 5 20 10 12 16 6 15 17 14 4 2 8 3
		554	104	3 13 1 14 20 5 16 4 11 6 17 2 8 15 10 19 7 9 18 12
		549	135	13 7 16 4 15 8 2 17 18 11 10 6 12 3 5 1 20 14 9 19
	Rata-rata	551.4	92.4	
5	442	480	19	4 10 6 16 9 15 3 12 2 17 7 18 20 13 5 11 14 1 8 19
		478	184	5 12 20 9 17 19 13 4 8 15 11 3 16 1 6 10 7 14 18 2
		482	142	4 1 2 18 15 7 3 9 17 20 19 14 10 6 8 5 13 11 16 12
		479	17	16 5 4 1 6 15 14 2 17 9 11 7 19 20 18 13 10 8 3 12
		476	112	2 5 3 17 20 13 1 6 4 9 7 19 14 15 16 11 18 10 8 12
	Rata-rata	479	94.8	

Tabel B.2: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 2 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	662	678	179	4 9 8 14 6 17 7 10 11 13 2 18 1 15 12 19 16 20 5 3
		678	191	11 17 3 8 18 12 16 2 10 4 13 6 9 15 14 1 19 7 20 5
		675	79	17 8 18 14 2 9 11 12 7 10 15 16 1 19 13 6 3 4 20 5
		675	129	19 14 3 4 7 10 9 1 13 12 16 2 17 18 15 8 20 11 6 5
		680	149	19 7 3 13 14 6 16 18 11 15 9 20 17 8 12 10 2 1 4 5
	Rata-rata	677.2	145.4	
7	488	501	137	13 7 12 15 5 1 14 17 9 20 6 8 18 10 2 19 16 11 3 4
		491	113	9 3 12 14 4 1 16 6 10 2 11 19 17 7 8 18 20 15 5 13
		499	185	12 9 18 10 2 3 7 5 13 15 17 6 20 19 8 1 16 11 14 4
		495	90	7 9 5 19 2 12 1 17 18 10 8 3 6 14 16 11 15 13 20 4
		494	121	7 9 15 8 1 20 10 17 6 11 16 4 3 19 2 18 14 5 12 13
	Rata-rata	496	129.2	
8	533	560	86	8 10 4 1 7 16 17 14 3 11 5 9 2 15 12 18 20 13 19 6
		562	190	8 10 4 11 3 7 17 13 5 16 19 2 15 12 14 1 18 20 9 6
		567	169	12 11 8 10 9 14 20 19 5 2 3 17 7 4 15 18 13 1 16 6
		562	174	8 15 4 10 3 7 9 16 17 5 14 1 2 11 12 18 20 13 19 6
		561	199	4 7 8 10 14 15 9 20 16 19 17 5 12 3 2 18 11 1 13 6
	Rata-rata	562.4	163.6	
9	494	538	158	1 5 12 17 10 18 13 2 3 9 7 19 6 8 16 20 11 15 14 4
		534	72	17 6 18 9 1 3 15 11 2 5 13 19 16 20 12 7 10 8 4 14
		533	22	2 17 12 19 8 13 16 15 11 6 20 9 1 4 5 14 3 18 10 7
		537	109	17 13 18 6 9 8 5 10 12 1 11 4 15 3 14 20 16 2 19 7
		537	100	1 18 3 7 4 9 5 8 12 17 6 20 14 19 16 13 11 2 10 15
	Rata-rata	535.8	92.2	
10	636	639	96	18 6 7 13 12 15 3 5 20 2 1 10 4 14 11 8 16 17 19 9
		639	64	14 17 20 7 13 4 2 15 8 16 10 12 19 9 18 11 3 5 1 6
		640	55	7 14 16 13 4 11 20 2 12 19 8 10 18 17 6 15 3 5 1 9
		640	118	11 3 13 1 2 8 15 6 5 16 9 14 10 7 20 18 4 12 17 19
		638	93	8 14 16 10 17 18 15 19 7 20 9 6 12 11 4 2 3 5 13 1
	Rata-rata	639.2	85.2	

Tabel B.3: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 2 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	556	574	195	12 19 11 5 16 4 17 2 18 20 7 8 9 3 14 10 1 13 15 6
		580	182	19 14 5 20 6 13 8 1 3 11 9 12 10 2 18 17 7 16 15 4
		583	17	14 10 12 8 17 9 13 1 2 6 11 5 18 4 20 16 3 19 15 7
		582	64	18 17 14 6 1 2 19 13 8 4 9 11 5 3 16 10 12 20 15 7
		577	154	12 19 11 13 16 5 8 17 2 20 7 1 14 6 10 3 18 9 15 4
	Rata-rata	579.2	122.4	
2	493	534	126	11 2 18 10 19 3 1 17 12 7 5 6 16 20 9 15 14 4 8 13
		534	72	13 11 2 5 9 16 3 19 17 12 1 20 18 8 10 15 14 4 6 7
		528	111	11 15 1 7 5 19 16 20 17 18 9 13 4 10 12 2 14 3 6 8
		517	167	14 11 2 20 18 13 9 12 17 16 19 3 1 4 6 10 15 5 8 7
		529	66	11 16 14 19 17 15 1 7 2 5 9 12 3 13 6 20 4 10 18 8
	Rata-rata	528.4	108.4	
3	516	556	126	13 12 10 6 18 3 7 8 14 15 20 19 4 1 17 5 11 2 16 9
		555	57	13 12 18 3 11 14 17 19 10 5 2 15 20 7 6 8 9 1 4 16
		555	131	1 12 17 14 2 15 20 19 8 9 5 7 13 6 11 10 4 16 3 18
		554	13	5 13 14 11 8 3 20 6 4 12 1 19 17 15 7 18 10 2 16 9
		555	28	1 6 14 4 11 10 20 5 12 15 13 3 19 2 17 8 7 16 9 18
	Rata-rata	555	71	
4	534	548	176	19 4 14 17 13 9 2 8 16 6 5 7 3 10 1 18 12 15 11 20
		546	85	13 4 14 19 1 2 6 16 8 11 7 18 3 15 5 10 20 12 9 17
		541	92	14 13 20 5 10 17 19 7 2 15 6 18 12 1 11 4 16 8 9 3
		540	61	5 14 7 11 4 10 9 15 1 3 13 2 6 19 20 8 16 17 12 18
		544	28	14 7 18 13 9 3 5 10 16 6 20 12 1 19 11 4 15 2 8 17
	Rata-rata	543.8	88.4	
5	442	475	145	15 12 3 2 11 5 18 17 20 13 4 9 10 7 8 1 6 14 19 16
		473	179	2 4 14 15 3 8 10 13 5 17 20 9 11 7 19 6 1 18 16 12
		475	96	15 20 14 3 11 9 5 18 4 7 10 6 16 13 12 2 1 19 8 17
		474	186	1 5 10 18 11 14 4 6 19 15 7 8 12 20 9 13 3 16 17 2
		475	85	14 5 10 17 11 6 7 9 3 8 2 12 15 1 18 20 13 4 16 19
	Rata-rata	474.4	138.2	

Tabel B.4: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 2 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	662	671	163	4 11 16 14 8 2 12 15 6 17 9 1 20 13 10 18 7 5 19 3
		674	35	4 10 14 2 12 7 18 9 8 20 15 17 11 6 16 13 1 19 5 3
		671	178	3 4 11 1 17 10 18 15 12 8 2 9 6 19 13 14 7 16 20 5
		673	41	16 4 17 2 9 14 15 18 12 1 13 6 10 19 7 11 8 20 5 3
		671	60	3 4 12 16 1 7 20 17 14 19 13 18 9 10 15 2 8 6 11 5
	Rata-rata	672	95.4	
7	488	495	52	9 14 7 15 17 13 12 10 8 1 19 2 20 18 6 5 16 3 11 4
		495	45	9 10 7 17 6 8 20 13 19 11 5 1 2 18 3 12 14 15 16 4
		491	50	9 7 5 10 2 6 11 19 17 18 14 16 8 1 4 12 15 20 13 3
		494	125	2 9 7 15 6 5 3 1 8 11 13 20 16 10 17 12 18 4 19 14
		491	176	9 6 3 1 13 12 15 2 7 10 8 5 19 17 18 14 11 16 4 20
	Rata-rata	493.2	89.6	
8	533	559	187	8 10 4 7 11 3 13 19 17 20 16 6 2 15 12 14 1 18 9 5
		559	165	16 17 20 2 15 12 14 1 18 9 19 6 8 10 4 7 11 3 13 5
		563	151	8 10 4 7 1 18 11 6 17 9 20 19 2 15 16 12 14 3 13 5
		561	172	8 10 4 7 11 9 17 3 14 13 5 2 15 12 18 1 6 20 16 19
		562	123	8 10 16 17 11 3 13 5 2 15 19 4 1 18 7 12 14 20 9 6
	Rata-rata	560.8	159.6	
9	494	526	168	1 17 3 10 16 15 13 8 2 12 11 4 6 9 5 20 19 18 7 14
		522	71	1 17 15 11 2 12 18 16 8 6 9 3 13 14 19 20 4 5 10 7
		519	120	1 19 16 20 18 15 14 5 13 8 4 9 17 3 6 12 11 2 10 7
		527	181	1 8 16 13 11 19 4 6 3 12 18 15 2 9 5 20 14 10 17 7
		527	178	1 17 20 8 3 6 9 16 13 2 10 14 11 5 12 18 15 19 4 7
	Rata-rata	524.2	143.6	
10	636	637	113	3 14 4 15 11 18 8 9 12 6 10 7 20 16 17 19 2 5 13 1
		636	192	10 6 2 18 19 11 4 14 12 8 7 20 17 15 13 16 3 5 1 9
		637	80	15 14 20 11 17 18 4 16 7 5 6 2 3 10 8 12 13 19 1 9
		637	60	3 14 7 8 4 5 19 12 20 6 16 18 13 10 15 2 11 17 1 9
		637	174	3 19 14 7 18 11 8 12 17 20 16 15 5 4 9 2 10 13 1 6
	Rata-rata	636.8	123.8	

Tabel B.5: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 3 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	366	395	150	3 17 14 9 12 18 15 16 6 5 10 8 7 11 2 13 19 20 4 1
		388	122	2 17 8 9 20 13 10 6 19 11 7 16 12 15 14 18 4 3 1 5
		385	191	9 4 14 17 3 6 12 10 7 2 16 11 18 13 8 5 15 20 1 19
		393	109	10 3 9 6 17 14 8 13 2 4 15 19 12 20 7 11 1 16 5 18
		393	135	9 12 17 2 20 16 7 10 18 13 6 14 8 4 15 11 19 1 5 3
	Rata-rata	390.8	141.4	
2	352	367	164	5 1 16 11 14 9 8 15 10 13 3 4 2 19 18 7 17 20 12 6
		363	87	11 4 6 14 10 12 5 7 15 16 13 3 18 1 19 2 8 20 17 9
		364	7	11 4 5 12 1 7 18 13 14 16 19 3 6 8 15 2 10 20 17 9
		362	81	11 14 7 16 5 20 12 8 15 18 3 6 4 19 10 2 1 17 13 9
		367	73	3 8 15 14 4 6 1 16 9 20 12 10 11 19 7 18 2 17 13 5
	Rata-rata	364.6	82.4	
3	352	370	72	9 6 16 13 18 8 11 1 19 14 10 20 15 4 7 3 5 2 12 17
		370	171	18 20 9 12 19 6 7 11 3 15 10 14 8 4 13 16 17 1 5 2
		379	135	12 9 10 16 6 14 3 11 8 13 17 5 4 15 7 2 18 20 1 19
		375	190	10 9 6 20 13 11 14 8 15 7 5 16 12 4 1 18 3 19 2 17
		379	183	12 20 9 10 11 3 18 17 19 14 8 13 1 15 4 5 16 7 2 6
	Rata-rata	374.6	150.2	
4	336	363	113	5 16 11 20 3 14 7 2 18 1 15 17 6 4 13 19 10 12 8 9
		362	165	11 14 1 19 12 6 4 5 13 3 20 16 9 7 15 18 10 17 2 8
		365	161	11 5 20 6 4 17 12 15 16 7 1 13 3 19 14 2 10 8 9 18
		362	27	6 16 2 1 14 11 4 5 10 13 3 7 17 20 18 19 15 12 8 9
		366	122	13 3 8 1 19 20 14 11 12 5 7 4 16 18 6 10 15 17 2 9
	Rata-rata	363.6	117.6	
5	364	386	56	4 8 3 12 7 18 9 17 14 2 10 6 11 20 16 15 1 5 13 19
		379	83	11 3 4 10 13 12 20 17 15 19 8 14 2 9 18 16 1 5 6 7
		388	84	4 14 12 3 20 17 8 10 19 15 1 13 11 16 18 7 2 5 9 6
		387	150	3 14 4 10 12 2 8 17 18 13 15 5 16 1 11 9 20 6 7 19
		383	62	4 12 3 18 16 6 20 10 11 2 17 1 15 7 5 14 8 9 13 19
	Rata-rata	384.6	87	

Tabel B.6: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 3 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	381	392	164	7 2 1 18 5 9 6 19 13 4 14 17 8 11 20 10 16 15 12 3
		395	170	8 18 9 13 10 7 1 3 5 19 2 14 20 4 17 6 15 11 16 12
		393	168	14 3 2 9 8 6 20 4 17 5 18 1 13 7 10 15 11 19 16 12
		384	157	11 15 18 1 13 14 3 17 9 8 5 2 4 20 6 10 16 19 12 7
		392	40	6 20 3 18 9 5 14 1 17 8 11 10 13 4 2 16 15 12 7 19
	Rata-rata	391.2	139.8	
7	365	381	167	13 6 9 2 10 8 12 11 15 18 19 16 20 1 5 4 7 17 3 14
		383	43	18 13 14 11 20 9 10 7 17 5 4 16 1 15 19 12 2 8 6 3
		382	112	12 9 4 14 11 15 2 5 19 7 18 20 1 8 13 10 16 17 6 3
		384	94	13 7 3 11 10 16 18 9 15 20 8 14 19 1 2 5 12 17 6 4
		381	184	6 4 12 11 10 18 13 7 16 19 2 8 1 15 20 14 9 17 5 3
	Rata-rata	382.2	120	
8	345	387	170	15 19 13 7 12 11 1 2 10 6 20 5 16 3 18 9 14 8 17 4
		379	7	15 13 12 7 11 2 20 5 4 18 17 10 14 16 8 3 1 19 6 9
		387	153	11 3 17 20 18 14 15 16 13 5 19 4 1 12 10 2 6 8 9 7
		382	109	13 7 2 19 11 4 3 1 15 10 5 9 20 18 16 12 8 14 6 17
		379	84	11 20 15 13 12 4 3 18 1 16 19 5 10 14 6 7 8 2 9 17
	Rata-rata	382.8	104.6	
9	421	432	97	7 8 11 13 3 14 5 6 1 18 4 12 9 17 16 10 15 20 2 19
		432	167	7 19 13 6 12 14 3 17 10 15 5 4 8 20 11 16 18 9 2 1
		434	85	19 5 13 4 8 20 9 1 12 3 7 16 11 14 15 6 10 18 17 2
		432	98	5 7 1 13 4 16 6 20 9 19 17 3 8 14 15 12 11 18 10 2
		433	87	16 13 11 3 15 1 17 19 12 4 20 8 7 2 18 6 5 14 10 9
	Rata-rata	432.6	106.8	
10	377	397	106	15 11 4 19 14 6 16 9 8 20 7 5 3 18 17 12 2 10 13 1
		400	27	19 14 15 13 6 5 20 9 17 4 11 2 18 16 12 10 7 8 3 1
		402	134	8 16 6 3 15 18 17 12 4 20 9 2 19 10 5 13 7 11 14 1
		398	89	4 5 15 19 17 11 2 13 18 7 20 9 3 12 10 14 16 8 6 1
		394	179	6 15 3 11 12 2 4 13 20 9 17 16 5 8 18 7 19 10 1 14
	Rata-rata	398.2	107	

Tabel B.7: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 3 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	366	386	142	3 4 17 2 9 20 11 14 13 8 6 15 18 10 7 19 12 16 1 5
		386	92	20 3 16 9 7 2 13 4 6 5 8 14 17 15 10 11 12 1 18 19
		385	4	17 4 3 6 9 2 20 7 8 15 12 10 5 14 11 16 13 1 19 18
		383	91	3 18 17 2 9 6 20 14 7 8 16 4 10 15 11 13 12 1 5 19
		385	21	10 11 9 3 17 19 4 15 7 2 14 16 13 6 8 12 20 1 18 5
	Rata-rata	385	70	
2	352	358	36	11 14 2 5 7 15 3 4 19 12 8 16 10 18 20 6 1 17 9 13
		362	84	10 6 11 8 16 1 9 7 15 19 5 3 4 2 18 20 12 17 13 14
		362	118	8 14 11 4 16 7 15 6 5 2 19 18 10 1 20 12 17 3 9 13
		361	185	4 2 3 1 15 12 20 11 9 7 16 19 14 5 18 8 17 10 13 6
		362	25	12 11 17 8 15 7 18 3 16 2 5 19 20 4 10 1 9 13 14 6
	Rata-rata	361	89.6	
3	352	361	166	12 7 9 6 15 3 18 19 8 11 10 1 4 5 14 20 13 16 17 2
		368	154	6 7 16 9 12 15 3 11 4 10 19 1 18 8 13 14 20 5 2 17
		365	179	13 16 9 4 15 12 14 17 6 1 11 7 10 5 8 3 18 20 19 2
		366	127	15 13 17 9 14 6 20 18 2 10 11 8 5 4 12 19 3 7 1 16
		369	118	16 9 12 4 18 8 19 7 20 3 5 11 15 17 13 10 14 1 6 2
	Rata-rata	365.8	148.8	
4	336	359	147	11 8 1 17 19 4 12 6 7 14 5 20 13 15 16 10 18 2 3 9
		349	31	15 2 1 11 5 17 13 6 14 4 20 3 7 16 19 10 12 8 9 18
		355	173	11 14 12 3 6 4 7 16 2 9 19 13 5 1 20 17 10 15 8 18
		357	137	12 11 13 7 14 17 6 4 3 15 5 2 19 16 10 20 8 18 1 9
		357	170	11 4 3 5 19 16 20 10 13 18 6 14 2 7 12 17 15 1 8 9
	Rata-rata	355.4	131.6	
5	364	381	123	12 20 13 15 17 4 2 5 9 10 11 16 19 8 14 3 18 6 1 7
		379	166	12 3 20 10 17 4 18 11 14 16 1 7 9 2 19 8 15 5 6 13
		376	69	4 10 20 14 12 3 8 17 18 16 13 5 15 6 7 9 2 11 1 19
		376	60	3 2 4 10 12 20 17 5 14 8 15 16 19 9 18 7 11 13 1 6
		378	155	4 14 13 3 12 20 10 2 16 7 5 15 17 9 18 8 11 19 1 6
	Rata-rata	378	114.6	

Tabel B.8: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 2 proses, 3 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	381	388	67	17 6 19 3 1 20 8 13 5 2 11 4 9 14 10 15 18 16 7 12
		389	45	1 19 2 14 3 8 9 15 5 13 10 11 17 18 4 6 20 16 7 12
		390	7	20 6 9 7 3 14 8 4 17 13 11 5 10 15 18 16 2 12 19 1
		388	104	1 20 8 17 19 14 11 7 15 10 18 5 9 4 13 6 2 16 12 3
		387	37	20 11 5 9 10 17 4 8 15 19 13 14 3 6 16 18 12 2 7 1
	Rata-rata	388.4	52	
7	365	374	122	9 13 7 5 12 11 10 8 17 1 2 15 18 20 16 4 19 6 3 14
		378	29	13 5 12 9 15 14 8 16 2 11 20 18 19 10 4 1 7 17 6 3
		374	182	9 8 13 7 10 19 16 5 12 11 15 17 1 18 20 2 6 3 14 4
		379	60	4 5 13 14 7 16 19 18 8 15 11 20 1 2 10 17 9 6 3 12
		377	66	4 19 13 11 15 12 5 16 8 2 14 7 20 1 18 10 17 9 6 3
	Rata-rata	376.4	91.8	
8	345	374	49	20 19 11 2 6 4 15 13 1 16 3 18 7 10 5 14 12 8 17 9
		377	99	2 3 11 20 18 17 6 12 4 13 5 16 1 14 10 15 19 8 9 7
		377	180	11 20 15 2 12 17 4 3 5 1 16 8 18 6 10 14 19 13 9 7
		378	161	20 5 15 1 7 2 6 17 3 16 11 13 18 19 12 14 10 8 9 4
		370	41	20 11 15 13 10 5 19 16 17 12 7 8 18 2 1 14 3 6 9 4
	Rata-rata	375.2	106	
9	421	425	179	18 8 4 13 7 5 17 14 16 19 1 6 15 12 11 3 20 10 9 2
		426	162	15 5 16 14 18 6 17 3 13 11 4 19 20 7 8 1 12 10 9 2
		426	80	4 16 17 15 20 11 19 14 8 7 18 6 5 9 3 12 10 13 2 1
		427	165	18 10 8 6 4 16 20 3 15 5 11 12 14 17 7 19 13 9 2 1
		426	57	15 4 16 18 9 8 17 5 14 3 11 13 6 20 7 19 12 10 2 1
	Rata-rata	426	128.6	
10	377	394	98	3 6 19 4 14 20 9 15 7 5 17 16 2 11 18 1 12 10 8 13
		398	56	14 18 6 5 3 20 7 16 2 19 9 17 15 13 12 11 10 8 4 1
		395	190	19 4 6 15 7 20 9 14 11 12 5 10 18 17 16 2 3 8 13 1
		396	100	6 4 15 18 7 19 5 2 20 17 9 11 10 16 13 12 8 3 14 1
		393	50	16 5 6 20 17 11 9 2 8 4 14 19 18 13 10 12 7 15 3 1
	Rata-rata	395.2	98.8	

Tabel B.9: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 2 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	600	708	28	9 15 16 17 20 14 2 6 5 12 10 4 3 13 8 18 1 7 19 11
		701	28	13 14 16 1 19 3 11 6 7 12 20 9 4 5 15 18 17 8 2 10
		683	190	5 16 14 7 12 13 19 8 17 18 3 6 1 4 15 9 2 20 11 10
		696	108	14 16 4 12 7 17 15 13 1 10 5 3 8 18 6 20 9 19 2 11
		704	186	7 14 1 16 17 20 3 6 4 15 9 8 18 12 10 5 2 13 19 11
	Rata-rata	698.4	108	
2	811	814	3	17 19 18 7 4 2 11 8 5 14 15 6 16 3 10 13 20 9 1 12
		815	147	3 14 18 16 19 11 9 20 8 4 5 1 13 10 17 6 15 12 7 2
		813	58	18 19 7 9 14 1 4 16 5 20 13 15 3 10 6 17 2 12 8 11
		815	125	7 17 18 19 15 10 16 12 20 11 8 4 6 14 1 9 13 5 2 3
		813	172	19 18 9 7 10 16 17 8 20 5 12 4 2 14 15 6 11 13 1 3
	Rata-rata	814	101	
3	592	652	118	20 17 14 4 18 3 9 13 8 19 12 10 15 11 1 16 5 2 7 6
		654	33	10 18 11 19 13 17 15 8 1 4 6 3 16 9 20 2 14 5 7 12
		645	103	11 12 15 10 2 3 19 13 8 9 20 5 16 18 4 14 6 1 7 17
		644	102	10 12 20 11 14 4 8 2 1 16 9 13 3 5 18 7 19 17 6 15
		645	17	10 13 18 4 20 14 11 6 3 5 16 9 17 8 19 15 1 2 12 7
	Rata-rata	648	74.6	
4	711	731	50	14 3 2 9 1 8 13 16 5 6 15 11 18 10 20 17 12 4 7 19
		729	141	4 16 3 1 10 18 11 20 13 8 14 9 12 7 6 15 19 17 2 5
		738	106	8 3 15 12 5 9 2 6 4 14 20 16 18 10 1 13 7 19 17 11
		728	145	3 1 10 20 7 9 13 16 15 12 8 18 6 5 14 2 19 4 17 11
		735	157	20 19 3 1 10 5 6 9 2 16 15 13 8 18 12 14 7 4 17 11
	Rata-rata	732.2	119.8	
5	552	634	25	11 10 3 8 4 12 17 9 5 18 13 14 6 16 1 15 20 2 7 19
		635	179	9 17 3 13 4 5 16 11 14 8 12 15 10 6 1 18 7 20 2 19
		636	50	15 7 18 11 9 8 4 5 12 1 6 20 14 10 2 13 3 16 17 19
		629	72	11 10 3 8 2 13 4 19 20 18 1 14 6 9 16 7 5 15 17 12
		626	113	9 3 8 16 11 14 1 13 4 15 6 18 10 17 20 2 7 12 5 19
	Rata-rata	632	87.8	

Tabel B.10: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 2 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	581	650	74	6 15 1 13 12 19 5 3 4 11 14 18 7 9 10 17 20 2 8 16
		650	158	1 2 20 14 12 13 17 16 11 5 6 15 4 18 9 8 10 3 19 7
		651	194	1 12 8 16 18 9 20 14 3 2 6 15 4 13 17 11 5 7 19 10
		642	152	1 3 16 6 20 18 5 17 12 2 11 9 14 15 4 13 7 8 19 10
		639	171	1 12 13 7 20 14 11 19 6 15 4 18 17 5 8 2 9 3 16 10
	Rata-rata	646.4	149.8	
7	695	731	189	17 4 16 13 20 10 12 1 7 19 9 5 15 2 8 6 11 18 14 3
		728	82	4 9 1 8 10 14 16 3 17 20 11 7 12 2 19 15 5 6 13 18
		723	152	1 19 14 16 12 7 5 9 4 15 2 8 6 18 20 13 3 17 11 10
		730	141	8 16 6 1 17 11 18 12 2 5 15 20 14 7 19 9 4 13 3 10
		734	188	4 3 14 10 19 15 1 5 9 2 8 11 18 12 16 7 6 13 20 17
	Rata-rata	729.2	150.4	
8	711	750	181	11 17 12 1 2 7 6 3 14 9 19 13 10 15 8 18 16 4 5 20
		737	51	18 16 11 8 6 13 2 15 12 1 17 4 3 10 9 19 7 14 5 20
		740	177	6 18 13 2 3 14 4 9 19 7 11 15 12 1 10 16 8 17 5 20
		742	184	18 16 6 17 3 14 12 19 7 11 1 2 15 5 10 4 13 9 8 20
		736	157	14 7 6 17 3 10 9 1 4 13 2 8 11 12 19 15 16 18 5 20
	Rata-rata	741	150	
9	662	748	188	15 4 14 18 19 3 7 2 16 20 13 8 12 1 9 17 5 10 11 6
		751	117	7 2 4 9 15 13 11 19 1 17 5 16 12 6 18 20 14 3 8 10
		752	126	7 4 14 11 9 15 19 3 8 2 13 18 12 20 1 6 16 17 10 5
		745	7	14 15 8 13 7 1 12 6 19 2 20 16 17 18 9 3 5 10 11 4
		754	170	4 14 16 12 20 6 2 19 11 18 17 10 7 9 15 13 8 3 1 5
	Rata-rata	750	121.6	
10	545	630	175	5 1 14 17 11 8 16 12 2 13 7 6 15 4 18 19 10 9 3 20
		628	121	1 18 7 20 11 16 5 2 14 19 10 8 17 4 13 6 12 3 9 15
		633	194	20 18 4 17 13 15 7 6 16 5 8 1 11 19 12 2 9 3 14 10
		640	65	1 11 20 4 14 7 12 17 5 15 18 19 6 16 13 2 8 3 9 10
		637	193	1 5 14 16 17 20 13 8 6 12 19 7 15 9 18 2 4 3 11 10
	Rata-rata	633.6	149.6	

Tabel B.11: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 2 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	600	679	121	7 9 8 12 14 4 1 3 17 5 13 18 16 15 6 19 20 10 2 11
		683	181	12 14 9 15 18 19 4 3 6 5 13 16 17 8 1 10 20 11 2 7
		687	132	14 12 2 7 5 13 18 16 3 6 4 15 8 9 1 17 10 20 19 11
		687	166	12 10 6 9 15 7 17 4 14 5 3 20 18 2 8 1 16 13 19 11
		673	119	3 16 6 5 12 14 8 4 17 1 7 13 18 19 15 9 20 2 10 11
	Rata-rata	681.8	143.8	
2	811	811	22	19 18 3 14 13 8 7 11 16 17 5 12 6 15 4 9 10 2 1 20
		812	18	19 18 11 10 7 9 1 3 14 6 5 12 2 4 13 20 17 8 16 15
		812	182	19 18 5 17 1 9 14 6 15 4 3 7 11 10 16 12 13 2 20 8
		811	117	18 19 6 14 4 2 11 13 10 7 1 15 20 8 16 3 5 9 17 12
		812	32	14 18 19 5 11 4 6 15 8 17 1 12 10 7 20 16 3 2 9 13
	Rata-rata	811.6	74.2	
3	592	635	163	17 20 4 9 10 13 1 14 3 19 16 18 2 8 5 6 15 11 7 12
		637	30	17 10 4 20 19 6 3 13 5 16 18 15 9 1 14 11 2 8 7 12
		637	68	17 15 10 20 9 13 11 19 12 4 3 2 1 16 18 5 8 14 6 7
		642	161	4 12 16 9 17 3 18 2 19 1 13 6 8 5 10 14 20 11 15 7
		637	163	4 17 1 16 10 2 15 8 19 20 18 13 3 9 14 11 5 6 7 12
	Rata-rata	637.6	117	
4	711	725	168	3 1 10 9 20 12 4 13 16 19 8 6 14 5 15 17 18 7 2 11
		728	176	16 2 20 12 3 15 6 5 8 1 10 18 13 4 14 19 7 9 17 11
		724	125	3 1 10 20 18 13 16 8 14 15 19 12 6 5 7 2 9 17 4 11
		722	155	9 15 14 2 3 16 5 19 1 10 4 7 20 12 18 13 8 6 17 11
		725	67	15 4 18 3 19 1 10 2 16 13 8 6 17 9 14 20 12 5 7 11
	Rata-rata	724.8	138.2	
5	552	607	30	11 9 5 1 6 13 4 3 18 17 14 10 16 8 20 15 2 7 19 12
		622	3	8 11 5 1 14 9 10 13 20 4 17 18 3 7 2 6 16 15 19 12
		621	69	9 11 3 6 18 1 13 4 2 8 20 14 7 10 15 17 16 5 19 12
		621	51	5 8 16 11 13 15 4 3 20 1 18 17 7 14 9 6 10 2 19 12
		611	181	8 9 18 5 3 17 2 1 13 4 11 10 20 7 14 6 16 15 19 12
	Rata-rata	616.4	66.8	

Tabel B.12: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 2 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	581	631	57	1 15 16 6 18 19 8 2 9 4 14 12 13 17 20 11 5 3 10 7
		638	130	1 7 16 17 20 18 14 5 12 13 6 15 10 3 4 2 8 9 11 19
		643	131	1 20 4 18 17 19 16 12 13 6 15 5 8 14 3 11 2 9 10 7
		634	155	1 20 2 6 15 5 3 14 11 9 4 18 7 16 12 13 17 10 8 19
		642	190	1 18 17 3 7 16 2 6 15 4 14 12 13 9 5 20 11 19 8 10
	Rata-rata	637.6	132.6	
7	695	725	119	16 3 2 5 10 17 1 8 11 18 12 7 19 9 4 15 6 20 13 14
		727	160	1 8 16 5 15 6 20 17 4 13 11 7 19 9 12 2 14 18 3 10
		728	119	20 14 16 3 11 7 4 15 1 8 6 2 5 9 19 13 18 12 17 10
		730	116	1 8 16 17 19 9 6 11 18 12 7 5 14 4 15 10 3 2 13 20
		722	178	1 8 16 17 20 6 14 2 3 7 19 9 5 15 4 11 13 18 12 10
	Rata-rata	726.4	138.4	
8	711	730	127	1 8 16 17 20 6 14 2 3 7 19 9 5 15 4 11 13 18 12 10
		732	128	14 7 11 5 6 9 2 15 3 17 16 4 13 10 19 1 8 18 12 20
		728	183	11 17 6 1 8 14 15 16 12 19 4 13 10 9 2 7 3 18 5 20
		736	60	6 18 19 5 14 9 1 16 4 3 2 13 10 8 11 15 7 12 20 17
		726	78	6 17 2 12 1 4 13 10 16 18 9 15 3 19 8 14 7 11 5 20
	Rata-rata	730.4	115.2	
9	662	738	153	14 7 3 17 18 20 8 15 13 9 2 19 11 16 6 1 12 10 5 4
		737	84	4 14 15 20 6 8 18 7 13 9 3 2 19 11 1 12 16 10 17 5
		736	152	14 2 4 7 18 5 15 19 16 20 9 12 13 11 1 17 3 8 10 6
		744	102	14 4 7 11 10 12 19 16 8 15 13 5 2 18 20 3 17 9 1 6
		738	168	7 13 14 2 19 17 9 15 5 1 11 18 20 8 16 12 3 10 4 6
	Rata-rata	738.6	131.8	
10	545	621	54	5 1 14 12 11 7 18 8 10 16 19 20 13 2 4 15 6 3 9 17
		612	182	1 11 14 5 2 12 15 3 13 9 17 8 18 7 16 4 20 19 6 10
		624	21	14 1 12 13 4 20 16 17 2 11 18 8 7 15 3 9 5 19 6 10
		620	83	1 18 7 15 13 5 12 8 19 16 6 2 14 17 11 4 20 3 9 10
		604	125	1 18 7 20 14 5 2 17 12 15 11 8 13 4 16 19 6 9 3 10
	Rata-rata	616.2	93	

Tabel B.13: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 3 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	442	488	142	6 18 20 7 10 14 13 15 8 11 4 16 1 9 12 19 2 17 5 3
		494	177	20 16 14 10 4 7 11 19 1 9 12 13 3 5 18 8 2 15 6 17
		500	130	15 6 20 11 8 10 1 12 13 9 5 4 19 18 3 14 2 17 7 16
		491	131	10 6 13 12 1 20 17 11 14 8 4 18 7 9 2 19 15 5 16 3
		494	90	20 7 13 4 12 19 8 5 14 1 6 9 11 18 10 15 2 17 16 3
	Rata-rata	493.4	134	
2	399	468	21	20 6 3 5 17 2 19 11 8 16 10 14 7 9 13 1 18 15 12 4
		469	192	17 5 7 6 16 3 20 10 9 2 14 18 8 19 13 1 4 12 11 15
		473	97	17 7 8 19 5 20 6 3 14 10 11 13 16 2 18 12 9 1 15 4
		467	71	2 9 18 13 19 10 17 6 11 8 3 14 7 16 12 5 15 20 1 4
		466	123	15 20 9 17 7 19 3 14 5 6 16 2 10 11 8 13 18 12 1 4
	Rata-rata	468.6	100.8	
3	403	440	192	8 14 5 16 18 17 12 4 10 3 9 7 1 15 2 11 19 6 20 13
		447	181	3 9 17 18 7 13 14 1 10 2 8 12 4 15 19 5 6 11 16 20
		446	60	18 12 11 8 7 16 4 10 9 5 17 2 1 15 19 6 3 20 14 13
		446	99	13 8 7 14 16 1 9 12 11 4 10 17 5 15 18 2 19 3 6 20
		436	73	16 18 13 5 7 1 12 9 2 17 4 10 15 14 19 6 3 11 20 8
	Rata-rata	443	121	
4	395	475	72	12 3 6 10 17 11 1 18 4 7 19 5 9 2 13 15 8 14 20 16
		465	83	2 14 4 6 12 17 13 19 9 20 1 7 5 3 15 11 16 8 10 18
		467	23	2 8 14 12 7 13 1 18 9 4 17 5 19 10 3 15 11 20 16 6
		463	165	4 1 12 14 6 3 17 2 19 18 15 10 9 11 16 5 7 20 13 8
		458	122	8 12 7 5 3 17 9 6 1 14 19 18 11 2 10 15 13 20 4 16
	Rata-rata	465.6	93	
5	469	546	13	3 7 6 9 16 5 1 10 13 12 14 20 2 11 18 4 17 15 19 8
		545	39	9 16 8 10 15 7 17 1 13 14 2 18 5 6 3 20 4 11 12 19
		540	175	6 9 16 17 5 7 3 12 13 15 19 18 2 1 20 4 14 11 10 8
		540	152	16 14 10 11 8 18 15 17 2 7 5 6 9 1 20 4 13 12 19 3
		546	181	13 16 9 6 7 5 15 20 2 18 4 17 3 11 12 1 8 10 19 14
	Rata-rata	543.4	112	

Tabel B.14: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 3 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	457	514	149	16 8 14 9 13 12 20 17 18 6 5 15 2 3 11 10 4 1 19 7
		511	188	17 13 6 5 10 15 9 2 18 20 11 1 12 16 8 4 3 14 7 19
		518	80	13 15 17 18 6 7 10 11 4 14 16 8 12 2 5 1 3 9 20 19
		510	110	14 19 17 1 15 13 5 8 12 10 11 20 18 6 16 2 9 4 3 7
		518	70	17 13 18 6 2 1 9 11 12 16 15 14 5 8 10 4 7 3 20 19
	Rata-rata	514.2	119.4	
7	492	522	113	3 12 2 16 5 19 15 10 9 18 13 17 7 1 20 4 8 6 14 11
		523	78	19 1 12 5 13 17 9 18 2 11 4 15 10 6 8 20 3 16 7 14
		525	186	4 2 12 7 5 1 17 19 9 18 13 15 14 8 20 3 16 6 11 10
		522	166	19 9 5 13 17 12 7 1 20 4 18 2 3 15 6 14 16 8 11 10
		523	32	1 12 4 19 16 8 6 13 18 9 20 14 7 17 15 2 5 10 3 11
	Rata-rata	523	115	
8	516	575	125	14 11 10 18 6 13 2 4 19 15 12 9 3 17 20 16 5 7 8 1
		575	64	11 19 15 9 13 3 10 14 18 17 6 2 4 12 8 7 20 16 5 1
		566	188	2 13 15 19 3 18 4 14 10 9 12 11 6 17 1 7 16 20 8 5
		568	94	3 17 11 10 14 2 18 12 9 19 1 4 6 16 15 13 7 20 5 8
		576	153	2 13 15 19 3 10 12 18 6 9 11 17 14 4 16 20 7 1 8 5
	Rata-rata	572	124.8	
9	461	507	151	4 2 11 19 13 6 20 7 9 3 1 8 10 18 14 5 15 16 12 17
		512	89	8 10 2 9 20 4 5 6 15 19 11 1 13 18 7 12 14 16 3 17
		510	197	6 7 20 2 1 4 15 17 8 11 13 9 5 10 19 18 14 3 16 12
		511	157	4 3 7 17 1 10 19 14 15 8 2 20 9 6 13 5 18 16 11 12
		508	92	20 2 19 4 7 18 11 6 14 1 13 16 10 5 15 9 3 8 12 17
	Rata-rata	509.6	137.2	
10	458	518	164	9 10 5 16 1 3 18 15 7 19 8 20 14 2 13 12 11 4 6 17
		514	146	9 18 7 19 5 16 1 3 11 13 4 2 20 14 8 12 10 15 6 17
		518	110	4 12 10 18 17 5 16 1 3 7 13 15 2 14 20 9 19 8 6 11
		514	46	7 5 9 15 3 13 20 18 1 11 12 10 19 16 4 8 14 2 17 6
		525	172	18 12 19 10 7 13 5 1 3 11 9 14 2 20 4 16 8 15 17 6
	Rata-rata	517.8	127.6	

Tabel B.15: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 3 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	442	481	169	6 10 12 1 20 8 19 18 2 14 4 13 9 11 7 17 16 15 3 5
		482	146	7 13 20 12 10 1 4 17 6 8 11 15 18 2 14 9 19 5 16 3
		488	58	6 12 7 13 20 5 3 19 18 11 8 4 1 10 2 15 16 9 17 14
		483	69	20 13 8 6 3 10 4 12 19 2 9 1 7 11 18 14 15 17 5 16
		487	36	20 1 7 19 10 2 6 3 11 8 18 12 14 4 15 13 17 9 16 5
	Rata-rata	484.2	95.6	
2	399	458	59	5 19 7 17 2 6 14 9 1 18 16 8 11 3 13 10 12 20 15 4
		466	153	15 19 20 11 9 8 3 14 5 7 17 6 16 10 13 2 12 18 1 4
		460	17	16 7 5 2 19 9 8 6 3 14 17 13 12 10 20 1 18 15 11 4
		459	177	17 20 6 16 10 19 2 14 9 5 1 8 3 7 13 18 15 12 11 4
		454	69	9 17 18 7 5 3 8 6 2 14 10 11 12 20 13 16 1 4 15 19
	Rata-rata	459.4	95	
3	403	432	140	9 12 5 17 1 8 19 7 4 10 18 3 14 15 13 2 6 16 20 11
		435	124	13 7 4 10 16 9 8 14 5 17 2 12 1 15 18 6 3 19 20 11
		434	164	7 14 8 15 9 18 10 17 16 1 12 4 5 3 19 2 6 20 13 11
		433	101	7 10 13 9 16 8 11 1 14 17 12 4 15 18 5 3 2 19 6 20
		437	104	8 12 9 7 15 18 5 2 19 13 4 14 1 10 17 3 16 6 20 11
	Rata-rata	434.2	126.6	
4	395	459	112	12 11 14 2 4 3 1 10 18 8 13 7 19 17 9 15 5 20 6 16
		459	97	4 17 6 14 9 7 13 18 12 19 15 5 1 10 2 3 16 20 11 8
		456	106	1 4 18 17 12 7 11 19 5 9 10 2 6 14 15 13 20 3 8 16
		457	163	12 4 11 9 3 6 13 10 18 17 14 19 1 2 7 5 15 20 8 16
		457	55	8 6 10 4 7 18 9 14 19 17 1 5 12 2 15 3 13 20 11 16
	Rata-rata	457.6	106.6	
5	469	536	174	6 10 3 16 15 7 20 9 4 1 2 17 13 11 18 5 12 8 19 14
		535	48	10 1 16 5 7 9 6 8 12 11 17 4 18 13 2 20 15 3 19 14
		539	65	15 18 16 7 5 20 9 6 19 1 4 3 12 8 13 2 14 17 10 11
		536	50	6 20 7 19 13 16 4 9 14 5 15 2 3 1 10 18 12 11 17 8
		536	129	7 1 20 9 6 2 16 12 3 4 18 15 5 14 13 8 11 17 10 19
	Rata-rata	536.4	93.2	

Tabel B.16: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 20 pekerjaan, 4 proses, 3 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	457	505	80	12 14 13 9 2 10 8 15 17 11 6 18 4 1 5 7 16 3 20 19
		504	171	17 18 6 13 5 2 10 3 16 8 12 1 20 9 15 4 11 14 19 7
		509	11	6 2 13 17 11 18 20 9 5 1 8 12 15 10 16 19 4 3 14 7
		507	145	17 1 12 13 8 2 18 6 5 16 9 15 4 11 10 20 3 14 7 19
		497	178	17 13 11 10 14 18 6 16 15 2 5 9 8 12 20 1 4 7 3 19
	Rata-rata	504.4	117	
7	492	518	110	2 12 13 17 5 19 1 8 15 10 9 3 16 18 4 20 7 6 11 14
		515	107	6 19 12 2 5 13 9 17 7 16 18 14 1 20 4 15 3 8 11 10
		520	37	13 5 19 1 16 2 18 12 9 10 8 17 6 3 15 14 20 4 7 11
		520	98	1 18 13 19 12 2 4 17 11 9 20 16 3 15 14 8 7 6 10 5
		517	129	12 2 13 17 19 14 16 15 1 18 5 6 4 20 9 8 3 7 10 11
	Rata-rata	518	96.2	
8	516	564	141	11 15 19 9 3 10 12 18 4 14 1 17 6 13 2 7 16 20 8 5
		565	156	2 16 11 17 6 9 3 10 14 15 12 18 4 19 13 5 20 7 8 1
		567	164	11 3 2 15 17 14 9 16 12 18 4 19 13 10 6 1 7 20 8 5
		567	29	15 17 11 3 10 14 18 6 16 8 2 9 4 12 19 13 1 20 7 5
		562	166	13 2 15 4 12 11 19 3 10 14 18 17 6 9 8 7 20 16 5 1
	Rata-rata	565	131.2	
9	461	497	76	4 11 17 6 20 7 3 18 15 2 9 19 10 5 1 13 8 16 12 14
		491	20	5 11 20 8 4 7 19 10 1 13 9 18 3 15 6 17 16 12 2 14
		501	97	20 17 18 4 13 9 19 7 6 10 3 1 15 5 2 12 11 16 8 14
		497	168	4 17 20 10 13 18 15 7 19 8 11 2 6 1 9 5 16 12 3 14
		500	18	4 11 19 15 20 17 6 10 5 3 9 7 18 8 1 12 13 16 2 14
	Rata-rata	497.2	75.8	
10	458	505	186	7 20 5 16 1 3 18 8 4 10 14 2 13 19 12 15 17 9 11 6
		514	12	3 5 7 4 18 16 11 1 14 13 10 19 9 20 15 2 12 8 6 17
		504	115	7 10 5 16 1 11 13 15 3 12 20 9 2 14 8 19 17 4 6 18
		503	19	5 12 16 18 19 7 1 3 13 9 10 20 14 17 2 4 8 15 6 11
		502	182	7 13 5 16 1 3 18 20 9 10 14 2 4 15 19 8 12 17 6 11
	Rata-rata	505.6	102.8	

LAMPIRAN C

DATA PENGUJIAN KASUS *HYBRID FLOW SHOP* DENGAN 50 PEKERJAAN DAN 5 PROSES

Tabel C.1: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	955	1168	88	44 5 34 40 22 25 9 20 29 31 6 15 19 43 41 36 33 24 26 32 28 14 12 13 46 39 50 45 7 2 35 49 42 47 3 8 30 1 27 23 21 10 37 16 11 17 48 18 4 38
		1169	193	17 44 25 1 29 33 23 28 10 13 7 45 48 8 30 39 32 31 49 50 46 37 42 47 21 27 16 24 35 36 22 26 43 3 11 20 5 6 19 12 15 34 18 40 41 14 9 4 2 38
		1172	148	39 28 32 38 44 37 23 3 13 49 22 25 30 7 1 29 6 18 50 20 21 16 26 41 47 40 27 43 24 17 34 42 14 8 4 45 12 15 33 11 19 9 48 5 35 2 46 10 36 31
		1177	119	27 29 6 44 47 18 33 28 1 12 19 24 41 38 13 17 42 5 20 40 4 39 9 16 10 25 46 36 37 49 8 48 2 43 31 21 14 22 15 35 50 32 3 23 30 26 45 34 11 7
		1179	68	10 28 1 43 7 6 41 29 34 46 9 23 25 13 27 26 21 40 14 3 30 35 19 16 20 36 39 42 31 12 49 50 17 4 15 24 44 33 18 48 37 8 38 45 2 47 22 5 11 32
	Rata-rata	1173	123.2	
2	909	1083	94	35 48 44 33 25 15 49 5 9 19 34 3 20 47 27 18 1 39 23 26 2 17 21 14 6 11 16 4 29 10 13 8 7 12 45 37 30 31 46 41 28 36 38 22 43 40 42 32 50 24
		1076	123	23 42 15 25 35 29 34 45 2 48 33 44 11 20 32 19 28 5 49 22 14 40 36 43 12 7 13 31 41 50 38 18 39 4 8 21 17 26 9 3 10 47 24 1 46 16 6 37 30 27
		1069	180	14 15 50 34 43 2 4 44 41 26 23 6 20 17 35 33 9 22 39 25 32 5 18 38 1 11 13 47 7 21 36 12 10 45 8 29 27 30 19 28 24 46 3 31 48 40 42 49 16 37
		1062	192	17 4 28 27 11 34 41 32 39 3 30 48 18 25 9 46 14 19 26 21 5 35 36 42 31 8 47 7 23 12 45 33 6 50 43 1 38 20 29 37 44 40 13 2 10 15 24 16 49 22
		1075	46	16 13 11 33 25 21 23 4 40 47 37 36 48 50 3 46 41 5 19 30 38 27 1 24 31 7 10 22 45 14 34 9 8 18 39 44 20 2 49 26 29 43 15 17 42 12 32 35 6 28
	Rata-rata	1073	127	

Tabel C.2: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	963	1151	183	13 31 34 4 6 24 50 15 2 3 12 5 37 19 47 33 36 14 7 1 20 10 42 23 29 21 28 16 17 32 46 11 39 48 27 25 26 44 49 41 38 9 8 18 40 22 43 30 45 35
		1155	148	22 30 1 24 15 12 25 43 33 26 41 7 16 36 46 42 34 3 4 37 48 31 39 47 5 23 13 32 29 28 11 44 2 8 14 6 40 38 50 10 17 49 35 18 20 45 27 9 21 19
		1152	55	32 6 31 7 33 21 25 34 29 35 13 4 43 17 49 10 27 39 15 2 26 24 22 14 5 3 47 28 36 9 48 12 11 37 41 20 18 19 38 1 8 45 44 50 46 30 16 40 42 23
		1153	148	25 44 46 40 4 49 31 11 16 32 12 39 6 29 45 28 24 2 37 17 41 19 36 10 8 27 42 15 13 22 50 48 33 3 1 14 43 26 34 5 7 23 21 30 18 47 35 38 9 20
		1159	165	28 16 14 37 41 15 4 47 50 26 35 48 44 9 31 3 1 24 36 8 21 5 42 10 11 27 46 20 39 17 12 25 7 2 49 32 19 22 43 30 18 13 23 29 33 38 34 40 6 45
	Rata-rata	1154	139.8	
4	1003	1233	112	32 42 10 19 14 4 45 47 25 11 49 24 27 9 37 6 36 7 28 17 38 30 40 43 23 48 41 5 1 21 13 31 20 18 12 22 46 50 34 35 2 3 33 44 26 29 39 16 15 8
		1222	93	13 34 15 11 22 45 35 8 26 40 46 10 37 2 3 49 27 20 48 7 36 19 5 42 6 31 30 47 44 33 9 29 14 50 21 28 16 41 25 23 24 43 1 38 17 18 32 4 12 39
		1219	182	8 21 3 1 10 37 25 47 42 12 41 18 7 27 19 35 22 31 17 45 33 49 23 26 36 6 28 39 2 50 24 48 29 34 43 13 15 30 40 38 5 44 14 32 11 16 9 4 46 20
		1229	126	41 31 13 37 8 39 11 32 25 20 16 46 29 27 45 12 36 42 10 26 35 14 22 21 40 43 2 3 6 34 49 33 44 19 5 9 28 18 7 23 50 30 4 17 47 1 48 15 24 38
		1210	188	14 1 16 44 34 47 12 46 33 10 20 45 2 31 5 6 30 38 17 11 49 42 48 13 28 35 3 23 27 7 41 50 19 24 43 40 8 9 36 21 39 26 22 29 18 25 37 15 4 32
	Rata-rata	1222.6	140.2	
5	993	1194	111	7 10 4 40 50 32 16 29 49 30 13 22 36 39 17 19 21 46 48 23 38 44 9 37 3 25 41 27 18 34 6 24 47 15 5 33 35 8 28 31 14 20 2 11 1 12 42 43 45 26
		1187	193	33 29 8 9 47 40 18 21 46 25 44 15 5 31 14 12 6 36 1 3 34 42 24 37 11 32 4 41 50 20 19 17 28 35 43 49 10 16 27 2 38 22 23 13 7 48 39 30 45 26
		1192	102	12 50 14 21 6 4 40 31 33 10 20 35 1 16 42 15 36 39 46 9 47 48 41 27 24 43 19 17 30 44 28 11 38 22 45 5 2 34 3 25 18 23 32 49 7 13 26 8 29 37
		1191	114	21 1 11 8 33 42 15 28 30 10 43 22 24 48 46 25 4 50 13 16 19 29 14 20 26 32 18 6 31 9 23 47 39 35 12 40 49 45 34 2 5 37 36 7 41 27 44 3 38 17
		1181	147	29 2 8 10 40 32 18 49 11 20 19 17 5 22 4 24 21 48 38 44 15 7 43 50 13 30 47 34 14 33 35 9 41 23 1 16 45 46 6 36 3 25 28 31 27 39 42 26 12 37
	Rata-rata	1189	133.4	

Tabel C.3: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^2 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	924	1118	64	47 22 1 19 36 42 26 15 14 25 29 6 35 48 18 33 13 3 46 5 20 10 45 8 16 30 24 44 12 31 21 23 39 28 9 38 50 41 37 40 4 34 43 2 49 7 11 27 17 32
		1109	72	37 1 32 43 5 41 26 12 46 48 39 11 33 10 50 38 45 47 20 31 30 14 27 28 13 4 24 9 23 7 19 3 44 34 36 29 17 8 18 6 15 16 25 35 40 22 21 49 2 42
		1111	195	1 43 25 33 32 8 20 22 10 36 35 13 12 40 5 48 45 30 11 28 15 49 46 2 47 21 3 26 4 6 42 14 7 24 27 37 44 29 23 16 17 50 38 39 34 18 31 41 19 9
		1090	181	11 29 26 32 45 47 43 46 25 40 28 7 36 10 6 50 22 30 41 31 15 16 1 33 19 13 24 21 9 12 14 20 48 17 34 44 3 18 23 37 5 4 2 27 35 39 49 8 42 38
		1109	183	32 8 12 43 26 9 25 3 37 2 20 17 33 7 11 29 50 39 28 23 41 34 21 22 42 1 40 6 31 30 48 5 15 45 24 35 36 10 14 18 44 16 47 13 38 46 19 49 4 27
	Rata-rata	1107.4	139	
7	969	1171	33	44 24 32 21 8 5 1 7 9 46 19 37 27 26 12 25 4 23 11 6 45 49 20 48 28 35 13 15 47 38 18 2 16 30 10 41 17 29 39 40 14 34 42 36 33 31 43 3 22 50
		1177	78	32 3 1 30 24 20 45 11 29 14 40 12 19 28 42 9 27 35 4 49 21 23 10 48 8 5 7 43 39 22 16 36 15 50 46 33 41 34 38 17 37 44 6 13 26 47 2 18 31 25
		1169	28	8 20 24 3 16 35 34 7 5 30 1 41 39 50 29 19 40 46 17 27 2 47 37 23 36 44 45 12 22 6 26 43 42 18 28 48 38 13 9 11 14 10 21 32 15 25 33 49 31 4
		1168	106	13 33 4 34 40 14 9 27 8 28 20 45 38 17 5 3 29 37 43 25 10 1 46 36 41 7 15 49 19 39 50 47 44 21 32 23 31 42 16 24 11 2 22 35 12 48 6 26 18 30
		1181	130	4 20 33 3 35 49 46 37 32 17 15 19 16 7 34 42 13 9 10 8 2 18 47 29 1 43 27 21 23 38 45 5 24 41 36 11 26 25 44 12 22 39 50 14 28 48 6 31 40 30
	Rata-rata	1173.2	75	
8	1080	1233	57	35 14 29 1 36 24 44 21 48 15 49 37 39 23 5 34 19 8 27 4 10 50 33 38 45 42 9 22 12 25 32 17 26 6 43 20 31 40 16 18 2 3 11 47 46 28 7 30 13 41
		1238	37	14 17 39 2 47 30 32 37 27 1 20 10 42 46 35 4 24 40 31 36 12 5 6 44 9 8 15 25 48 13 33 45 7 3 49 43 50 29 23 11 22 16 19 28 26 34 38 18 21 41
		1235	128	5 6 18 8 45 46 35 41 23 12 14 15 32 25 10 27 21 38 43 24 36 39 17 20 47 48 31 1 13 11 19 50 29 40 4 33 44 37 49 22 3 26 7 30 2 9 42 16 28 34
		1240	181	35 48 5 43 49 17 11 36 39 13 33 47 41 24 21 15 14 8 25 7 37 32 28 19 2 3 12 6 50 45 46 29 42 30 44 20 16 31 22 40 9 1 23 10 27 34 38 26 4 18
		1231	63	5 39 8 15 14 47 4 35 25 21 29 32 48 36 12 24 50 26 19 11 10 41 34 42 43 23 22 6 33 40 44 16 2 9 37 17 18 1 20 49 3 31 28 13 27 30 46 45 7 38
	Rata-rata	1235.4	93.2	

Tabel C.4: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^2 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	946	1169	155	44 16 26 17 7 12 40 50 39 37 34 41 6 3 5 24 15 43 11 30 29 2 20 42 46 4 36 48 49 13 21 31 32 47 18 33 22 25 38 35 10 27 14 1 8 28 45 19 9 23
		1169	175	12 27 36 15 16 17 38 6 49 41 22 4 24 37 20 30 26 10 32 7 43 25 14 50 21 31 1 34 19 44 45 8 35 23 13 28 33 40 48 18 2 29 42 46 11 39 9 3 5 47
		1153	180	12 5 27 1 28 26 3 33 18 43 46 24 8 7 11 32 37 39 17 6 47 50 31 35 38 20 29 21 41 13 42 10 4 44 2 40 22 25 34 14 36 23 49 45 15 9 30 16 48 19
		1167	120	34 46 7 26 18 39 1 5 14 42 10 35 9 8 12 4 2 21 31 15 29 30 13 28 44 17 49 45 48 6 22 33 20 38 36 41 47 27 11 37 32 50 25 24 16 40 19 3 23 43
		1160	132	50 39 18 1 15 27 17 40 43 22 48 29 37 11 42 23 33 26 38 24 21 41 20 3 14 46 4 5 8 32 7 35 31 44 16 34 45 10 36 25 13 28 6 2 12 30 49 9 47 19
	Rata-rata	1163.6	152.4	
10	977	1178	188	39 22 19 14 26 45 9 7 21 49 40 41 38 48 13 27 15 46 50 12 11 42 23 24 28 2 36 8 30 29 10 1 43 35 34 47 3 5 32 44 37 6 18 16 17 25 31 20 4 33
		1141	159	38 14 5 13 31 17 1 9 20 16 21 25 45 28 8 2 15 32 24 34 44 12 7 33 10 3 36 35 41 40 46 49 29 6 4 39 11 48 42 23 19 50 18 30 27 47 43 26 37 22
		1174	144	48 19 30 15 17 39 28 40 12 45 25 23 4 32 3 21 31 7 49 38 46 27 41 13 18 20 35 10 11 36 14 34 50 2 1 42 47 16 44 29 43 8 24 5 26 37 9 6 22 33
		1172	55	15 3 14 33 38 23 19 27 31 25 17 40 26 13 24 4 45 10 9 46 32 2 28 48 12 41 16 30 22 44 34 6 11 20 7 43 36 1 50 18 21 8 35 29 49 5 47 39 37 42
		1175	140	19 10 21 33 30 37 17 45 24 18 36 23 26 44 14 20 47 35 32 2 7 3 1 43 16 39 15 25 6 13 41 27 4 50 49 29 46 9 40 22 11 28 34 38 12 48 8 42 5 31
	Rata-rata	1173.2	75	

Tabel C.5: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	955	1153	162	18 10 42 39 29 48 47 12 32 15 2 8 43 44 7 25 21 13 33 23 41 20 46 9 16 19 45 3 30 14 4 37 50 22 28 27 6 11 31 5 35 34 36 40 24 17 49 1 26 38
		1151	186	5 28 39 25 29 1 37 36 48 50 40 20 43 17 46 23 27 47 35 19 24 32 3 22 13 26 30 34 11 49 41 8 16 2 6 31 7 14 44 33 42 10 12 21 9 4 45 38 15 18
		1158	187	1 29 37 13 44 16 42 10 43 8 28 34 6 19 27 2 12 40 20 33 17 32 3 14 4 9 7 23 41 5 35 39 49 50 47 24 21 31 45 46 18 30 22 36 48 38 11 15 25 26
		1148	59	30 25 14 10 32 44 29 48 18 33 45 47 1 43 7 35 39 11 17 27 4 19 12 37 8 5 34 36 2 16 31 42 40 50 6 49 9 20 23 13 24 28 22 3 21 38 41 15 46 26
		1148	186	39 23 41 45 29 1 30 17 14 48 11 32 31 43 2 20 9 7 25 27 35 13 3 4 10 28 50 6 44 34 8 5 36 33 24 19 26 37 16 22 38 18 40 12 42 47 46 15 49 21
	Rata-rata	1151.6	156	
2	909	1060	37	28 15 44 4 38 50 41 34 32 47 3 24 48 40 13 2 18 10 17 29 46 9 12 11 39 26 20 45 7 21 43 25 23 36 19 5 27 42 31 14 8 1 33 49 6 35 16 37 22 30
		1058	83	27 15 34 26 9 21 28 31 25 14 5 20 37 39 38 42 44 8 4 17 43 24 40 7 13 11 19 46 36 47 1 30 23 18 10 6 41 48 33 45 29 2 3 12 16 49 50 32 22 35
		1063	115	44 15 22 25 26 31 17 43 20 2 39 30 9 40 13 34 3 41 28 1 5 48 7 35 14 10 42 11 38 45 29 33 19 8 27 24 18 47 37 12 36 32 6 4 21 46 50 23 49 16
		1061	140	15 27 42 30 26 8 35 13 33 4 2 38 40 41 44 1 39 20 18 45 12 9 37 50 43 24 11 31 29 5 48 22 19 17 28 14 10 36 47 46 6 34 7 3 49 21 23 32 16 25
		1055	77	15 25 28 35 41 30 23 7 4 12 39 18 20 46 9 31 2 48 49 22 45 5 8 29 43 36 1 38 21 47 44 11 42 26 6 10 13 34 33 19 40 27 24 17 3 50 32 16 37 14
	Rata-rata	1059.4	90.4	

Tabel C.6: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	963	1126	106	1 37 24 5 25 22 48 35 23 11 46 29 27 36 8 32 40 16 14 26 19 49 41 4 3 12 13 6 15 21 50 7 33 47 34 2 43 30 38 18 9 45 31 44 28 20 17 39 42 10
		1128	186	32 37 46 16 18 19 33 4 50 34 3 44 30 36 8 13 1 9 41 38 23 43 49 22 2 40 47 17 48 24 11 26 31 7 39 35 45 15 12 21 29 10 5 14 27 25 6 20 28 42
		1134	53	33 9 1 12 10 44 24 3 5 36 28 16 30 26 25 34 35 49 41 40 27 11 23 7 2 15 18 13 14 38 22 50 48 19 17 45 37 47 32 6 39 31 29 46 20 21 8 43 42 4
		1133	84	25 46 12 40 10 49 5 2 16 11 39 9 28 8 36 37 23 18 4 6 30 45 33 26 43 14 21 22 17 15 1 27 38 50 41 3 24 19 44 29 32 48 47 31 7 34 35 42 13 20
		1135	25	24 28 22 35 13 41 5 18 17 1 40 38 3 50 23 11 27 42 34 44 9 7 48 16 15 19 46 26 43 21 33 6 37 49 20 31 29 32 2 14 10 36 25 47 12 8 30 39 45 4
	Rata-rata	1131.2	90.8	
4	1003	1193	128	40 21 1 17 10 45 44 14 28 47 35 26 42 29 30 32 22 48 19 7 3 46 38 12 27 6 31 2 37 36 13 23 25 41 49 8 18 50 34 16 11 9 5 43 39 20 33 4 15 24
		1198	131	47 30 1 45 5 46 28 14 24 23 35 26 40 11 22 34 44 42 50 7 25 43 19 6 31 37 13 17 49 3 4 48 21 29 2 12 9 32 8 15 27 18 41 10 38 33 36 16 20 39
		1197	160	21 41 10 43 47 33 2 28 46 5 18 16 7 49 8 37 20 3 14 13 31 23 35 25 48 44 36 17 29 26 39 22 42 12 40 11 32 30 27 38 6 9 24 50 19 1 45 15 4 34
		1191	149	30 41 21 11 37 33 49 13 16 18 50 26 9 10 43 44 22 31 45 5 1 25 24 42 35 15 32 12 27 36 6 28 34 17 8 46 3 2 40 38 7 4 14 29 23 19 39 48 47 20
		1204	27	31 37 1 13 23 41 2 40 8 15 25 11 42 49 28 24 30 35 44 48 12 26 50 5 36 34 3 7 39 18 45 14 19 46 22 33 6 4 20 10 43 27 29 16 38 17 32 9 21 47
	Rata-rata	1196.6	119	
5	993	1162	56	33 35 45 50 23 40 2 8 10 3 9 27 46 22 29 39 38 25 48 19 49 11 32 36 44 30 17 24 7 21 31 28 4 5 13 42 15 47 34 1 6 16 14 18 20 41 43 26 12 37
		1165	56	23 7 39 8 46 47 2 38 43 9 30 6 31 13 14 28 44 22 50 32 19 10 20 25 35 40 45 33 15 49 3 11 34 29 4 5 12 36 21 27 18 1 24 42 17 48 41 26 16 37
		1172	86	8 29 42 22 21 39 49 9 19 15 38 28 5 16 10 43 7 3 47 23 37 24 14 27 48 1 50 25 6 46 31 44 34 33 20 11 41 4 13 40 35 30 32 18 36 2 17 45 12 26
		1181	114	21 35 33 47 16 50 43 38 11 48 23 41 29 2 17 32 10 19 9 18 14 37 31 15 36 39 5 1 28 30 25 40 3 6 4 49 13 46 7 27 22 34 20 12 8 44 24 26 42 45
		1164	80	33 45 7 35 50 27 47 25 2 46 40 41 4 1 6 17 13 19 38 48 29 49 21 18 23 32 36 20 30 44 39 9 15 31 8 12 10 3 14 5 34 28 11 24 22 16 42 43 26 37
	Rata-rata	1168.8	78.4	

Tabel C.7: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^3 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	924	1088	188	43 25 20 49 1 3 46 4 39 45 28 30 31 35 11 29 33 12 7 22 47 21 37 40 17 44 50 5 36 10 23 41 34 48 26 42 18 2 13 6 15 16 14 24 8 19 38 27 32 9
		1088	109	29 25 41 1 30 45 26 27 39 3 34 33 13 12 15 14 5 43 35 2 21 9 46 22 42 24 36 50 4 16 6 44 17 18 23 37 28 48 49 47 20 8 38 11 10 32 40 19 31 7
		1096	167	36 45 28 20 43 5 15 37 33 10 21 12 48 35 6 18 22 44 16 26 17 39 34 19 25 30 8 47 14 49 50 41 2 1 23 46 27 4 13 24 9 11 29 3 42 40 32 7 38 31
		1087	99	29 47 3 8 11 5 36 35 43 33 49 32 48 28 12 7 30 6 38 10 14 45 23 18 44 22 25 13 1 20 26 46 19 39 17 15 16 37 41 50 24 21 2 40 34 4 42 31 27 9
		1091	109	13 45 11 47 43 1 19 20 22 42 28 27 14 30 3 29 4 46 5 36 2 24 44 16 21 17 23 41 18 49 25 35 32 34 50 12 26 7 8 15 37 9 48 6 33 39 10 40 31 38
	Rata-rata	1090	134.4	
7	969	1155	114	3 32 17 44 7 21 48 28 18 20 5 2 24 39 8 27 33 23 45 34 49 46 31 42 30 40 15 50 19 43 16 11 36 41 25 13 14 38 22 35 12 37 9 29 1 4 10 26 6 47
		1143	187	16 32 7 41 8 6 35 43 1 11 29 3 33 19 50 21 23 4 22 37 28 20 2 39 14 38 10 30 44 45 34 15 9 42 46 36 5 48 31 27 26 24 12 18 13 40 17 47 25 49
		1158	170	3 24 20 29 21 12 45 11 6 9 44 18 42 49 41 32 19 16 38 31 10 2 28 5 4 22 43 13 30 34 15 17 27 33 35 14 40 39 36 50 1 23 26 7 46 48 8 37 47 25
		1156	173	11 6 8 46 7 3 18 24 25 31 32 21 48 17 27 41 15 50 43 39 14 38 45 12 19 37 23 20 36 30 35 1 22 44 29 10 9 34 42 13 33 28 5 16 2 47 26 40 49 4
		1147	179	28 27 24 37 8 3 46 23 14 48 5 12 17 22 33 13 36 39 19 49 38 7 20 9 4 40 41 16 35 32 34 15 43 44 45 21 25 11 30 31 29 50 6 47 42 10 2 26 1 18
	Rata-rata	1151.8	164.6	
8	1080	1210	153	45 6 39 5 17 21 19 14 12 49 27 40 48 31 35 46 43 22 3 20 13 38 33 36 1 29 41 26 50 23 37 2 11 8 7 44 24 9 42 15 4 16 25 32 47 18 10 28 30 34
		1205	127	35 7 45 14 10 48 6 37 44 2 49 40 9 4 11 16 24 33 22 18 41 8 23 26 38 27 15 34 25 21 28 29 32 17 39 31 50 30 46 3 36 20 12 1 43 19 5 47 42 13
		1219	165	3 35 31 13 27 39 32 42 8 6 45 21 36 2 18 34 47 29 20 12 11 24 38 4 41 14 15 22 49 37 19 33 50 40 9 5 7 23 1 46 43 25 10 16 44 17 28 48 26 30
		1219	168	14 39 16 43 18 50 48 42 17 49 15 30 3 41 47 29 23 44 20 31 9 37 1 35 2 5 33 8 22 25 13 6 12 40 24 11 21 32 27 7 45 46 28 19 10 36 4 38 26 34
		1215	119	39 14 49 34 18 9 5 21 48 12 15 6 43 33 40 36 23 42 28 10 19 11 44 13 47 27 16 25 31 38 50 26 37 8 29 2 1 3 17 20 24 45 7 22 46 4 41 35 32 30
	Rata-rata	1213.6	146.4	

Tabel C.8: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 3 mesin, dan n^3 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	946	1131	165	26 18 27 38 42 12 35 7 37 46 4 45 19 32 24 31 11 14 28 34 40 36 43 30 41 13 21 10 3 1 17 44 25 9 8 2 5 22 33 6 20 47 39 48 15 29 23 49 16 50
		1138	150	5 43 38 18 45 39 26 27 44 31 8 20 42 1 11 46 10 21 4 25 6 12 24 40 22 49 3 36 7 2 37 23 41 13 50 16 35 14 17 47 9 29 15 48 32 33 34 19 28 30
		1145	88	26 12 1 31 35 46 40 21 23 39 11 37 34 43 3 25 24 22 41 17 42 9 27 47 44 36 14 28 4 33 13 19 18 50 29 32 7 15 10 30 6 8 20 2 5 45 49 38 48 16
		1147	108	12 27 44 26 49 39 15 33 30 36 3 28 13 16 45 22 25 35 42 6 10 18 41 38 14 8 32 20 37 21 31 29 23 1 24 2 5 4 43 40 7 11 46 17 9 34 19 48 50 47
		1142	158	12 43 24 44 31 26 5 27 50 17 33 4 21 7 35 25 16 3 36 34 14 10 8 47 40 13 19 32 11 30 37 2 46 38 41 18 29 15 9 22 1 42 20 39 23 45 49 6 28 48
	Rata-rata	1140.6	133.8	
10	977	1150	70	14 34 22 44 3 4 29 21 47 1 17 26 2 28 5 37 30 43 23 24 39 18 45 19 50 27 41 49 38 16 7 20 36 15 32 9 42 13 10 35 46 11 25 40 12 6 8 48 31 33
		1144	153	42 19 25 30 13 33 7 45 24 6 34 36 22 48 35 47 17 32 3 37 23 4 28 44 5 14 1 40 18 46 38 27 15 21 43 16 10 49 50 41 12 29 39 11 2 9 8 20 26 31
		1161	138	14 13 3 44 24 30 46 23 29 49 35 34 1 19 8 38 10 28 16 48 2 41 36 22 50 42 32 43 33 31 17 40 15 25 18 12 45 11 21 5 27 39 4 7 20 47 26 37 6 9
		1153	62	44 34 19 11 27 38 21 29 31 12 41 18 2 9 40 45 23 33 36 20 30 49 13 1 42 14 15 25 35 22 17 39 28 47 50 7 43 10 6 16 4 32 24 5 8 46 48 26 37 3
		1131	184	38 42 17 3 21 45 1 25 37 43 11 13 24 23 16 39 41 46 47 19 28 50 9 10 5 35 26 32 48 31 49 8 34 7 2 12 36 30 44 40 18 15 27 14 22 29 6 20 4 33
	Rata-rata	1147.8	121.4	

Tabel C.9: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	682	813	122	43 35 29 34 12 25 41 26 23 6 22 1 9 40 27 10 50 7 33 31 48 24 8 46 19 47 14 17 4 37 18 36 16 5 44 11 2 20 13 32 21 49 38 39 28 45 15 42 3 30
		811	184	38 8 10 37 36 18 41 48 40 44 34 22 24 47 31 39 11 2 50 42 5 35 32 1 14 15 19 49 28 12 27 33 6 20 16 9 43 13 46 26 29 4 3 21 25 7 30 23 45 17
		810	63	25 42 38 46 12 7 21 37 8 26 40 6 15 32 48 18 5 3 16 33 9 14 10 36 11 31 23 27 50 28 2 49 24 1 20 29 44 43 4 34 39 47 35 19 13 45 22 17 41 30
		811	3	35 29 13 37 40 26 31 14 8 41 1 5 2 7 39 30 10 21 50 25 48 22 47 4 6 3 38 20 33 34 9 43 12 11 49 32 23 18 28 45 27 42 16 19 44 15 36 46 17 24
		813	10	17 18 27 31 33 37 9 5 13 23 40 24 6 26 4 12 47 28 3 35 8 15 46 32 7 48 22 2 14 43 29 25 39 1 38 10 21 34 44 42 11 16 49 19 36 20 30 45 50 41
	Rata-rata	811.6	76.4	
2	618	757	169	43 29 37 12 27 22 49 17 46 8 9 42 2 48 6 45 1 40 44 11 41 13 19 38 15 36 23 7 32 5 25 30 28 18 20 31 14 39 10 21 16 47 50 3 24 26 4 34 35 33
		764	175	25 44 16 29 43 11 33 46 36 45 38 20 2 50 35 18 41 1 19 39 8 28 13 48 3 26 47 7 40 12 21 15 14 42 4 17 37 32 49 27 10 30 5 22 23 24 31 6 34 9
		769	68	29 44 23 22 7 43 50 36 2 46 9 35 21 39 42 10 49 1 48 13 45 14 40 31 11 41 47 16 18 32 27 37 12 20 30 17 28 5 19 3 34 8 26 33 15 38 25 4 24 6
		769	163	49 31 29 46 7 13 48 32 10 27 26 3 44 43 40 5 30 45 22 33 36 21 25 35 18 16 20 4 1 23 28 39 9 2 17 50 11 38 24 15 14 37 41 19 12 34 47 8 42 6
		764	64	12 44 1 26 39 22 13 40 3 2 18 16 34 37 48 7 4 32 46 42 21 8 24 5 20 10 15 43 11 45 25 29 14 36 41 17 6 19 28 50 31 23 9 49 33 38 35 47 30 27
	Rata-rata	764.6	127.8	

Tabel C.10: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	589	703	48	6 7 41 12 9 40 22 48 5 42 20 30 31 43 50 38 35 16 34 33 8 4 13 15 18 39 21 26 44 14 23 49 47 46 11 10 45 27 37 3 2 28 25 29 19 36 32 1 17 24
		703	180	20 25 16 34 1 43 24 36 13 5 50 21 35 46 31 4 7 41 48 18 15 23 49 6 9 17 28 14 45 30 12 44 19 40 42 26 37 10 38 29 47 2 3 22 8 33 11 32 39 27
		698	182	21 40 29 8 1 28 41 35 12 48 26 13 27 10 42 30 3 38 50 34 5 39 45 16 14 49 20 23 17 25 33 43 4 7 36 6 15 46 11 18 2 31 47 9 19 24 32 22 44 37
		705	164	28 3 31 15 13 7 20 25 14 37 46 39 22 21 43 1 2 49 41 42 16 36 45 33 5 29 8 24 11 40 10 23 26 35 50 48 12 47 34 32 18 4 19 38 9 6 30 27 17 44
		694	115	28 13 40 42 11 43 17 48 26 37 3 6 15 1 36 29 8 46 2 18 7 33 30 5 4 34 9 10 41 44 16 50 45 12 47 25 14 49 20 38 21 23 32 35 24 22 31 19 27 39
	Rata-rata	700.6	137.8	
4	616	746	142	6 32 36 48 38 10 16 24 45 19 9 14 49 8 28 13 30 43 20 26 7 4 35 1 25 39 41 2 37 47 40 27 15 46 29 5 11 50 44 18 17 12 34 33 42 22 3 21 31 23
		741	78	13 11 2 1 8 29 36 28 10 34 48 40 21 20 18 50 25 9 41 5 6 35 42 44 37 27 12 38 43 16 39 7 24 3 14 46 49 17 22 23 15 4 19 32 31 33 26 47 30 45
		747	36	12 2 27 34 16 24 43 42 7 41 8 47 37 28 38 6 48 40 18 13 10 17 20 23 35 3 46 32 30 11 1 14 49 15 9 45 39 29 50 26 22 33 31 44 21 5 4 36 25 19
		750	128	20 46 47 11 36 7 15 31 48 44 30 33 34 10 49 43 39 2 18 17 5 6 3 38 16 19 42 32 50 40 28 23 13 14 12 9 35 45 4 41 24 27 1 21 29 26 8 22 25 37
		747	137	11 2 28 36 20 15 45 18 43 41 49 31 3 13 1 12 40 38 46 48 33 29 42 30 21 39 7 10 27 5 44 16 19 37 32 24 25 50 14 35 8 9 34 26 17 22 4 47 23 6
	Rata-rata	746.2	104.2	
5	639	742	159	35 15 32 26 12 9 14 31 40 42 33 44 11 21 37 1 2 24 34 28 50 30 10 20 46 38 23 22 17 41 47 4 13 18 8 43 27 3 5 49 25 48 7 6 45 29 16 39 19 36
		730	132	12 35 38 34 23 21 8 14 11 6 16 1 2 24 15 13 41 36 25 31 28 27 49 5 33 3 32 18 29 40 50 26 22 46 17 4 45 44 19 20 48 37 10 43 30 47 9 7 42 39
		742	59	25 45 13 9 23 5 3 38 30 8 2 27 11 16 47 18 50 1 36 22 37 41 32 10 14 4 33 26 44 28 17 34 31 42 49 46 24 35 39 7 12 15 21 29 20 6 43 40 48 19
		741	143	4 16 21 45 47 44 5 1 7 3 27 42 30 15 10 2 31 23 38 35 24 12 33 13 37 34 26 39 22 49 50 25 40 6 8 41 46 28 14 17 18 32 11 36 29 43 20 9 48 19
		740	42	44 13 21 32 35 41 2 6 50 12 17 5 15 45 4 10 42 49 16 40 27 18 14 8 28 31 30 37 47 46 22 1 29 38 34 24 23 11 25 33 26 3 20 36 43 48 9 19 39 7
	Rata-rata	739	107	

Tabel C.11: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^2 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	689	793	66	19 28 48 47 16 20 27 50 9 18 5 43 4 41 40 13 45 33 32 31 2 29 46 44 36 17 35 49 7 37 30 42 10 22 12 8 23 11 6 38 34 25 1 15 3 26 39 21 14 24
		792	133	49 50 30 19 22 32 36 9 45 13 1 35 46 47 38 44 40 39 43 12 2 27 3 31 5 7 37 29 26 23 8 20 34 24 15 11 4 48 17 18 42 21 16 33 10 41 14 28 6 25
		791	86	48 2 19 35 36 28 7 18 9 23 17 6 14 50 1 5 32 20 27 11 34 33 31 8 3 38 47 40 43 41 44 29 24 12 22 16 49 46 30 15 37 13 4 39 42 45 25 10 26 21
		802	43	45 34 19 20 18 29 22 21 17 7 33 9 1 48 30 31 6 27 36 11 35 43 23 28 42 38 5 15 2 40 44 49 13 37 14 24 8 32 16 10 46 12 41 25 4 50 39 26 3 47
		799	178	19 36 45 46 11 22 38 40 6 27 17 13 2 4 33 50 18 7 35 24 8 34 26 42 32 49 48 10 41 44 31 43 12 30 39 28 29 37 5 9 15 23 20 21 3 16 47 14 1 25
	Rata-rata	795.4	101.2	
7	642	759	144	16 37 20 24 1 36 14 32 6 11 2 17 45 50 19 30 34 39 29 40 10 21 47 33 27 43 25 49 48 3 9 8 42 22 4 28 23 12 15 7 18 5 31 35 13 26 46 38 41 44
		757	83	17 41 30 15 24 4 36 43 2 28 19 13 12 37 16 3 49 11 45 33 48 8 42 5 44 6 46 14 26 1 23 27 18 34 21 47 38 9 7 40 50 29 35 39 31 20 25 10 22 32
		760	122	37 14 2 44 11 13 16 23 9 26 46 10 19 21 7 15 25 49 29 18 41 31 43 5 40 20 47 27 34 39 4 30 3 38 1 36 12 33 6 45 50 48 24 28 42 17 35 22 32 8
		761	114	42 2 11 37 26 33 35 9 34 38 6 47 18 4 49 41 50 22 44 14 27 1 21 30 20 19 24 5 28 16 31 46 29 40 3 45 17 43 48 15 23 39 7 36 10 12 8 32 25 13
		762	121	46 20 2 15 17 33 45 19 36 18 12 24 1 39 7 16 8 32 21 28 47 49 14 23 50 22 35 3 27 9 11 13 37 26 42 34 4 48 40 6 43 29 5 30 38 31 41 10 44 25
	Rata-rata	759.8	116.8	
8	641	773	188	46 11 28 8 2 9 48 5 22 27 10 6 35 12 23 40 24 33 38 26 7 17 13 32 21 19 42 1 50 44 41 34 47 30 14 3 39 18 29 45 25 37 4 16 43 49 20 31 36 15
		776	186	4 23 39 40 38 3 22 48 17 35 31 2 25 11 36 33 16 10 7 29 34 1 50 21 44 26 14 13 24 46 5 45 47 27 32 20 30 8 9 28 19 41 18 37 42 12 15 43 6 49
		782	51	20 39 25 2 13 9 30 33 32 6 19 23 43 40 17 3 8 45 16 38 10 35 28 21 5 4 24 27 37 15 1 34 22 47 11 42 26 46 49 31 41 14 50 7 36 12 18 48 44 29
		777	150	21 20 27 34 4 39 30 35 5 29 48 17 13 22 24 41 43 7 14 49 3 1 6 23 16 28 8 47 31 42 12 26 2 10 44 33 32 25 45 46 38 19 37 40 9 36 11 50 15 18
		775	176	4 34 24 7 6 28 18 30 42 41 21 23 3 1 47 5 26 2 38 31 33 17 50 25 22 36 39 43 16 14 35 10 32 13 45 20 40 46 44 8 12 27 9 37 11 15 49 19 48 29
	Rata-rata	776.6	150.2	

Tabel C.12: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^2 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	595	726	26	6 18 33 24 40 3 19 27 42 41 36 4 1 29 30 20 28 12 35 9 26 46 31 34 37 25 32 5 11 15 17 7 2 14 16 47 10 48 45 13 38 49 8 39 50 44 43 21 22 23
		733	21	27 24 16 35 33 47 25 5 19 45 9 21 15 4 7 20 29 40 42 10 26 44 28 6 12 17 48 34 37 13 50 39 49 14 36 32 38 3 30 22 41 8 1 46 2 31 18 23 11 43
		728	119	45 9 32 42 36 30 37 18 3 35 7 10 31 33 4 43 14 28 27 41 16 20 17 25 23 21 6 13 40 38 46 34 26 12 47 48 1 8 50 49 19 5 24 39 44 15 29 11 2 22
		737	152	31 32 36 45 28 49 50 33 27 23 5 17 48 7 25 9 12 3 47 21 40 29 15 41 26 16 46 38 19 6 13 10 42 8 30 14 37 20 18 2 24 39 4 11 1 35 43 34 22 44
		722	154	5 11 49 48 20 18 28 36 33 6 47 30 39 4 32 25 24 14 31 45 12 43 10 7 22 9 50 37 1 29 40 3 13 27 19 26 35 41 17 21 42 16 46 44 34 8 23 2 15 38
	Rata-rata	729.2	94.4	
10	610	732	50	46 49 37 6 26 34 21 15 27 47 4 13 35 3 40 30 8 12 16 29 9 48 36 50 1 11 38 31 22 20 45 42 23 33 24 10 7 17 28 43 14 41 44 32 2 19 39 5 18 25
		746	23	37 8 3 26 21 2 39 1 16 6 43 30 12 38 4 11 47 45 28 13 24 15 23 7 18 19 46 9 42 49 29 10 33 14 20 17 34 50 31 48 44 27 41 25 32 40 22 35 36 5
		741	112	34 24 6 3 30 46 2 18 41 33 21 26 47 27 49 5 10 28 17 23 29 43 8 13 1 20 38 31 14 36 42 7 45 11 39 37 48 32 16 50 19 44 9 25 12 4 22 35 15 40
		743	18	3 34 42 15 21 37 30 17 2 4 6 16 26 22 49 40 43 45 27 14 13 9 48 36 11 7 33 29 23 1 50 10 35 24 47 18 38 25 8 46 41 28 31 20 32 19 12 5 44 39
		753	186	1 21 14 29 47 6 45 24 15 44 22 12 43 41 38 42 32 16 50 48 19 10 8 49 37 30 2 28 34 33 23 7 46 39 26 31 3 17 11 40 9 18 27 13 25 5 20 4 35 36
	Rata-rata	743	77.8	

Tabel C.13: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	682	802	139	26 31 41 10 35 6 38 12 16 7 22 24 21 23 5 47 19 50 33 37 32 9 30 4 43 20 39 40 25 1 13 34 29 14 11 49 2 48 3 18 15 28 8 27 42 44 17 46 36 45
		800	108	14 12 7 37 40 49 1 35 44 19 25 41 48 46 8 24 16 11 29 26 33 23 10 28 50 43 45 18 4 22 27 47 9 32 13 34 2 38 42 39 5 3 20 21 6 17 36 30 15 31
		794	117	5 11 41 15 1 31 37 10 25 22 8 39 40 12 24 43 26 7 48 33 34 21 9 47 28 36 18 44 2 49 6 27 20 17 32 38 14 16 13 35 19 4 42 3 23 50 29 30 46 45
		794	121	6 31 32 25 5 9 37 24 11 47 42 18 21 23 13 14 39 36 16 50 48 41 28 1 3 10 43 45 26 12 49 34 4 19 38 35 27 40 22 2 7 44 33 20 8 46 29 15 30 17
		803	116	47 35 41 17 5 33 26 18 28 3 50 37 49 9 10 22 19 7 34 44 13 11 40 31 29 38 32 16 6 15 20 48 2 8 39 12 30 27 25 14 43 45 23 46 42 4 36 1 21 24
	Rata-rata	798.6	120.2	
2	618	748	68	2 43 20 15 46 50 1 18 13 45 21 22 42 48 38 10 33 24 5 3 40 8 9 26 25 30 44 37 17 49 35 14 19 29 31 7 32 41 11 36 28 16 39 23 4 12 27 34 47 6
		745	69	29 35 26 41 43 46 36 12 11 50 30 40 10 27 23 1 32 45 7 16 21 24 9 17 44 37 31 5 18 47 38 42 3 14 22 49 48 34 20 19 33 8 13 28 2 39 25 6 4 15
		750	68	42 34 22 21 2 25 38 46 39 3 35 9 8 31 18 30 49 1 17 26 48 33 20 50 29 40 36 47 13 32 10 23 7 41 45 19 14 44 11 15 28 27 24 37 16 12 4 6 5 43
		746	147	50 8 29 13 22 43 7 46 16 10 30 33 2 9 48 5 40 11 38 20 12 32 49 41 37 44 15 18 39 17 35 27 1 45 4 3 26 36 21 31 19 14 28 24 25 23 42 47 6 34
		747	164	41 37 22 39 26 29 2 46 27 25 12 10 4 35 8 38 28 40 48 45 50 9 42 33 17 7 13 43 32 15 18 3 47 21 36 24 11 20 14 1 16 19 34 31 5 23 30 44 6 49
	Rata-rata	747.2	103.2	

Tabel C.14: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	589	695	122	30 32 43 29 1 21 37 27 3 39 20 28 44 11 14 33 5 46 13 2 22 7 12 25 8 41 4 34 18 16 24 48 45 35 49 6 42 10 23 47 50 15 36 26 9 17 40 31 19 38
		687	106	2 34 13 11 48 47 22 41 30 43 23 33 26 19 46 28 20 39 35 27 40 44 31 25 24 45 14 5 3 15 7 17 4 1 16 9 12 49 6 29 10 42 21 18 8 50 32 36 38 37
		692	74	42 30 29 6 13 8 48 49 10 20 33 47 12 34 7 24 46 40 2 21 44 31 22 3 11 19 39 36 5 4 18 14 23 37 9 35 41 45 16 15 17 50 26 27 28 32 25 38 1 43
		693	139	13 43 37 1 9 16 48 35 28 32 20 36 41 5 50 39 49 24 30 22 31 25 21 15 33 47 4 45 40 34 6 46 7 23 2 42 44 14 19 3 38 18 10 26 11 12 29 8 27 17
		692	149	41 24 11 3 45 30 39 13 49 26 33 23 38 35 32 48 4 10 16 12 21 46 42 18 20 28 1 36 15 31 29 7 8 43 9 40 17 14 2 47 34 25 6 37 22 19 50 5 44 27
	Rata-rata	691.8	118	
4	616	724	178	15 48 12 18 10 6 34 13 32 1 44 33 36 43 20 40 50 28 16 31 22 25 26 47 29 3 38 45 39 35 4 19 41 7 14 49 9 2 46 24 11 42 8 27 17 37 5 23 21 30
		730	160	39 10 40 29 24 13 7 28 48 2 34 19 46 30 11 35 38 47 14 37 41 50 16 44 20 15 22 27 25 43 9 6 32 21 36 49 3 4 18 12 8 33 1 23 42 45 26 5 31 17
		734	166	31 24 46 12 7 14 42 48 49 41 34 20 21 3 11 8 13 16 32 9 19 39 1 15 45 22 50 23 5 38 43 4 37 18 10 29 25 40 28 47 30 33 44 27 36 17 26 35 6 2
		730	181	34 13 9 29 21 14 12 42 36 39 47 48 32 40 28 38 41 49 10 8 35 7 24 3 37 16 26 6 33 15 50 1 45 20 43 46 18 44 2 11 22 27 5 19 31 4 23 25 30 17
		737	72	48 45 12 38 43 40 34 32 1 41 21 42 14 18 17 49 37 15 35 16 11 20 39 24 28 31 7 3 33 6 2 36 46 44 13 22 29 50 8 27 10 4 9 26 47 30 5 25 23 19
	Rata-rata	731	151.4	
5	639	717	138	26 23 21 31 44 15 37 16 35 41 29 14 45 30 32 34 38 42 2 27 11 18 33 49 50 13 25 43 8 3 12 6 5 24 28 47 46 22 10 40 17 9 20 36 4 39 48 7 1 19
		728	120	30 14 5 25 16 19 43 44 46 6 23 31 38 40 24 37 22 11 8 35 48 17 3 7 26 4 15 32 9 36 20 10 50 27 28 41 34 33 49 13 45 12 2 18 21 29 1 42 47 39
		725	131	21 22 32 2 40 5 49 23 48 10 12 3 26 20 14 44 35 33 31 16 24 17 15 42 8 38 30 50 37 9 25 28 46 47 19 41 4 27 34 1 13 43 18 11 6 45 7 29 36 39
		723	117	40 27 21 30 5 41 4 13 44 33 38 32 42 49 34 26 23 9 14 20 50 17 35 24 28 6 2 31 22 45 12 43 10 8 25 16 15 46 19 3 11 47 36 1 7 37 18 29 39 48
		724	93	25 43 44 13 31 22 47 14 24 35 17 19 33 28 40 34 21 5 3 16 49 36 11 27 10 46 6 38 32 20 37 23 41 2 45 15 26 18 48 4 50 42 8 9 7 29 30 12 1 39
	Rata-rata	723.4	119.8	

Tabel C.15: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^3 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	689	778	130	24 29 19 36 22 50 33 43 8 47 20 16 30 49 3 44 13 31 42 4 21 2 35 18 27 46 1 15 40 6 37 26 45 39 32 17 25 38 48 14 41 23 10 9 12 5 28 7 34 11
		782	139	22 45 29 4 13 18 2 21 44 48 47 33 49 19 50 36 32 27 11 12 41 3 37 23 26 8 46 38 28 30 34 17 9 39 20 10 6 35 43 5 15 40 42 24 14 25 16 7 31 1
		784	96	45 19 36 33 3 2 37 40 14 32 16 50 31 28 6 10 27 20 38 8 7 42 29 21 11 30 44 18 5 9 39 25 48 47 43 49 46 41 17 22 23 26 13 24 12 35 34 15 1 4
		781	138	31 2 48 45 38 25 9 36 24 17 37 41 18 14 33 10 50 49 20 44 27 19 46 4 5 22 16 29 30 32 40 35 13 47 21 39 23 8 3 28 12 34 42 15 6 7 1 43 26 11
		779	175	34 19 35 29 32 16 2 6 50 5 39 33 10 8 18 13 21 27 22 36 48 49 17 4 38 40 31 3 44 43 23 42 28 46 37 30 20 7 24 9 15 41 12 45 14 25 47 11 1 26
	Rata-rata	780.8	135.6	
7	642	744	76	24 20 23 17 33 43 49 40 11 14 19 34 5 16 42 31 9 7 36 37 8 2 12 35 1 27 29 48 6 30 50 45 18 15 13 38 46 39 25 3 21 26 4 44 32 28 10 47 22 41
		752	156	2 27 9 37 16 5 20 12 6 33 44 26 21 36 38 3 42 50 30 31 49 24 14 39 43 29 46 40 15 7 41 18 34 11 8 47 28 23 17 19 4 1 48 45 25 13 10 35 32 22
		751	116	30 45 20 36 33 16 49 14 6 3 42 25 50 34 2 35 24 10 12 37 22 4 39 8 1 15 17 7 23 38 21 41 27 28 18 19 40 31 43 44 48 11 26 5 46 9 29 47 13 32
		746	172	49 16 33 23 1 18 27 17 44 25 46 21 2 13 40 6 3 45 7 30 24 26 11 5 38 34 50 15 14 41 4 37 42 43 28 36 31 19 29 39 35 48 12 10 8 22 9 20 47 32
		747	131	21 2 26 36 30 35 33 44 3 40 28 8 11 47 4 18 15 19 17 24 43 48 13 23 14 49 50 20 27 10 6 12 9 29 34 5 37 31 42 7 16 1 45 25 39 38 46 32 22 41
	Rata-rata	748	130.2	
8	641	760	114	20 33 39 10 42 4 34 32 28 43 11 6 7 19 37 22 26 45 2 3 30 8 48 38 40 13 1 5 25 24 16 41 14 23 47 27 18 36 12 46 17 31 21 50 35 49 15 29 44 9
		758	132	23 27 42 22 33 34 44 1 7 15 21 17 38 16 46 43 19 48 12 13 35 4 24 45 5 47 8 9 2 11 10 30 36 20 40 41 6 14 28 26 37 32 49 25 18 31 39 50 3 29
		768	80	10 7 2 33 23 15 50 5 3 6 13 40 29 44 49 48 8 9 17 46 47 34 1 24 4 18 19 38 45 30 27 35 22 32 14 26 43 16 37 36 41 12 20 39 25 28 21 31 42 11
		759	131	16 25 7 48 17 23 39 49 33 20 10 34 12 6 8 37 44 1 45 27 35 22 47 18 28 32 26 3 46 11 5 38 40 41 14 21 2 13 24 31 42 19 15 4 36 30 9 50 29 43
		745	176	28 27 14 34 11 2 4 46 40 20 16 39 45 49 37 43 42 47 3 35 23 38 17 12 6 15 1 5 18 26 48 41 22 13 32 7 24 33 25 19 21 8 10 50 30 31 44 36 9 29
	Rata-rata	758	126.6	

Tabel C.16: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 5 proses, 5 mesin, dan n^3 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	595	711	57	36 5 18 28 3 40 9 29 47 27 45 25 11 17 41 16 20 22 39 48 7 42 8 49 6 37 21 50 13 34 12 1 30 4 31 32 38 46 35 14 19 15 10 2 26 43 33 24 23 44
		706	45	48 36 12 1 32 25 45 3 18 40 29 7 31 9 4 50 49 17 6 15 41 30 24 27 20 26 5 2 19 10 28 35 21 47 16 39 13 14 46 11 8 42 44 37 22 33 43 23 34 38
		717	118	36 44 34 9 33 45 5 40 46 37 29 28 25 47 12 11 18 6 24 1 42 27 14 35 31 41 50 21 7 3 15 48 17 20 38 32 39 30 13 10 16 43 49 19 22 4 8 26 2 23
		705	111	30 39 41 9 29 45 8 28 17 49 3 5 23 36 1 14 12 25 47 42 27 6 24 21 20 38 32 7 43 40 35 37 46 11 2 13 10 50 16 15 4 31 18 19 22 44 33 34 26 48
		711	164	38 39 29 28 20 33 5 3 18 10 36 32 42 9 40 13 17 24 30 31 1 49 45 2 15 6 7 25 41 16 47 50 48 46 19 44 35 12 26 22 34 27 37 21 14 11 4 23 43 8
	Rata-rata	710	99	
10	610	735	171	25 37 39 26 21 6 15 3 10 28 45 23 43 30 42 7 24 17 34 48 29 38 14 50 4 8 33 12 2 19 9 16 13 40 1 11 47 49 44 46 20 36 32 27 31 41 18 35 22 5
		737	116	26 3 48 35 27 5 6 12 49 41 38 2 23 14 47 8 21 37 31 19 34 24 7 1 44 30 40 45 46 13 42 22 28 43 33 4 9 17 15 39 10 11 32 16 29 50 20 18 25 36
		729	180	37 34 7 6 49 41 26 36 21 19 15 45 33 8 3 39 47 4 14 5 50 20 35 2 44 30 38 42 17 24 10 40 1 9 23 11 46 27 32 28 12 16 29 31 25 13 18 43 48 22
		735	144	35 26 3 24 34 6 10 4 39 21 49 17 1 14 5 20 33 29 32 46 2 50 12 42 31 16 8 27 30 13 23 7 47 44 41 43 38 36 48 45 11 22 40 28 19 37 25 9 18 15
		735	156	6 46 4 37 39 21 10 1 20 7 45 38 43 28 26 3 18 17 42 27 35 2 15 23 13 24 32 11 12 8 19 9 34 47 5 33 44 30 16 49 14 41 36 31 29 50 22 25 48 40
	Rata-rata	734.2	153.4	

LAMPIRAN D

DATA PENGUJIAN KASUS *HYBRID FLOW SHOP* DENGAN 50 PEKERJAAN DAN 10 PROSES

Tabel D.1: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	1125	1465	115	40 49 5 4 6 25 42 48 15 35 31 43 10 26 50 2 44 3 36 1 23 45 46 24 21 20 7 27 41 11 29 33 9 19 37 12 47 8 28 16 22 17 32 30 38 34 39 13 18 14
		1441	33	28 4 23 27 15 24 7 33 36 13 5 34 16 35 45 8 17 25 3 29 50 19 11 42 2 32 48 47 44 40 9 43 6 10 1 41 20 37 22 46 30 39 31 49 26 21 18 38 12 14
		1471	58	22 4 44 5 34 27 37 47 45 42 3 11 12 26 38 46 2 32 41 29 24 7 1 20 13 14 15 19 36 35 31 9 8 23 50 18 30 21 25 16 40 10 6 33 48 28 43 49 39 17
		1464	43	28 5 37 8 13 31 42 33 24 27 11 35 49 47 4 26 2 41 6 46 40 48 20 34 9 16 45 23 39 12 38 19 50 14 17 29 43 22 44 1 25 30 10 7 36 21 15 18 32 3
		1448	178	5 16 37 2 39 13 34 31 6 24 29 10 3 26 25 43 30 18 50 20 48 45 15 4 35 23 7 1 40 41 27 36 8 33 32 47 11 28 9 42 22 49 21 12 44 46 17 19 14 38
	Rata-rata	1457.8	85.4	
2	1165	1459	123	3 5 32 29 43 10 6 1 42 35 50 26 30 47 37 33 28 34 20 9 46 12 45 13 36 8 2 48 17 40 22 44 31 19 15 11 49 14 7 41 27 21 38 23 39 18 24 4 16 25
		1471	25	8 32 27 43 47 1 50 42 21 11 46 48 41 36 40 16 34 33 2 35 4 20 44 13 9 17 6 28 10 19 29 30 7 31 37 45 22 24 14 15 18 26 38 12 39 5 49 3 23 25
		1475	138	30 28 44 16 32 48 10 41 40 15 19 37 43 49 25 26 5 42 2 8 20 22 36 35 38 3 34 45 27 13 12 47 7 11 31 29 21 46 24 6 33 23 39 50 9 14 1 4 17 18
		1465	68	38 5 47 10 29 19 44 15 7 40 6 35 50 49 20 9 31 21 32 48 36 2 3 1 34 37 46 27 43 33 17 39 26 24 11 4 12 28 45 25 16 13 14 22 8 42 23 30 18 41
		1466	37	43 32 15 40 50 19 23 6 28 29 44 48 27 42 36 49 34 14 2 8 16 13 24 38 47 41 39 26 9 33 3 21 10 12 30 46 20 35 4 11 5 17 45 7 1 25 37 31 22 18
	Rata-rata	1467.2	78.2	

Tabel D.2: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	1235	1537	153	10 20 34 15 23 16 14 17 5 19 40 29 43 18 4 38 42 9 26 11 31 21 25 49 13 30 12 50 22 44 6 3 24 47 1 27 45 32 41 35 39 28 33 2 37 48 8 46 36 7
		1507	105	29 10 22 48 18 16 7 21 46 40 31 12 4 2 25 26 24 47 3 33 20 38 28 27 39 42 32 50 15 13 1 35 17 49 14 6 11 43 5 37 45 30 19 8 36 44 34 23 9 41
		1517	148	22 16 20 41 43 13 39 3 23 49 47 25 36 1 35 29 31 45 18 10 33 38 6 40 27 5 4 19 12 14 48 34 24 21 28 11 8 30 2 32 26 42 46 9 17 37 15 50 7 44
		1514	119	13 48 6 30 24 15 16 22 1 44 18 43 39 31 20 3 10 5 42 8 45 25 14 32 21 35 41 37 11 12 40 47 28 29 34 27 49 4 46 2 26 38 19 9 33 17 36 50 23 7
		1517	182	31 6 4 15 16 40 24 22 2 38 25 23 46 1 29 20 10 28 19 41 37 48 39 44 17 5 21 18 50 12 26 7 13 3 35 32 42 14 49 27 11 45 8 47 30 43 33 34 36 9
	Rata-rata	1518.4	141.4	
4	1274	1500	176	22 50 38 42 12 13 16 31 8 33 19 10 23 9 15 43 46 36 39 7 49 24 6 28 1 3 35 29 25 26 40 30 44 18 32 5 4 11 47 37 41 20 17 48 2 34 27 14 45 21
		1503	134	38 42 36 8 31 12 50 41 3 13 4 24 49 10 14 40 39 48 18 27 43 46 1 35 44 7 30 23 6 9 16 19 29 26 33 11 22 2 34 17 25 28 45 21 5 32 15 37 47 20
		1496	173	50 42 29 44 12 6 31 3 38 19 40 5 43 1 24 22 15 10 25 21 37 26 39 18 30 28 9 20 7 4 32 11 49 13 23 45 33 35 16 8 27 34 41 46 36 48 47 2 17 14
		1503	93	50 42 44 31 3 12 41 22 19 27 7 45 16 24 49 11 43 29 26 4 23 13 33 15 9 32 8 14 10 39 1 36 37 34 28 20 35 18 25 30 48 2 40 38 46 6 21 5 47 17
		1508	125	31 3 44 50 12 22 29 38 1 20 25 5 33 17 43 23 27 36 28 19 10 4 46 45 16 42 24 11 40 49 39 2 26 35 8 15 32 7 9 48 37 30 13 34 18 6 41 14 21 47
	Rata-rata	1502	140.2	
5	1143	1475	48	36 48 30 8 7 20 33 14 43 1 41 5 12 50 9 38 17 39 29 13 34 15 19 21 4 35 44 32 40 6 2 22 16 3 45 24 47 26 46 11 42 27 23 25 37 18 10 49 28 31
		1467	178	47 48 20 3 41 5 50 36 4 17 10 13 18 43 42 28 30 22 1 34 46 26 19 9 25 7 12 2 29 21 32 16 24 45 44 27 14 40 39 8 11 38 33 35 23 6 31 15 37 49
		1491	134	43 35 29 47 5 2 18 7 48 34 8 36 38 1 12 20 25 44 21 46 19 33 31 28 13 27 3 14 37 4 6 16 24 45 50 23 39 49 32 30 9 17 41 22 42 26 10 40 11 15
		1492	167	42 21 24 9 35 26 29 34 41 49 18 47 36 48 32 6 16 8 10 15 17 44 19 46 27 30 39 25 12 4 23 2 37 50 5 13 38 11 22 7 3 33 43 31 1 40 14 20 28 45
		1487	81	41 49 23 17 13 5 50 38 10 20 19 33 48 36 16 35 29 44 21 46 43 30 7 9 47 22 11 26 4 1 24 40 2 39 25 6 27 3 12 45 34 18 42 8 37 32 15 14 31 28
	Rata-rata	1482.4	121.6	

Tabel D.3: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^2 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	1191	1469	156	3 46 24 44 38 6 23 18 5 1 19 48 50 39 16 43 41 35 28 29 4 13 10 11 27 2 12 36 25 40 15 26 21 9 31 30 34 32 17 45 42 14 22 49 47 8 20 37 7 33
		1474	131	24 14 32 48 49 27 9 23 45 5 28 42 40 1 11 44 15 4 2 7 8 36 43 47 25 35 22 18 16 30 34 21 39 26 12 46 29 19 10 13 41 6 31 50 17 20 37 3 38 33
		1472	94	46 39 27 50 15 34 42 11 44 19 9 48 1 21 13 8 36 24 22 5 2 10 37 3 14 41 32 40 29 31 28 23 20 43 49 35 18 12 30 6 17 7 45 33 26 25 47 4 38 16
		1469	92	48 46 23 50 18 12 26 39 30 11 1 16 15 31 2 40 42 13 29 43 47 24 19 34 32 35 37 36 28 33 25 22 8 38 9 20 5 27 10 41 44 3 14 4 17 6 7 45 21 49
		1476	181	37 23 15 27 7 14 30 32 28 10 46 40 29 38 26 19 17 6 18 1 25 3 5 2 20 31 13 43 47 41 35 34 9 12 44 4 22 45 8 42 11 36 49 48 16 33 24 50 39 21
	Rata-rata	1472	130.8	
7	1126	1474	159	37 23 15 27 7 14 30 32 28 10 46 40 29 38 26 19 17 6 18 1 25 3 5 2 20 31 13 43 47 41 35 34 9 12 44 4 22 45 8 42 11 36 49 48 16 33 24 50 39 21
		1474	166	37 23 4 50 13 36 1 28 41 33 17 10 27 16 25 6 15 32 12 30 5 20 19 26 46 47 38 9 29 42 22 35 49 31 21 14 44 11 48 7 34 2 18 40 3 45 8 43 39 24
		1483	133	49 43 8 24 47 50 25 4 26 34 21 41 3 15 19 38 14 32 29 46 28 1 44 13 31 16 27 36 35 42 48 18 22 23 9 10 37 6 39 7 2 33 11 12 30 40 17 5 45 20
		1480	181	43 13 17 21 41 47 50 30 10 4 22 18 25 39 3 14 35 36 19 6 11 45 32 7 37 15 46 28 31 33 9 20 42 38 2 48 49 26 44 24 29 34 1 12 16 8 27 5 23 40
		1474	103	9 37 24 13 16 6 19 12 42 11 32 15 26 43 18 28 49 48 25 14 46 33 29 3 17 50 30 44 45 21 27 38 20 1 2 41 10 36 7 34 5 23 4 22 8 35 40 31 39 47
	Rata-rata	1477	148.4	
8	1215	1524	159	45 29 44 3 4 18 6 5 17 33 36 28 25 49 32 12 41 24 34 10 30 7 15 40 26 42 43 2 37 19 35 31 48 14 23 8 38 21 20 27 50 1 46 16 22 47 39 9 13 11
		1524	61	23 50 42 35 18 17 6 9 38 14 30 8 49 24 2 19 36 22 5 37 25 46 7 43 29 33 21 32 40 41 16 11 27 10 34 20 26 15 44 47 45 12 13 3 48 39 28 1 4 31
		1523	130	42 3 9 30 29 48 33 41 14 18 25 50 5 11 16 8 10 49 31 12 35 28 4 2 23 38 6 36 26 37 24 27 22 32 45 43 17 7 1 44 46 20 34 13 19 40 15 47 21 39
		1519	140	41 19 36 17 49 42 37 38 46 23 25 20 12 27 44 26 43 29 2 6 31 1 47 28 32 33 14 30 10 5 35 7 50 8 48 11 45 40 22 16 34 21 18 15 3 9 39 4 13 24
		1507	129	46 36 25 42 45 22 32 44 19 14 23 26 10 31 35 4 3 5 7 43 33 20 30 50 38 17 49 15 18 29 37 39 2 28 6 27 1 8 21 41 48 12 16 13 47 9 34 40 11 24
	Rata-rata	1519.4	123.8	

Tabel D.4: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^2 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	1145	1465	189	16 31 21 25 18 30 11 42 2 32 8 29 22 50 15 34 19 37 40 3 24 33 38 36 45 26 27 13 28 39 43 4 6 35 9 12 1 7 10 47 20 49 23 46 5 41 44 14 48 17
		1470	118	25 30 8 13 3 31 22 35 20 28 17 19 2 6 4 36 24 38 44 18 49 15 7 12 16 10 39 23 11 33 50 9 26 45 48 37 1 5 42 47 29 46 41 34 27 40 43 32 21 14
		1473	107	40 13 39 45 42 46 19 35 22 25 4 12 10 30 49 29 26 41 48 21 33 17 36 15 37 2 32 28 6 47 16 31 11 23 20 7 43 1 18 5 8 3 27 44 50 9 34 38 14 24
		1468	50	3 31 46 42 36 13 45 10 48 47 30 22 28 20 14 11 15 21 16 19 27 18 34 12 7 25 23 49 32 8 24 33 17 29 2 1 9 6 26 41 37 43 5 44 39 40 50 4 35 38
		1475	94	36 6 11 42 48 44 7 43 30 13 18 29 50 10 25 40 45 49 39 20 23 15 32 1 16 27 31 33 2 3 9 47 4 46 41 5 12 19 34 26 38 37 22 17 14 24 21 35 8 28
	Rata-rata	1470.2	111.6	
10	1230	1436	160	6 16 20 4 27 18 41 5 12 26 22 30 10 37 32 47 43 38 29 1 14 48 44 31 33 45 46 17 40 15 28 42 2 8 11 23 3 13 24 25 49 36 7 21 39 19 50 34 35 9
		1456	11	24 2 32 34 23 7 38 5 22 30 50 16 31 35 12 49 10 6 45 29 36 11 27 19 41 20 37 21 1 13 47 43 33 25 3 4 44 8 40 46 39 28 9 42 14 26 15 18 48 17
		1452	186	6 32 15 24 26 49 12 20 5 27 30 22 31 28 10 43 14 21 8 25 16 19 18 3 36 17 45 33 46 42 38 41 39 2 4 47 44 29 1 7 37 40 13 50 11 48 34 9 35 23
		1447	79	38 2 11 6 36 7 16 21 23 33 32 25 49 3 45 42 15 43 17 41 28 29 14 48 12 9 47 19 1 4 5 26 50 31 40 10 35 20 22 46 44 18 8 24 13 30 27 34 37 39
		1443	152	38 28 31 6 43 29 49 25 37 15 20 22 32 46 39 13 45 14 10 19 4 7 18 26 34 3 17 27 48 2 35 36 47 12 30 1 41 24 33 23 21 50 5 42 40 11 8 16 9 44
	Rata-rata	1446.8	117.6	

Tabel D.5: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	1125	1430	174	27 23 34 19 5 1 31 9 30 29 16 11 38 13 37 33 12 42 8 49 24 10 40 43 47 45 35 22 15 17 7 48 2 21 39 3 20 50 4 41 26 25 46 6 36 14 44 32 28 18
		1440	168	26 15 30 48 34 4 50 38 41 24 45 13 5 33 22 16 31 35 42 25 1 11 6 3 10 32 44 47 37 8 49 23 29 27 19 20 9 2 43 40 7 36 46 12 17 39 28 21 14 18
		1444	138	5 12 30 45 15 22 16 37 43 1 46 21 9 13 19 24 3 2 25 7 50 4 35 23 20 29 10 49 47 38 40 34 11 41 28 6 18 33 31 39 42 48 26 36 27 17 32 44 8 14
		1439	152	27 23 38 34 13 42 21 50 8 20 36 29 5 16 40 44 22 25 7 37 28 10 15 47 31 4 46 33 2 1 45 35 6 24 11 41 12 18 17 14 9 3 43 30 48 49 19 39 26 32
		1413	190	34 28 5 45 15 48 25 27 31 1 2 26 10 4 42 47 13 36 8 33 19 11 18 35 22 44 9 6 23 41 3 37 24 46 7 40 16 14 12 43 29 50 20 49 39 38 30 21 17 32
	Rata-rata	1433.2	164.4	
2	1165	1441	137	44 47 19 28 15 25 29 46 50 45 13 14 26 48 5 49 4 42 12 23 39 30 9 3 36 6 40 21 22 11 10 7 17 43 38 32 1 35 37 34 31 8 20 24 27 2 41 18 16 33
		1451	125	21 28 6 19 15 43 35 47 5 44 30 8 34 48 14 12 23 42 32 39 29 24 4 36 10 27 45 49 13 1 2 50 11 40 7 16 20 26 25 41 18 3 9 31 22 46 33 38 37 17
		1446	113	14 15 32 44 33 48 43 11 40 42 28 49 39 38 27 16 17 5 1 29 46 47 35 50 4 31 36 2 30 34 8 13 12 22 41 9 24 10 7 26 23 20 25 6 37 21 19 45 18 3
		1445	157	44 30 42 48 10 28 14 49 27 12 5 35 45 47 50 41 22 31 36 15 23 32 33 1 34 37 16 4 11 17 26 43 6 21 29 46 20 7 13 19 39 38 24 9 3 2 8 18 40 25
		1450	93	19 5 47 23 21 48 43 3 41 30 36 31 6 33 46 35 40 50 14 49 20 11 28 32 2 45 1 22 44 26 13 4 24 39 27 38 9 15 37 8 25 42 34 17 16 12 10 7 29 18
	Rata-rata	1446.6	125	

Tabel D.6: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	1235	1491	141	31 22 49 32 17 38 18 42 44 1 16 34 48 6 33 28 40 13 11 20 45 35 23 36 14 5 43 46 47 21 50 26 39 30 4 10 27 12 3 25 19 9 24 15 8 37 2 7 29 41
		1492	151	19 22 49 41 45 40 14 16 42 15 48 46 7 29 20 31 2 1 23 17 47 28 43 39 27 37 25 5 26 21 34 24 11 8 4 50 12 6 9 30 13 32 33 35 3 44 38 36 18 10
		1489	153	17 6 15 19 22 48 45 40 27 9 2 1 13 16 46 5 18 31 14 50 33 38 3 11 10 24 43 42 4 8 32 28 35 49 39 44 7 20 12 29 26 34 21 37 25 30 23 47 36 41
		1495	130	48 23 10 31 17 5 6 14 3 39 16 38 35 32 22 27 49 25 43 26 50 9 29 28 40 19 1 33 20 34 42 4 45 30 37 46 12 11 21 8 36 18 41 47 24 13 2 15 7 44
		1495	108	49 4 16 22 18 8 44 34 13 28 15 30 10 1 31 24 19 36 35 2 26 42 39 45 27 17 12 21 5 41 33 38 48 50 9 40 14 6 25 32 23 20 11 46 7 37 43 47 29 3
	Rata-rata	1492.4	136.6	
4	1274	1486	167	12 50 31 3 42 44 13 26 27 43 9 16 21 4 25 22 29 39 32 46 35 41 47 6 5 1 24 36 28 49 20 10 8 17 15 40 23 7 30 38 18 33 19 11 2 48 45 34 37 14
		1483	149	3 31 50 44 12 4 6 24 13 8 15 42 29 22 7 9 5 1 11 21 48 28 35 46 43 20 14 30 10 26 17 40 36 16 19 18 25 32 33 23 49 41 2 39 34 38 27 37 45 47
		1492	154	13 12 50 44 31 3 42 36 16 24 9 22 15 38 30 45 32 19 33 23 49 11 43 46 48 2 26 39 25 28 5 10 14 18 8 4 1 21 41 35 34 6 17 40 7 20 47 37 27 29
		1477	122	12 42 50 44 31 3 6 19 20 39 10 7 22 11 45 43 46 32 30 13 28 1 23 2 40 36 8 33 21 5 14 49 16 9 38 4 29 17 35 24 41 15 18 25 48 26 37 27 34 47
		1473	137	50 12 44 31 3 42 36 8 9 4 40 7 35 2 26 43 24 33 18 37 30 28 1 46 34 41 15 22 13 19 10 11 49 48 6 23 20 32 27 16 39 25 17 5 38 29 47 45 21 14
	Rata-rata	1482.2	145.8	
5	1143	1462	103	20 24 11 28 50 36 35 27 48 34 4 42 44 26 49 30 43 29 13 46 21 5 47 12 6 9 15 39 8 3 33 19 10 25 18 22 41 2 14 23 1 32 37 17 45 16 7 38 40 31
		1444	78	11 15 38 2 3 13 29 7 22 41 1 12 19 21 43 8 5 50 30 35 27 48 42 34 24 6 18 26 17 46 4 9 36 47 32 44 39 45 16 20 23 25 37 28 40 10 33 14 49 31
		1467	79	49 41 9 1 8 20 2 42 5 4 30 14 43 18 46 33 11 3 48 23 45 24 7 26 27 17 44 10 35 25 36 50 29 39 32 47 19 21 37 16 28 22 12 38 34 6 15 40 13 31
		1455	140	19 1 50 42 35 12 11 14 7 43 27 24 47 32 13 22 9 41 38 44 25 45 48 26 46 3 36 37 34 2 20 8 28 33 5 21 4 29 30 23 15 17 16 39 40 10 6 31 18 49
		1460	138	34 49 24 50 18 35 11 45 22 41 43 46 4 9 7 48 32 8 36 47 23 28 12 31 29 40 39 26 37 33 19 6 5 30 27 20 14 3 25 2 16 44 17 42 10 1 15 21 38 13
	Rata-rata	1457.6	107.6	

Tabel D.7: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^3 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	1191	1457	51	46 15 48 28 24 14 38 21 39 33 18 12 47 40 29 6 26 31 41 17 19 3 1 44 22 30 27 49 8 16 32 34 42 5 13 35 7 20 36 11 10 9 2 25 4 43 23 50 37 45
		1449	137	42 36 48 27 15 4 29 30 24 22 35 1 11 3 12 9 34 10 46 43 47 8 2 49 21 14 41 19 37 44 40 25 23 6 28 38 13 26 17 32 7 5 20 39 16 18 50 45 33 31
		1446	160	48 15 27 36 40 32 46 24 35 34 1 14 41 17 45 23 3 2 11 42 44 12 30 21 22 39 49 5 38 26 28 20 8 13 9 29 16 10 19 18 33 4 47 50 7 31 43 25 37 6
		1442	101	32 48 31 6 1 19 23 40 15 22 3 16 9 25 7 50 46 17 2 18 8 13 12 21 34 29 49 35 24 36 11 45 30 43 10 4 14 38 27 41 5 44 28 26 20 42 39 37 47 33
		1426	141	24 32 9 39 15 13 35 36 14 34 2 41 10 6 46 1 44 8 43 30 18 27 17 19 23 5 25 3 40 42 11 20 12 38 47 22 29 26 28 21 48 7 37 16 50 33 49 4 45 31
	Rata-rata	1444	118	
7	1126	1458	72	17 25 34 36 33 6 28 48 46 7 19 29 8 40 14 13 42 43 11 45 15 37 9 38 26 21 47 18 12 2 22 35 50 1 44 30 16 23 32 10 5 49 20 3 4 31 39 24 41 27
		1450	177	13 34 37 38 19 3 10 25 17 42 6 14 11 44 43 27 15 7 23 28 32 18 16 1 47 4 31 41 36 26 33 9 2 50 48 20 49 21 12 46 30 29 22 5 40 45 8 35 24 39
		1441	174	17 13 36 15 27 47 18 30 28 9 19 26 42 41 50 22 34 25 38 2 21 37 40 39 29 1 5 11 32 23 44 48 14 10 35 43 4 3 7 6 24 12 16 33 8 45 49 31 20 46
		1438	141	2 37 13 9 47 27 29 11 32 15 3 7 25 17 18 12 19 4 49 34 1 26 31 24 22 6 40 50 35 8 38 23 28 48 16 42 30 44 36 41 43 21 46 33 5 14 10 39 45 20
		1447	122	37 36 3 8 25 48 17 14 33 15 47 43 32 42 31 46 40 18 19 9 29 2 50 26 24 34 28 30 10 1 5 41 7 6 49 16 13 38 21 22 35 44 23 12 11 4 45 39 20 27
	Rata-rata	1446.8	137.2	
8	1215	1489	153	3 41 18 46 23 25 9 38 28 10 32 43 5 24 1 36 34 33 45 6 11 48 30 29 35 17 7 27 15 47 50 39 8 13 49 19 20 42 44 2 31 16 12 14 40 4 22 26 37 21
		1488	97	29 1 27 18 36 3 25 48 38 39 10 41 17 30 32 26 15 43 23 8 5 21 44 4 6 47 2 12 14 42 31 19 46 45 33 7 49 35 50 20 16 28 24 37 22 9 11 13 40 34
		1490	105	26 29 38 27 33 30 14 48 39 7 3 23 22 18 46 15 28 6 44 34 43 5 10 35 49 8 42 21 41 36 25 12 9 19 24 16 17 1 47 50 20 32 2 40 4 31 13 45 37 11
		1499	74	10 29 39 49 27 46 5 24 25 15 7 26 38 41 1 14 35 8 17 36 45 47 34 43 42 20 3 23 16 6 50 30 33 48 9 19 2 28 21 18 4 12 44 32 40 11 22 31 13 37
		1497	88	29 49 50 27 44 16 4 47 48 31 5 42 33 6 11 9 17 46 23 36 43 12 3 30 35 26 18 40 20 19 7 8 38 41 1 39 25 2 10 14 32 22 28 13 24 15 34 45 21 37
	Rata-rata	1492.6	103.4	

Tabel D.8: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 3 mesin, dan n^3 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	1145	1447	165	5 1 36 31 39 34 6 40 45 11 28 10 7 4 38 27 42 44 13 35 47 26 8 17 15 24 48 29 19 9 43 22 33 2 32 46 12 50 20 16 25 41 49 3 18 30 23 37 14 21
		1447	125	6 14 31 13 21 40 20 46 4 30 25 10 3 47 7 49 8 50 15 23 22 1 2 32 45 44 26 16 27 35 19 17 11 29 34 18 43 39 42 28 9 33 48 24 12 41 36 38 5 37
		1439	152	36 40 11 38 31 21 47 48 12 42 22 30 49 15 6 35 20 28 46 5 43 19 2 26 32 4 27 10 9 50 45 23 1 7 25 33 8 3 18 16 17 34 29 41 44 24 14 39 37 13
		1428	167	50 25 31 18 40 35 19 34 46 5 24 41 6 4 13 32 49 17 10 20 15 33 22 2 26 3 1 12 39 16 28 47 7 30 37 8 44 45 23 48 36 29 27 43 9 14 42 21 38 11
		1429	161	24 4 31 36 21 39 15 10 43 28 19 47 7 16 35 2 49 18 14 23 11 9 20 13 26 6 46 42 27 25 8 40 32 50 12 45 48 44 22 38 33 29 41 17 1 5 3 34 30 37
	Rata-rata	1438	154	
10	1230	1435	121	26 30 25 16 50 15 19 4 1 41 45 34 11 40 46 10 29 6 7 31 44 24 43 18 42 17 2 36 20 32 22 28 5 3 21 38 33 13 14 23 8 39 47 35 49 37 12 48 27 9
		1430	155	3 24 14 7 20 22 15 49 43 50 6 32 36 33 26 44 41 47 31 34 28 5 10 25 45 1 42 16 30 8 29 35 37 12 19 46 11 17 21 13 18 23 4 38 39 2 48 27 40 9
		1437	127	29 18 6 32 38 10 47 37 21 36 49 12 4 41 33 39 3 26 40 19 43 28 7 17 45 2 25 35 30 22 16 27 14 15 20 50 46 8 24 5 31 11 48 13 23 1 34 42 9 44
		1428	142	6 47 20 41 15 5 50 38 32 7 37 18 29 26 44 2 33 3 31 10 43 30 40 46 34 16 21 22 14 12 28 24 27 13 25 45 11 4 8 17 35 23 39 19 42 36 1 48 9 49
		1430	167	28 11 23 41 42 45 30 33 5 44 8 18 32 34 36 50 16 31 10 20 46 6 43 9 29 13 24 21 47 25 35 27 2 15 7 37 22 12 26 3 38 4 17 19 1 14 48 39 40 49
	Rata-rata	1432	142.4	

Tabel D.9: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^2 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	864	1073	179	12 40 25 5 2 1 33 16 49 15 9 34 50 43 37 7 23 21 22 14 18 44 4 28 42 11 39 27 32 17 29 46 31 47 6 13 3 10 35 26 38 45 20 36 48 24 8 41 30 19
		1077	53	24 37 31 44 49 11 3 46 13 15 14 2 9 23 42 32 22 6 38 33 17 1 5 7 18 27 45 30 47 26 35 16 19 34 21 20 43 4 39 36 29 41 25 8 48 28 12 50 40 10
		1083	116	28 40 42 31 45 1 6 27 11 34 23 35 17 15 5 32 13 9 25 7 18 29 21 48 44 39 36 30 3 10 2 46 37 33 8 26 4 38 19 20 14 22 24 12 41 47 49 16 50 43
		1086	17	24 14 18 49 1 3 39 6 30 46 33 47 37 12 40 11 38 16 21 29 2 35 34 36 50 13 32 27 5 45 28 8 23 7 15 26 9 31 44 4 20 41 10 48 25 17 43 22 19 42
		1079	88	42 4 46 5 45 16 2 28 21 49 15 22 11 30 24 48 43 9 37 31 38 41 7 29 10 35 39 27 1 13 25 32 47 6 20 3 18 14 40 12 36 19 34 50 26 8 23 33 17 44
	Rata-rata	1079.6	90.6	
2	846	1036	41	19 27 31 22 11 49 41 26 13 46 21 50 45 18 8 23 44 15 3 10 38 42 6 28 17 29 47 12 40 25 2 4 43 16 20 37 35 14 39 30 24 48 7 33 1 36 34 9 32 5
		1033	76	13 8 36 20 37 48 11 40 9 14 19 28 18 27 50 43 29 25 5 4 31 30 49 23 6 34 24 16 39 47 10 41 3 44 15 38 7 46 35 22 21 32 17 12 42 26 2 45 33 1
		1046	139	49 13 45 6 35 43 3 8 17 41 46 11 14 27 23 4 44 20 28 26 47 24 19 5 38 32 7 18 9 22 25 16 15 37 50 33 30 39 40 2 42 48 12 31 21 10 34 36 1 29
		1041	153	46 36 20 45 11 4 42 48 38 23 28 12 18 37 24 6 40 50 9 44 14 27 25 16 10 3 35 22 5 34 43 13 8 49 47 15 19 17 39 21 30 1 26 2 41 33 31 32 7 29
		1048	110	37 49 43 32 45 18 4 33 29 16 40 3 15 39 19 20 22 7 28 13 11 47 36 31 35 44 24 14 2 42 48 41 46 27 5 8 23 26 50 25 9 6 12 38 30 17 1 10 21 34
	Rata-rata	1040.8	103.8	

Tabel D.10: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^2 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	890	1024	182	24 36 49 8 5 14 47 11 22 43 37 33 44 34 35 10 28 25 45 50 6 42 21 2 30 18 39 9 1 32 29 48 38 7 19 15 17 26 20 16 40 12 27 4 3 13 23 46 41 31
		1011	29	18 47 45 36 1 32 28 37 4 35 6 26 25 31 2 34 19 21 11 22 39 14 50 8 10 12 27 16 48 42 44 7 43 9 24 49 5 29 17 23 40 30 41 20 13 3 38 33 46 15
		1022	54	40 50 17 14 2 34 38 5 49 46 8 32 15 43 9 36 29 44 4 16 10 47 18 27 45 39 22 12 26 37 42 28 3 30 11 21 7 24 6 1 35 19 25 20 33 13 41 23 31 48
		1018	52	2 1 7 37 16 38 5 8 18 47 15 50 29 48 35 40 22 42 32 39 14 34 3 12 44 28 13 31 26 49 9 21 46 43 11 19 25 6 17 33 30 10 24 20 27 36 45 41 4 23
		1024	32	1 36 46 30 27 44 22 43 16 2 39 9 14 33 47 32 25 8 12 49 40 18 21 11 3 4 23 35 26 10 19 5 38 17 37 34 7 24 29 45 28 41 13 42 6 31 20 50 48 15
	Rata-rata	1019.8	69.8	
4	841	1034	107	38 49 47 18 41 28 15 43 5 31 35 30 36 23 44 22 4 50 2 33 3 26 6 20 12 32 16 21 45 19 13 48 17 42 40 27 34 25 11 1 37 7 9 14 24 46 10 29 39 8
		1025	113	38 18 1 11 43 32 49 37 28 36 15 31 41 26 46 42 21 8 4 23 48 47 22 40 20 50 13 24 45 3 27 5 19 44 39 7 12 30 35 25 29 2 34 6 14 9 16 17 33 10
		1032	59	1 41 13 22 14 38 16 44 17 4 42 3 21 2 27 40 23 34 50 26 6 7 36 19 12 33 11 20 47 25 15 48 9 43 49 18 45 5 28 32 37 35 30 8 29 39 31 46 10 24
		1031	100	30 38 19 12 28 35 40 47 27 24 23 50 31 4 22 13 36 44 49 37 20 26 3 48 45 21 11 33 18 42 41 25 5 46 17 32 1 9 7 16 34 2 14 29 10 43 6 39 8 15
		1031	175	32 38 21 20 28 36 43 8 11 49 12 50 37 1 27 23 22 26 18 7 41 44 48 9 5 4 46 33 35 15 39 42 14 47 45 6 2 25 19 24 16 3 13 17 29 10 40 31 30 34
	Rata-rata	1030.6	110.8	
5	819	1001	180	5 13 2 9 50 3 1 8 36 49 14 7 33 34 30 44 27 48 10 21 31 47 23 25 40 42 43 46 4 16 32 24 35 12 15 11 39 18 28 22 17 26 29 19 41 38 45 20 6 37
		1015	99	25 31 10 8 48 41 47 36 33 23 2 1 30 40 3 14 39 34 27 4 16 9 24 15 38 46 43 13 44 35 50 28 37 45 18 12 19 22 49 17 6 11 21 26 7 20 29 5 32 42
		1017	93	33 46 12 14 10 32 13 47 2 15 35 28 25 23 40 1 44 36 5 49 9 3 48 34 41 50 4 37 24 16 20 6 30 27 31 43 38 8 17 19 45 22 21 26 39 18 42 7 11 29
		1024	141	21 38 50 44 33 40 13 4 1 47 12 25 23 16 10 17 5 27 7 14 49 37 42 24 30 15 6 36 46 48 18 22 34 43 29 11 2 32 39 3 35 9 31 8 28 26 45 19 41 20
		1020	97	22 18 13 10 12 8 44 28 9 1 41 3 5 14 36 21 26 48 50 30 43 25 33 23 32 6 49 20 4 29 35 27 38 2 16 24 40 34 46 47 11 37 17 31 42 15 19 39 7 45
	Rata-rata	1015.4	122	

Tabel D.11: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^2 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	803	1032	178	11 34 45 10 25 1 2 17 33 47 4 50 16 13 37 9 36 18 35 31 8 48 22 43 15 27 42 46 5 32 49 30 19 41 20 39 26 6 3 23 14 24 7 29 38 21 12 44 40 28
		1035	68	25 17 9 18 40 26 12 30 6 33 47 42 4 35 32 34 44 13 1 5 36 2 16 8 14 27 19 10 37 15 11 23 45 49 50 21 43 3 48 41 29 28 20 7 24 46 38 39 31 22
		1031	120	40 42 10 4 18 26 27 2 49 23 36 35 9 16 3 43 21 31 46 37 32 8 13 25 7 1 45 48 6 12 14 28 41 15 17 11 34 20 33 5 50 30 38 22 44 39 47 29 19 24
		1032	102	45 4 47 18 6 13 26 22 29 10 1 19 3 5 41 24 25 8 48 40 7 31 32 11 37 43 23 33 44 36 15 16 21 46 14 50 30 35 49 17 28 34 27 20 9 38 42 12 2 39
		1038	166	21 18 39 30 36 19 26 37 17 49 45 32 6 14 47 42 1 11 22 34 44 4 8 40 15 31 24 25 33 9 50 46 13 23 20 16 3 41 5 7 43 2 12 29 27 38 28 35 48 10
	Rata-rata	1033.6	126.8	
7	842	1054	113	40 19 26 12 11 42 8 1 10 9 31 36 34 41 30 21 48 49 4 7 47 33 23 13 50 2 38 5 37 46 35 15 44 45 3 16 39 18 6 25 17 32 28 14 24 43 27 29 20 22
		1054	152	6 33 30 44 43 5 20 23 32 3 1 27 17 10 46 8 39 49 4 7 25 2 38 9 35 28 14 41 48 37 13 45 16 22 40 26 19 36 11 21 18 34 15 47 24 12 42 50 31 29
		1050	124	36 42 19 31 27 17 15 3 26 49 6 39 9 30 5 38 1 29 44 48 2 50 32 25 47 46 4 7 45 40 20 35 16 21 8 22 23 41 12 28 18 13 11 10 37 24 33 34 14 43
		1058	127	36 11 20 27 9 31 30 34 25 26 50 49 3 48 15 32 35 10 19 28 39 41 4 7 37 46 45 12 40 33 16 38 47 8 18 44 5 6 1 22 13 21 42 23 24 2 17 43 14 29
		1049	123	21 11 8 28 40 47 9 6 42 49 27 16 4 7 1 32 41 45 35 31 5 18 46 37 29 48 2 15 36 26 34 17 13 23 20 30 39 3 25 14 33 44 43 22 19 24 10 50 12 38
	Rata-rata	1053	127.8	
8	831	987	100	6 2 19 4 33 9 41 7 18 27 10 3 1 24 15 34 42 23 29 46 47 43 21 30 39 37 13 17 26 36 49 14 32 20 40 35 48 25 45 31 44 38 8 50 12 22 16 5 11 28
		991	124	3 20 41 18 4 7 24 31 44 28 36 38 29 35 17 27 40 1 46 39 13 14 48 25 50 21 8 32 6 26 34 43 9 15 37 33 19 11 45 2 47 16 49 5 12 23 22 42 10 30
		987	156	19 27 33 13 4 41 7 1 24 16 44 28 6 50 14 35 47 20 2 15 9 17 40 46 31 45 42 25 5 34 37 12 38 30 26 48 18 39 10 29 49 11 32 36 3 43 22 8 21 23
		988	145	10 50 4 11 19 20 39 41 31 16 17 24 9 46 1 7 3 38 12 42 36 23 28 45 47 30 22 2 34 18 40 35 48 6 33 21 29 15 44 37 32 8 13 27 14 5 43 26 25 49
		990	139	19 20 48 3 10 4 45 35 2 7 1 24 29 15 42 34 11 22 27 44 31 9 46 17 40 18 41 25 21 38 6 39 37 13 16 32 28 50 33 47 49 14 43 12 30 26 8 23 5 36
	Rata-rata	988.6	132.8	

Tabel D.12: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^2 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	830	1040	68	8 14 26 23 6 12 46 33 22 1 2 25 44 43 38 39 18 24 28 48 21 42 20 34 7 40 13 3 30 27 5 36 37 47 41 31 11 19 17 45 29 10 16 4 49 9 35 50 15 32
		1024	167	48 29 6 3 42 44 45 15 40 47 27 16 36 19 20 39 24 41 31 22 7 8 4 21 17 46 5 10 37 38 34 26 23 43 14 12 2 50 18 13 25 49 1 9 28 32 35 33 11 30
		1036	45	44 30 6 22 1 17 12 34 24 14 39 8 18 13 3 23 28 5 47 35 48 38 26 20 10 27 41 37 19 25 46 50 31 9 43 42 7 4 15 32 29 45 11 49 16 36 21 2 33 40
		1030	102	8 6 3 42 48 30 7 12 25 13 24 10 14 39 44 40 5 23 26 33 38 34 37 21 20 35 47 22 29 11 18 36 43 19 46 31 4 1 9 16 28 41 27 49 17 45 2 32 15 50
		1039	115	42 40 30 24 26 4 6 3 23 39 29 45 10 12 25 1 15 33 13 37 34 7 44 35 47 19 48 28 46 31 38 18 20 49 5 22 17 36 14 43 21 9 16 11 27 8 32 41 2 50
	Rata-rata	1033.8	99.4	
10	897	1090	132	48 2 28 15 7 12 25 37 41 49 4 35 20 16 47 9 34 23 17 46 3 11 19 33 30 32 1 31 10 43 8 36 13 26 44 5 39 27 6 50 24 14 22 45 18 40 29 21 38 42
		1106	124	43 7 13 27 31 14 15 46 50 37 2 8 29 26 34 6 47 16 41 45 32 28 44 22 12 23 36 20 30 25 35 19 9 3 17 10 5 4 49 18 39 33 24 42 21 38 40 11 1 48
		1101	126	37 2 14 35 15 8 43 16 50 7 30 3 13 29 21 4 42 9 24 49 23 19 17 33 25 18 34 32 20 27 47 5 28 1 31 26 41 40 44 36 6 10 39 22 38 11 12 48 45 46
		1106	151	16 7 49 25 6 41 19 34 8 13 43 21 12 2 35 14 48 3 11 18 31 45 4 15 44 9 23 40 47 17 20 32 26 5 22 33 37 28 29 46 30 50 36 10 24 1 27 42 38 39
		1091	170	33 27 13 50 49 28 11 23 19 21 8 7 14 18 35 12 29 25 10 20 22 17 43 48 44 16 38 34 9 37 36 15 47 2 41 39 30 42 3 4 6 32 26 40 46 5 31 45 24 1
	Rata-rata	1098.8	140.6	

Tabel D.13: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^3 semut (1)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
1	864	1064	155	42 50 28 14 25 20 34 43 3 46 5 32 45 18 13 29 11 12 24 7 40 15 44 22 37 27 4 8 35 33 39 9 31 23 1 36 17 21 26 2 16 38 49 19 6 30 48 47 41 10
		1064	142	49 39 18 9 37 23 24 45 28 25 41 14 20 8 50 16 29 4 44 5 31 27 33 35 26 11 46 38 42 1 15 13 10 3 21 40 12 7 6 22 36 30 19 43 34 47 2 48 32 17
		1059	185	18 49 24 6 43 42 22 7 36 31 27 46 50 29 30 16 35 21 33 28 25 11 14 39 15 13 44 4 23 26 32 3 1 5 34 8 38 9 48 47 19 40 45 12 37 20 41 10 2 17
		1064	151	31 44 29 6 32 24 38 10 18 3 35 42 45 34 13 37 27 25 21 23 16 9 30 14 5 49 22 4 12 8 2 11 46 7 33 39 15 20 1 28 17 36 43 48 19 40 41 47 26 50
		1065	150	24 28 14 13 15 31 45 37 27 46 4 36 8 39 6 35 21 9 25 23 44 26 11 32 38 33 3 22 40 29 7 12 49 50 17 5 1 10 43 16 48 2 19 34 30 41 47 18 20 42
	Rata-rata	1063.2	156.6	
2	846	1022	105	22 45 11 46 34 14 5 16 49 24 50 35 15 28 23 27 31 39 37 4 48 13 36 3 42 29 20 9 43 12 2 19 18 40 47 44 26 6 1 8 25 10 33 38 30 17 7 32 41 21
		1035	92	31 4 36 15 21 33 11 42 32 16 19 14 43 49 7 20 12 26 22 46 27 24 47 50 48 3 35 44 8 37 41 1 28 40 23 39 10 6 29 17 18 38 2 34 25 5 13 45 30 9
		1030	78	12 48 40 4 31 28 45 2 3 11 46 9 14 49 39 6 7 22 8 23 33 47 43 29 38 25 36 50 15 18 13 24 16 37 21 27 19 44 35 32 10 20 1 5 17 34 26 30 41 42
		1033	84	2 11 21 48 49 23 15 39 7 6 40 28 10 26 20 3 47 32 41 16 29 44 9 43 12 35 13 19 31 4 37 24 14 50 8 33 22 27 17 5 25 46 42 1 30 18 36 38 45 34
		1025	144	36 45 50 18 27 3 37 11 49 44 35 15 32 5 48 31 24 7 28 8 30 26 12 10 4 42 43 46 21 20 14 19 25 47 39 40 6 41 23 9 13 22 29 33 16 17 38 34 2 1
	Rata-rata	1029	100.6	

Tabel D.14: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^3 semut (2)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
3	890	1009	52	47 37 2 1 50 48 23 8 4 10 28 33 7 14 21 12 44 16 35 9 11 18 15 34 19 36 38 46 17 29 43 32 27 20 39 3 22 30 26 31 24 40 13 25 6 5 45 41 42 49
		1007	105	37 17 43 8 18 16 40 2 14 34 1 29 7 42 48 28 35 12 46 39 9 47 38 30 22 36 21 5 31 32 44 6 3 11 33 26 10 19 24 45 15 13 50 27 20 4 49 41 25 23
		1003	129	8 1 42 47 10 27 6 20 37 24 32 38 28 41 12 40 36 4 17 33 43 19 2 11 14 22 9 39 34 18 30 7 16 46 35 21 48 5 44 50 26 15 31 29 13 23 3 45 25 49
		994	143	8 44 1 34 2 18 47 14 16 19 41 29 50 32 26 35 7 24 5 43 39 37 10 21 9 12 40 49 36 11 33 42 27 28 31 17 6 22 30 3 48 25 4 45 46 20 13 15 38 23
		1008	158	2 18 6 14 50 13 32 7 36 12 19 33 1 22 25 10 39 47 8 24 43 11 34 37 9 21 16 42 28 41 27 29 40 45 44 30 46 35 15 23 4 17 26 3 48 20 49 31 38 5
	Rata-rata	1004.2	117.4	
4	841	1021	165	23 5 22 1 18 13 19 39 45 40 11 50 43 48 7 2 20 3 16 42 47 29 17 26 38 49 34 41 36 44 35 12 33 24 27 30 31 6 14 4 10 37 15 25 8 9 46 32 28 21
		1006	107	16 21 32 6 22 1 37 20 49 31 39 26 43 28 19 48 27 13 23 3 47 25 5 29 33 36 41 12 45 35 10 15 17 50 34 18 42 7 2 38 44 11 40 24 4 8 9 30 46 14
		1020	169	30 20 23 43 2 22 21 34 38 12 35 4 37 16 47 41 49 7 36 42 19 6 46 50 1 25 40 18 11 45 27 26 31 17 32 15 48 3 24 44 13 33 8 28 9 5 39 29 14 10
		1008	125	28 22 31 19 1 38 47 49 34 18 5 26 46 23 4 36 43 13 33 12 37 20 50 48 27 40 17 45 30 35 2 41 32 42 44 3 8 9 14 24 11 6 7 25 21 10 29 16 15 39
		1009	118	38 1 27 43 42 26 23 31 3 19 32 29 40 5 47 22 2 4 36 16 25 44 20 21 18 50 48 37 8 12 11 30 7 46 41 15 6 35 49 34 45 13 39 17 14 33 28 9 24 10
	Rata-rata	1012.8	136.8	
5	819	997	140	20 10 40 1 30 16 3 43 38 2 44 27 47 14 34 49 11 32 46 22 39 13 50 25 4 9 23 48 5 28 31 26 6 33 12 36 42 18 15 24 35 29 17 7 21 37 41 45 19 8
		1003	104	21 47 2 10 50 14 49 33 48 3 44 23 35 25 36 11 38 16 40 34 39 46 7 42 28 12 17 1 27 26 5 15 37 20 32 24 45 43 9 31 13 41 30 4 8 22 19 29 18 6
		997	114	10 31 33 13 32 48 38 3 46 37 39 50 5 49 29 35 47 14 17 34 23 2 20 12 15 44 21 7 9 18 25 40 28 30 24 36 4 27 43 26 16 42 6 8 45 41 1 22 19 11
		995	111	32 41 33 40 25 35 3 2 22 46 1 18 23 12 30 34 10 47 48 26 16 39 43 17 5 14 50 15 49 42 27 44 9 21 45 13 36 38 8 28 24 4 37 29 20 11 6 31 7 19
		1001	126	42 49 32 13 2 44 48 35 30 1 50 14 34 23 25 29 10 21 17 15 22 24 40 33 28 5 16 12 3 38 47 39 4 8 9 7 36 43 20 19 27 31 46 45 26 41 37 11 18 6
	Rata-rata	998.6	119	

Tabel D.15: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^3 semut (3)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
6	803	1005	169	50 9 45 25 11 42 35 34 4 14 26 10 13 23 27 5 44 49 15 30 31 43 41 32 28 22 21 16 2 7 18 37 40 46 20 39 38 3 17 1 36 8 47 19 33 6 24 48 12 29
		1014	129	26 45 28 21 36 18 42 7 43 34 5 41 40 14 32 46 23 29 15 48 9 37 44 49 47 22 6 50 20 33 16 31 11 30 25 2 13 35 3 27 1 24 8 38 12 10 4 39 17 19
		1018	73	17 13 1 5 40 30 9 16 47 19 7 3 34 50 43 45 26 28 46 8 25 20 27 10 22 21 42 4 49 41 15 37 32 31 33 11 38 18 35 44 14 23 2 36 6 24 39 29 48 12
		1012	180	26 4 49 47 36 43 33 22 6 38 45 50 32 13 40 37 44 3 35 41 1 18 46 30 34 5 28 25 7 23 8 16 15 9 11 31 20 14 2 10 19 42 48 12 21 27 29 24 39 17
		1017	121	6 26 45 19 11 40 20 27 18 44 22 25 46 33 8 4 42 35 37 32 43 23 31 9 39 36 1 49 30 13 16 41 15 5 47 48 28 14 12 21 7 34 3 38 2 50 29 10 24 17
	Rata-rata	1013.2	134.4	
7	842	1038	166	31 32 28 1 23 5 25 9 30 11 39 35 8 19 26 49 46 10 15 36 22 40 42 4 7 3 18 24 44 21 37 13 45 27 17 43 2 50 47 16 34 41 20 48 6 14 38 29 12 33
		1033	135	27 44 48 11 5 42 47 24 35 1 18 19 15 28 23 13 32 25 49 26 40 6 41 4 7 16 10 38 34 8 2 39 9 31 45 20 36 33 43 21 12 3 46 17 29 14 50 22 30 37
		1036	144	37 26 30 10 28 46 40 21 1 23 2 32 44 12 11 36 27 48 42 4 7 25 19 49 38 16 47 29 24 17 41 8 9 13 45 20 43 5 34 3 31 14 35 18 39 33 15 6 50 22
		1028	106	40 5 21 28 46 1 3 2 20 50 6 45 32 11 36 33 12 9 7 49 25 13 16 34 30 39 38 23 4 8 42 47 37 18 15 31 26 44 22 48 41 10 24 43 17 35 27 29 19 14
		1029	155	6 39 2 46 11 20 28 40 7 31 32 49 13 9 36 25 50 1 41 24 27 8 3 34 22 17 30 4 12 5 35 45 19 23 16 18 26 47 10 44 14 43 42 48 37 38 15 33 21 29
	Rata-rata	1032.8	141.2	
8	831	978	151	11 13 7 41 16 20 3 10 47 28 25 6 31 44 17 24 15 34 2 9 46 29 50 1 18 35 40 19 43 21 48 27 22 38 39 37 42 36 32 26 45 4 14 49 12 23 5 8 30 33
		973	131	22 38 33 6 34 41 4 27 7 18 31 19 39 11 42 13 1 23 29 24 15 16 2 35 17 40 14 3 37 46 9 25 20 32 47 45 44 36 5 21 8 48 26 12 43 30 49 28 10 50
		983	155	42 4 27 3 33 41 28 32 14 16 17 24 15 1 44 2 6 13 34 40 35 29 30 20 37 11 49 9 46 7 18 5 22 31 23 26 21 47 8 38 19 48 39 45 36 12 43 50 25 10
		982	144	7 27 4 32 33 20 31 35 17 46 41 2 34 36 39 25 40 1 28 13 19 45 24 15 3 29 18 10 5 8 47 48 37 49 16 44 43 11 12 22 9 42 21 30 50 38 26 6 23 14
		987	137	33 16 12 27 22 31 3 45 4 40 10 23 19 17 43 48 14 9 46 34 1 24 15 29 18 7 6 20 41 37 49 25 44 35 39 42 50 26 8 47 2 11 38 36 32 21 13 28 30 5
	Rata-rata	980.6	143.6	

Tabel D.16: Tabel data hasil pengujian kasus *hybrid flow shop* dengan 50 pekerjaan, 10 proses, 5 mesin, dan n^3 semut (4)

Test Case	*UpperBound*	*Makespan*	*Training*	Urutan Pekerjaan
9	830	1022	105	34 22 48 16 18 49 23 36 44 10 29 46 17 39 13 14 43 12 45 47 38 8 3 6 40 4 24 27 20 32 26 37 42 35 41 11 7 50 5 30 28 25 31 15 1 21 2 19 9 33
		1024	103	28 44 17 1 36 48 32 37 43 3 42 6 41 8 25 31 21 5 35 40 30 26 46 49 22 38 39 34 10 14 4 50 29 12 7 18 23 24 27 13 2 20 16 47 15 9 19 33 45 11
		1026	97	50 1 48 10 4 30 47 31 37 13 12 25 24 23 39 8 6 3 42 34 43 21 36 9 28 35 44 5 20 26 22 38 14 29 32 33 45 15 18 7 46 41 49 27 17 2 16 11 40 19
		1018	134	30 42 3 34 8 13 38 24 31 6 4 45 33 5 23 18 10 35 11 50 14 22 21 44 43 40 39 20 46 41 1 48 2 26 32 25 37 7 47 9 27 29 16 49 12 36 28 15 17 19
		1024	147	44 48 17 47 43 15 1 21 22 29 23 6 3 40 46 30 16 8 12 26 34 39 41 27 10 36 14 37 13 38 42 50 5 20 24 25 7 18 33 4 32 9 49 31 11 35 19 2 28 45
	Rata-rata	1022.8	117.2	
10	897	1089	79	24 49 48 41 30 5 50 2 15 29 28 9 13 16 37 8 33 25 27 36 47 23 31 43 7 14 12 44 26 19 4 3 34 18 11 40 17 45 6 35 10 20 32 22 21 39 38 42 1 46
		1090	70	26 8 18 50 16 7 49 5 32 20 15 43 34 36 48 47 2 23 29 9 28 45 22 30 19 4 17 44 37 6 14 13 35 40 3 11 33 41 31 12 38 27 42 21 10 39 46 24 25 1
		1073	109	7 49 2 15 28 14 16 5 48 8 43 37 33 50 27 17 6 9 19 10 44 47 26 30 31 29 12 25 22 42 32 36 18 23 11 3 34 41 20 38 45 4 35 40 1 13 39 24 46 21
		1089	72	9 39 12 28 17 50 22 7 40 23 43 21 37 41 42 48 14 27 20 16 26 36 18 15 47 5 44 24 1 49 8 19 3 35 34 4 33 2 6 31 25 45 11 32 30 10 38 13 29 46
		1090	70	12 7 45 39 48 14 27 17 13 22 9 40 44 4 35 47 26 50 41 37 20 43 15 46 23 6 32 49 33 36 18 25 28 1 3 34 16 19 29 24 21 42 2 30 38 31 8 11 10 5
	Rata-rata	1086.2	80	

LAMPIRAN E

THE SOURCE CODE

Listing E.1: Problem.java

```
/**
 * Kelas ini merepresentasikan salah satu persoalan hybrid flow shop.
 * Kelas ini berisi detail-detail mengenai banyaknya job, banyaknya proses, banyaknya mesin pada masing-masing proses, informasi masing-masing pekerjaan.
 * Kelas ini juga berisi upper bound dan lower bound yang berfungsi sebagai batas untuk menghitung keoptimalan penjadwalan hybrid flow shop.
 *
 * @author ALEX
 */
public class Problem {
    private int   jumlahJob;     /*banyaknya job pada permasalahan hybrid flow shop*/
    private int   jumlahProses; /*banyaknya proses pada permasalahan hybrid flow shop*/
    private int[] jumlahMesin;  /*banyaknya mesin yang mampu mengerjakan proses ke-x*/
    private Job[] jobDetail;    /*detail pekerjaan dengan nama/nomor proses ke-x*/
    private int   upperBound;   /*batas atas makespan yang mungkin dihasilkan dari permasalahan hybrid flow shop*/
    private int   lowerBound;   /*batas bawah makespan yang mungkin dihasilkan dari permasalahan hybrid flow shop*/

    /**
     * Constructor
     *
     * @param job        : banyaknya pekerjaan
     * @param proses     : banyaknya proses
     * @param jobDetail  : detail pekerjaan
     * @param mesin      : detail banyaknya mesin untuk masing-masing proses
     * @param upperBound : batas atas pengujian
     * @param lowerBound : batas bawah pengujian
     */
    public Problem(int job, int proses, Job[] jobDetail, int[] mesin, int upperBound, int lowerBound){
        this.jumlahJob          = job;
        this.jumlahProses       = proses;
        this.jobDetail          = jobDetail;
        this.jumlahMesin        = mesin;
        this.upperBound         = upperBound;
        this.lowerBound         = lowerBound;
    }

    /**
     * menginformasikan banyaknya pekerjaan.
     *
     * @return int banyaknya pekerjaan
     */
    public int getJumlahJob(){
        return this.jumlahJob;
    }

    /**
     * menginformasikan banyaknya proses.
     *
     * @return int banyaknya proses
     */
    public int getJumlahProses(){
        return this.jumlahProses;
    }

    /**
     * menginformasikan detail pekerjaan-pekerjaan.
     *
     * @return Job[] detail masing-masing pekerjaan
     */
    public Job[] getJobDetail(){
        return this.jobDetail;
    }

    /**
     * menginformasikan detail pekerjaan dengan nomor/nama pekerjaan number.
     *
     * @param number : nama/nomor pekerjaan
     * @return Job detail pekerjaan dengan nomor pekerjaan number
     */
    public Job getJobDetail(int number){
```

```
69          return this.jobDetail[number];
70      }
71
72      /**
73       * menginformasikan banyaknya mesin untuk setiap proses.
74       *
75       * @return int[x] banyaknya mesin yang mampu mengerjakan proses x
76       */
77      public int[] getJumlahMesin(){
78          return this.jumlahMesin;
79      }
80
81      /**
82       * menginformasikan banyaknya mesin untuk pengerjaan proses index.
83       *
84       * @param index : nomor proses
85       * @return int banyaknya mesin untuk mengerjakan proses ke-index
86       */
87      public int getJumlahMesin(int index){
88          return this.jumlahMesin[index];
89      }
90
91      /**
92       * menginformasikan batas atas untuk pengujian dari permasalahan hybrid flow shop.
93       *
94       * @return int batas atas untuk pengujian permasalahan
95       */
96      public int getUpperBound(){
97          return this.upperBound;
98      }
99
100     /**
101      * menginformasikan batas bawah untuk pengujian dari permasalahan hybrid flow shop.
102      *
103      * @return int batas bawah untuk pengujian permasalahan
104      */
105     public int getLowerBound(){
106         return this.lowerBound;
107     }
108 }
```

Listing E.2: Job.java

```
1  /**
2   * Kelas ini merepresentasikan sebuah job dalam penjadwalan flowshop.
3   * Kelas ini menyimpan detail waktu masing-masing proses dari sebuah job.
4   *
5   * @author ALEX
6   */
7  public class Job {
8      private int   jobNumber;     /*Nama/nomor pekerjaan dari sebuah job*/
9      private int[] processTime;   /*detail waktu proses-proses yang ada pada job*/
10     private int   jumlahProses; /*banyaknya proses pada sebuah job*/
11
12     /**
13      * Constructor
14      *
15      * @param jobNumber
16      *          input nama/nomor pekerjaan
17      * @param processTime
18      *          input detail waktu pengerjaan dari masing-masing proses pada job
19      * @param jumlahProses
20      *          input banyaknya proses yang ada pada sebuah job
21      */
22     public Job(int jobNumber, int[] processTime){
23         this.jobNumber    = jobNumber;
24         this.processTime  = processTime;
25         this.jumlahProses = processTime.length;
26     }
27
28     /**
29      * menjabarkan datail waktu proses-proses ke dalam sebuah String.
30      *
31      * @return String yang berisi detail waktu proses-proses.
32      */
33     public String toString(){
34         String res = "";
35         int count = 0;
36         while(count<jumlahProses){
37             res+=this.processTime[count]+" ";
38             count++;
39         }
40         return res;
41     }
42
43     /**
44      * menginformasikan nama/nomor pekerjaan.
45      *
46      * @return int nomor pekerjaan
47      */
48     public int getJobNumber(){
49         return this.jobNumber;
50     }
51
52     /**
53      * menginformasikan detail waktu proses-proses yang ada.
54      *
55      * @return int[] detail waktu masing-masing proses
```

```
56          */
57      public int[] getProcessTime(){
58          return this.processTime;
59      }
60
61      /**
62       * menginformasikan detail waktu proses-proses yang ada.
63       *
64       * @param number : indeks proses
65       * @return int detail waktu proses ke-number
66       */
67      public int getProcessTime(int number){
68          return this.processTime[number];
69      }
70
71      /**
72       * menginformasikan banyaknya proses yang ada pada pekerjaan.
73       *
74       * @return int banyaknya proses
75       */
76      public int getJumlahProses(){
77          return this.jumlahProses;
78      }
79  }
```

Listing E.3: Mesin.java

```
1   /**
2    * Kelas ini merepresentasikan sebuah mesin pada penjadwalan hybrid flow shop.
3    *
4    * @author ALEX
5    */
6   public class Mesin {
7       private Job     job;           /*Job yang saat ini deikerjakan oleh mesin*/
8       private int     processNumber; /*nomor proses yang dikerjakan oleh mesin*/
9       private int     processTime;   /*waktu yang telah dilalui untuk mengerjakan job saat ini*/
10      private boolean finished;      /*status mesin selesai mengerjakan sebuah job*/
11      private boolean waiting;       /*status mesin menunggu masuknya job baru*/
12
13      /**
14       * Constructor
15       *
16       * @param processNumber : nomor proses yang dikerjakan oleh mesin
17       */
18      public Mesin(int processNumber){
19          this.job           = null;
20          this.processNumber = processNumber;
21          this.processTime   = 0;
22          this.finished      = true;
23          this.waiting       = true;
24      }
25
26      /**
27       * menambah waktu proses untuk pengerjaan suatu job pada mesin, jika mesin sedang mengerjakan sebuah
             job.
28       *
29       */
30      public void processing(){
31          if(!this.finished){
32              this.processTime++;
33              //System.out.println(this.processTime==this.job.getProcessTime(this.processNumber));
34              if(this.processTime>=this.job.getProcessTime(this.processNumber)){
35                  this.finished    = true;
36                  this.processTime = 0;
37              }
38          }
39      }
40
41      /**
42       * Mengoper job ke mesin selanjutnya.
43       * Mengubah status mesin menjadi menunggu pekerjaan baru.
44       *
45       * @return Job job yang ingin dioper
46       */
47      public Job passJob(){
48          Job toBePassed = this.job;
49          this.waiting   = true;
50          this.job       = null;
51          return toBePassed;
52      }
53
54      /**'
55       * menerima job baru untuk dikerjakan.
56       * mengubah status mesin menjadi tidak menunggu pekerjaan baru.
57       *
58       * @param job : job baru yang akan dikerjakan oleh mesin
59       */
60      public void receiveJob(Job job){
61          this.job       = job;
62          this.waiting  = false;
63          this.finished = false;
64      }
65      /**
66       * mencari tahu kondisi mesin, apakah mesin sedang menunggu pekerjaan baru.
67       *
68       * @return boolean : kondisi menunggu mesin
69       */
70      public boolean isWaiting(){
```

```
71          return this.waiting;
72      }
73
74      /**
75       * mencari tahu kondisi mesin, apakah mesin sudah menyelesaikan pekerjaannya.
76       *
77       * @return boolean : kondisi mesin telah menyelesaikan pekerjaannya
78       */
79      public boolean isFinished(){
80          return this.finished;
81      }
82
83      /**
84       * menginformasikan detil pekerjaan yang sedang ada/dikerjakan pada mesin.
85       * jika mesin sedang kosong akan mengembalikan nilai null.
86       *
87       * @return job : pekerjaan yang sedang ada/dikerjakan pada mesin
88       */
89      public Job getJob(){
90          return this.job;
91      }
92  }
```

Listing E.4: HybridFlowShop.java

```
1   /**
2    * Kelas ini merepresentasikan sistem untuk proses hybrid flow shop
3    *
4    * @author ALEX
5    */
6   public class HybridFlowShop {
7       private Mesin[][]           mesin;   /*mesin-mesin yang ada pada hybrid flow shop*/
8
9       /**
10       * Constructor
11       *
12       * @param problem : permasalahan yang ingin diselesaikan
13       */
14      public HybridFlowShop(Problem problem){
15          int mesinData[] = problem.getJumlahMesin();
16          Mesin mesin[][] = new Mesin[problem.getJumlahProses()][];
17          int count1 = 0;
18          while(count1<mesin.length){
19              mesin[count1] = new Mesin[mesinData[count1]];
20              int count2 = 0;
21              while(count2<mesinData[count1]){
22                  mesin[count1][count2] = new Mesin(count1);
23                  count2++;
24              }
25              count1++;
26          }
27          this.mesin = mesin;
28      }
29
30      /**
31       * menghitung makespan yang dihasilkan dari urutan pengambilan pekerjaan tertentu.
32       *
33       * @param job : urutan pengambilan pekerjaan yang dipilih
34       * @return int makespan yang dihasilkan
35       */
36      public int countMakespan(Job[] job){
37          int makespan = 0;
38          int jobcount = 0;
39          while(true){
40              //cek job yang sudah selesai
41              int count = 0;
42              while(count<this.mesin[this.mesin.length-1].length){
43                  if(this.mesin[this.mesin.length-1][count].isFinished()){
44                      this.mesin[this.mesin.length-1][count].passJob();
45                  }
46                  count++;
47              }
48              //cek job yang bisa dioper
49              //untuk setiap proses selain proses terakhir
50              count = this.mesin.length-2;
51              while(count>=0){
52                  //cek setiap mesin yang bisa ngoper job
53                  int count2     = 0;
54                  int mesinCount = 0;
55                  Mesin[] mesins = new Mesin[this.mesin[count].length];
56                  while(count2<this.mesin[count].length){
57                      if(this.mesin[count][count2].isFinished()&&!(this.mesin[count][count2].isWaiting())){
58                          mesins[mesinCount] = this.mesin[count][count2];
59                          mesinCount++;
60                      }
61                      count2++;
62                  }
63
64
65                  if(mesinCount!=0){
66                      //cek mesin yang bisa dioper
67                      if(mesinCount==1){
68                          int count3 = 0;
69                          while(count3<this.mesin[count+1].length){
70                              if(this.mesin[count+1][count3].isWaiting()){
71                                  Job temp = mesins[0].passJob();
72                                  this.mesin[count+1][count3].receiveJob(temp);
73                                  break;
```

```
74                              }
75                              count3++;
76                          }
77                      }
78                      else{
79                          int count3 = 0;
80                          while(count3<this.mesin[count+1].length){
81                              if(this.mesin[count+1][count3].isWaiting()){
82                                  int count4 = 0;
83                                  boolean notReceive = true;
84                                  //liat urutan awal pengambilan job
85                                  while(count4<job.length&&notReceive){
86                                      //untuk setiap mesin yang bisa ngoper
87                                      //yang nomor jobnya pertama kali disebut pada urutan
88                                      int count5 = 0;
89                                      while(count5<mesinCount&&notReceive){
90                                          //jaga-jaga kalau terdapat job yang sudah dioper sebelumnya
91                                          if(!mesins[count5].isWaiting()){
92                                              if(mesins[count5].getJob().getJobNumber()==job[count4].
                                                    getJobNumber()){
93                                                  Job temp = mesins[count5].passJob();
94                                                  this.mesin[count+1][count3].receiveJob(temp);
95                                                  notReceive = false;
96                                              }
97                                          }
98                                          count5++;
99                                      }
100                                     count4++;
101                                 }
102                             }
103                             count3++;
104                         }
105                     }
106                 }
107                 count--;
108             }
109             //cek job yang bisa dimasukkin
110             if(jobcount<job.length){
111                 //untuk setiap mesin proses pertama
112                 int count2 = 0;
113                 while((count2<this.mesin[0].length)&&(jobcount<job.length)){
114                     if(this.mesin[0][count2].isWaiting()){
115                         this.mesin[0][count2].receiveJob(job[jobcount]);
116                         jobcount++;
117                     }
118                     count2++;
119                 }
120             }
121             //cek selesai
122             if(this.finishCheck()){
123                 break;
124             }
125             else{
126                 this.proccessAll();
127                 makespan++;
128             }
129         }
130         return makespan;
131     }
132
133     /**
134      * menjalankan proses pada setiap mesin.
135      *
136      */
137     public void proccessAll(){
138         int count1 = 0;
139         while(count1<this.mesin.length){
140             int count2 = 0;
141             while(count2<this.mesin[count1].length){
142                 if(!this.mesin[count1][count2].isFinished()){
143                     this.mesin[count1][count2].processing();
144                 }
145                 count2++;
146             }
147             count1++;
148         }
149     }
150
151     /**
152      * memeriksa kondisi mesin-mesin pada hybrid flow shop, apakah sudah selesai menjalankan suatu solusi
            atau belum.
153      *
154      * @return boolean kondisi hybrid flow shop sudah selesai atau belum
155      */
156     public boolean finishCheck(){
157         int count1 = 0;
158         while(count1<this.mesin.length){
159             int count2 = 0;
160             while(count2<this.mesin[count1].length){
161                 if((!this.mesin[count1][count2].isFinished())||(!this.mesin[count1][count2].isWaiting())){
162                     return false;
163                 }
164                 count2++;
165             }
166             count1++;
167         }
168         return true;
169     }
170 }
```

Listing E.5: Ant.java

```java
/**
 * Kelas ini merepresentasikan semut pada sistem alrgoritma koloni semut.
 * Kelas ini menyimpan solusi lokal dalam algoritma koloni semut untuk permasalahan hybrid flow shop.
 * Solusi tersebut berupa urutan pengambilan pekerjaan yang diambil, dan makespan yang dihasilkan dari urutan pengambilan pekerjaan tersebut.
 *
 * @author ALEX
 */
public class Ant {
    private Job[] jobSequence; /*urutan pengambilan pekerjaan yang dipilih oleh semut*/
    private int makespan;      /*makespan yang dihasilkan berdasarkan urutan pengambilan pekerjaan yang dipilih*/

    /**
     * Constructor
     *
     */
    public Ant(){
        this.jobSequence = null;
        this.makespan    = -1;
    }

    /**
     * Constructor
     *
     * @param jobSequence : urutan pengambilan pekerjaan yang diambil semut
     * @param makespan    : makespan yang dihasilkan berdasarkan urutan pengambilan pekerjaan
     */
    public Ant(Job[] jobSequence, int makespan){
        this.jobSequence = jobSequence;
        this.makespan    = makespan;
    }

    /**
     *
     * @return String informasi mengenai urutan pengambilan pekerjaan dan makespan yang dihasilkan dari urutan pengambilan tersebut.
     */
    public String toString(){
        String res = "Makespan : "+this.makespan+"\n";
        res       += this.jobPathToString(jobSequence);
        return res;
    }


    /**
     * mendeskripsikan urutan nomor job yang ada pada sebuah kumpulan job.
     *
     * @param jobs : kumpulan job
     * @return String detail mengenai urutan nomor job pada kumpulan job
     */
    public String jobPathToString(Job[] jobs){
        String res = "";
        int count = 0;
        while(count<jobs.length){
            int jobNumber = jobs[count].getJobNumber()+1;
            res += jobNumber+" ";
            count++;
        }
        return res;
    }

    /**
     * menginformasikan makespan yang dihasilkan dari urutan pengambilan pekerjaan yang dimiliki semut.
     *
     * @return int makespan yang dihasilkan
     */
    public int getMakespan(){
        return this.makespan;
    }

    /**
     * menginformasikan urutan pengambilan pekerjaan dan detail-detail pekerjaan yang dipilih oleh semut.
     *
     * @return Job[] urutan pengambilan pekerjaan
     */
    public Job[] getJobSequence(){
        return this.jobSequence;
    }

    /**
     * menginformasikan detail pekerjaan yang diambil oleh semut pada urutan tertentu.
     *
     * @return Job detail job yang diambil ke-index oleh semut
     */
    public Job getJobSequence(int index){
        return this.jobSequence[index];
    }

    /**
     * mengubah urutan pengambilan pekerjaan/solusi yang dipilih oleh semut.
     *
     * @param jobSequence : urutan pengambilan pekerjaan yang baru
     */
    public void setJobSequence(Job[] jobSequence){
        this.jobSequence = jobSequence;
    }

```

```
96      /**
97       * mengubah makespan yang dihasilkan dari urutan pengambilan pekerjaan/solusi yang dipilih oleh semut.
98       *
99       * @param makespan : makespan yang baru
100      */
101     public void setMakespan(int makespan){
102         this.makespan = makespan;
103     }
104 }
```

Listing E.6: PheromoneDatabase.java

```
1  /**
2   * Kelas ini merepresentasikan database feromon yang berpengaruh pada proses pemilihan jalur/solusi pada
        algoritma koloni semut.
3   * Kelas ini menyimpan feromon untuk penyelesaian kasus hybrid flow shop.
4   * Kelas ini menyimpan feromon dalam array 2 dimensi.
5   *
6   * @author ALEX
7   */
8  public class PheromoneDatabase {
9      private int[][] dataFeromon;                    /*data feromon untuk algoritma kolon semut, indeks pertama
           menunjukkan job yang sebelumnya diambil, indeks kedua menunjukkan job selanjtnya untuk diambil*/
10     private static final int startValue = 300; /*nilai feromon awal untuk masing-masing indeks feromon*/
11
12     /**
13      * Constructor
14      *
15      * @param jumlahPekerjaan : jumlah pilihan pekerjaan yang ada
16      */
17     public PheromoneDatabase(int jumlahPekerjaan){
18         this.dataFeromon = new int[jumlahPekerjaan][jumlahPekerjaan];
19         int count1 = 0;
20         while(count1<jumlahPekerjaan){
21             int count2 = 0;
22             while(count2<jumlahPekerjaan){
23                 this.dataFeromon[count1][count2] = this.startValue;
24                 count2++;
25             }
26             count1++;
27         }
28         int count3 = 0;
29         while(count3<jumlahPekerjaan){
30             this.dataFeromon[count3][count3] = 0;
31             count3++;
32         }
33     }
34
35     /**
36      * menginformasikan nilai feromon untuk pengambilan pekerjaan ke-target setelah pengambilan pekerjaan
            ke-now.
37      * indeks feromon yang diambil adalah [now][target].
38      *
39      * @param now    : indeks pekerjaan yang baru saja diambil.
40      * @param target : indeks pekerjaan yang mungkin diambil untuk pengambilan selanjutnya.
41      * @return int nilai feromon pengambilan pekerjaan ke-target setelah pengambilan pekerjaan ke-now
42      */
43     public int getPheromone(int now, int target){
44         return this.dataFeromon[now][target];
45     }
46
47     /**
48      * menambahkan nilai feromon untuk pengambilan pekerjaan ke-target setelah pengambilan pekerjaan ke-
            now sejumlah value.
49      * indeks feromon yang dipengaruhi adalah [now][target].
50      *
51      * @param now    : indeks pekerjaan yang baru saja diambil.
52      * @param target : indeks pekerjaan yang diambil selanjutnya.
53      * @param value  : nilai feromon yang ditambahkan.
54      */
55     public void addPheromone(int now, int target, int value){
56         if(value>0){
57             this.dataFeromon[now][target] += value;
58         }
59     }
60
61     /**
62      * melakukan penguapan/pengurangan nilai pada feromon-feromon yang telah disebar.
63      * nilai feromon yang dikurangi adalah sebanyak 30% dari nilai feromon masing-masing.
64      * penguapan feromon tidak akan menghabiskan nilai feromon, agar suatu jalur masih tetap memiliki
            kemungkinan untuk dipilih.
65      *
66      */
67     public void evaporate(){
68         int count1 = 0;
69         while(count1<this.dataFeromon.length){
70             int count2 = 0;
71             while(count2<this.dataFeromon[count1].length){
72                 this.dataFeromon[count1][count2] = (this.dataFeromon[count1][count2]/10)*7;
73                 if((this.dataFeromon[count1][count2]<=0)&&(count1!=count2)){
74                     this.dataFeromon[count1][count2] = 1;
75                 }
76                 count2++;
77             }
78             count1++;
79         }
80     }
81
```

```
82      /**
83       * menampilkan detail mengenai nilai-nilai feromon yang disimpan.
84       *
85       * @return String detail nilai-nilai feromon yang disimpan
86       */
87      public String toString(){
88          String res = "";
89          int count1 = 0;
90          while(count1<this.dataFeromon.length){
91              int count2 = 0;
92              while(count2<this.dataFeromon[count1].length){
93                  res += this.dataFeromon[count1][count2]+"\t";
94                  count2++;
95              }
96              res+= "\n";
97              count1++;
98          }
99          return res;
100     }
101 }
```

Listing E.7: AntColonySub.java

```
1  import java.util.Random;
2  /**
3   * Kelas ini merepresentasikan sebuah kelompok semut pada algoritma ant colony.
4   *
5   * @author ALEX
6   */
7  public class AntColonySub implements Runnable{
8      private PheromoneDatabase pheromone;       /*penyimpan data feromon yang berpengaruh pada pemilihan
           solusi yang optimal*/
9      private Problem           problem;         /*permasalahan hybrid flow shop yang ingin diselesaikan*/
10     private HybridFlowShop    hfs;             /*hybrid flow shop untuk menghitung makespan yang dihasilkan
           dari solusi-solusi yang dipilih*/
11     private Ant[]             antParty;        /*Kelompok semut yang telah disebar pada proses pelatihan
           terakhir, dan memiliki solusi lokalny masing-masing*/
12     private Ant               antOpt;          /*Semut yang memiliki solusi yang paling optimal*/
13     private int               antNumber;       /*Jumlah semut yang akan disebar setiap kali proses
           pelatihan dijalankan*/
14
15     /**
16      * Constructor
17      *
18      * @param pheromone : database feromon yang dimiliki oleh algoritma ant colony.
19      * @param problem   : permasalahan hybrid flow shop yang ingin dicari hasil optimalnya.
20      * @param antNumber : jumlah semut yang akan disebar pada setiap proses pelatihan.
21      * @param antOpt    : semut yang menyimpan suatu solusi kasus hybrid flow shop yang digunakan sebagai
            nilai perbandingan awal
22      */
23     public AntColonySub(PheromoneDatabase pheromone, Problem problem, int antNumber, Ant antOpt){
24         this.pheromone = pheromone;
25         this.problem   = problem;
26         this.hfs       = new HybridFlowShop(problem);
27         this.antNumber = antNumber;
28         this.antOpt    = antOpt;
29     }
30
31
32     @Override
33     public void run() {
34         this.spreadAnt(this.antNumber);
35         Thread.currentThread().interrupt();
36     }
37
38      /**
39      * melakukan pelatihan untuk mencari solusi optimal dari permasalahan pada atribut problem.
40      * pelatihan dilakukan dengan menyebarkan semut dengan solusi lokalnya masing-masing.
41      * semut yang disebarkan sejumlah parameter trainingCount.
42      *
43      * @param  trainingCount : jumlah semut yang akan disebar
44      * @return Ant semut yang menyimpan solusi(urutan pengerjaan dan makespan) yang dianggap paling
            optimal saat ini.
45      */
46     public Ant spreadAnt(int trainingCount){
47         Ant antOpt     = this.antOpt;
48         Ant[] ants     = new Ant[trainingCount];
49         int count      = 0;
50         while(count<trainingCount){
51
52             Job[] jobPath = this.constructPath();
53             int makespan  = this.hfs.countMakespan(jobPath);
54             ants[count]   = new Ant(jobPath, makespan);
55
56             if(makespan<antOpt.getMakespan()){
57                 antOpt = ants[count];
58                 this.antOpt = ants[count];
59             }
60             count++;
61         }
62         this.antParty = ants;
63         return antOpt;
64     }
65
66     /**
67      * membangun contoh solusi secara acak berdasarkan nilai feromon pada atribut pheromone.
68      * pekerjaan yang diambil pertama kali perlu disebutkan, karena database feromon tidak menyimpan nilai
           untuk pengambilan job pertama kali.
```

```
69          * (belum ada job yang sudah diambil sebelumnya)
70          *
71          * @return Job[] urutan pengambilan pekerjaan/solusi.
72          */
73         public Job[] constructPath(){
74             //duplikat job dari problem
75             Job[] jobDupe = new Job[this.problem.getJumlahJob()];
76             int count = 0;
77             while(count<jobDupe.length){
78                 jobDupe[count] = this.problem.getJobDetail(count);
79                 count++;
80             }
81 
82             //job pertama dimasukkan
83             Job[] jobSequence = new Job[this.problem.getJumlahJob()];
84             int jobCount      = 0;
85             int lastJob       = -1;
86             //random pengambilan job
87             while(jobCount<jobSequence.length){
88                 Job jobTarget          = this.pickRandomJob(jobDupe, lastJob);
89                 jobSequence[jobCount] = jobTarget;
90                 lastJob               = jobTarget.getJobNumber();
91                 jobDupe               = this.removeJobFrom(lastJob, jobDupe);
92                 jobCount++;
93             }
94             return jobSequence;
95         }
96 
97         /**
98          * mengambil 1 job secara acak dari sebuah kumpulan job.
99          * job yang diambil dipengaruhi oleh job terakhir yang dipilih dan nilai feromon pada atribut
                pheromone.
100         * lastJobNumber akan bernilai -1, jika belum ada job yang pernah diambil(pengambilan pertama.
101         *
102         * @param jobs          : kumpulan job yang ada
103         * @param lastJobNumber : nama/nomor job terakhir yang diambil
104         * @return Job salah satu job dari kumpulan job
105         */
106        public Job pickRandomJob(Job[] jobs, int lastJobNumber){
107            //cari nilai feromon maksimal
108            int maxPheromone = 0;
109            int count1 = 0;
110            while(count1<jobs.length){
111                if(lastJobNumber==-1){
112                    int count3 = 0;
113                    while(count3<jobs.length){
114                        maxPheromone += this.pheromone.getPheromone(count3, jobs[count1].getJobNumber());
115                        count3++;
116                    }
117                }
118                else{
119                    maxPheromone+= this.pheromone.getPheromone(lastJobNumber, jobs[count1].getJobNumber());
120                }
121                count1++;
122            }
123            //target 1 angka antara 1 sampai nilai maksimal feromon
124            Random rd = new Random();
125            int pheromoneTarget = rd.nextInt(maxPheromone)+1;
126            //tambahkan nilai-nilai feromon berdasarkan job yang ada sampai melebihi/sama dengan nilai target.
127            //indeks terakhir yang di cek merupakan job yang dipilih
128            int count2          = 0;
129            int pheromoneCount = 0;
130            while(count2<jobs.length){
131                if(lastJobNumber==-1){
132                    int count3 = 0;
133                    while(count3<jobs.length){
134                        pheromoneCount += this.pheromone.getPheromone(count3, jobs[count2].getJobNumber());
135                        count3++;
136                    }
137                }
138                else{
139                    pheromoneCount += this.pheromone.getPheromone(lastJobNumber, jobs[count2].getJobNumber());
140                }
141 
142                if(pheromoneTarget<=pheromoneCount){
143                    break;
144                }
145                else{
146                    count2++;
147                }
148            }
149            return jobs[count2];
150        }
151 
152        /**
153         * menyisihkan job dengan nomor/nama tertentu dari sekumpulan job.
154         *
155         * @param jobNumber : nomor/nama job yang ingin disisihkan
156         * @param jobs      : kumpulan job
157         * @return Job[] kumpulan job baru tanpa job ke-jobNumber
158         */
159        public Job[] removeJobFrom(int jobNumber, Job[] jobs){
160            Job[] jobsResult = new Job[jobs.length-1];
161            int jobCount = 0;
162            int count    = 0;
163            while(jobCount<jobsResult.length){
164                if(jobs[count].getJobNumber()!=jobNumber){
165                    jobsResult[jobCount] = jobs[count];
166                    jobCount++;
```

```
167             }
168             count++;
169         }
170         return jobsResult;
171     }
172 
173     /**
174      * mengembalikan informasi mengenai solusi-solusi lokal yang dipilih oleh semut-semut pada proses
            pelatihan terakhir
175      *
176      * @return Ant[] solusi-solusi lokal yang disimpan dalam bentuk sebuah objek kelas Ant
177      */
178     public Ant[] getAntParty(){
179         return this.antParty;
180     }
181 
182     /**
183      * mengembalikan informasi mengenai solusi yang dianggap paling optimal saat ini oleh kelompok semut
            ini.
184      *
185      * @return Ant Solusi lokal yang dianggap paling optimal.
186      */
187     public Ant getAntOpt(){
188         return this.antOpt;
189     }
190 
191     /**
192      * menginformasikan dan mengganti nilai dari solusi yang saat ini dianggap paling optimal.
193      *
194      * @param ant : solusi optimal yang baru
195      */
196     public void setAntOpt(Ant ant){
197         this.antOpt = ant;
198     }
199 }
```

Listing E.8: AntColonyAlgorithm.java

```
1  /**
2   * Kelas ini merpresentasikan algoritma koloni semut.
3   * Kelas ini berguna untuk mencari pilihan urutan pengambilan/solusi yang paling optimal untuk suatu
        permasalahan hybrid flow shop.
4   *
5   * @author ALEX
6   */
7  public class AntColonyAlgorithm {
8      private PheromoneDatabase pheromone;       /*penyimpan data feromon yang berpengaruh pada pemilihan
           solusi yang optimal*/
9      private Problem           problem;         /*permasalahan hybrid flow shop yang ingin diselesaikan*/
10     private Ant               optimalSolution;/*semut dengan solusi nilai paling optimal saat ini*/
11     private int               trainingCount;   /*jumlah pelatihan yang telah dilakukan pada sistem semut
           ini*/
12     private int               pheromoneGap;    /*nilai yang digunakan sebagai konstanta untuk penambahan
           feromon*/
13     private int               antNumber;       /*banyaknya semut pada masing-masing kelompok semut*/
14     private AntColonySub[]    antColonySub;    /*Kelompok-kelompok semut yang digunakan untuk proses
           pelatihan secara serentak*/
15     private Thread[]          antThread;       /*Thread penunjuk informasi mengenai selesainya pelatihan
           suatu kelompok semut*/
16 
17     /**
18      * Constructor
19      *
20      * @param problem : permasalahan hybrid flow shop yang ingin diselesaikan
21      */
22     public AntColonyAlgorithm(Problem problem){
23         this.pheromone        = new PheromoneDatabase(problem.getJumlahJob());
24         this.problem          = problem;
25         HybridFlowShop hfs    = new HybridFlowShop(problem);
26         int makespan          = hfs.countMakespan(problem.getJobDetail());
27         this.optimalSolution = new Ant(problem.getJobDetail(),makespan);
28         this.trainingCount    = 0;
29         this.pheromoneGap     = makespan;
30         this.antNumber        = problem.getJumlahJob();//*problem.getJumlahJob();
31         this.antColonySub     = new AntColonySub[this.problem.getJumlahJob()];
32         this.antThread        = new Thread[this.problem.getJumlahJob()];
33         int count = 0;
34         while(count<this.problem.getJumlahJob()){
35             this.antColonySub[count] = new AntColonySub(pheromone, problem, antNumber, this.
                   optimalSolution);
36             this.antThread[count]    = new Thread(this.antColonySub[count]);
37             count++;
38         }
39     }
40 
41     /**
42      * Melakukan 1 kali proses pelatihan dari algoritma ant colony dengan cara menjalankan method
            spreadAnt().
43      *
44      * @return String informasi hasil optimisasi yang dilakukan
45      */
46     public String trainOnce() throws InterruptedException{
47         Ant opt = this.spreadAnt();
48         String res = "";
49         System.out.println("Training "+this.trainingCount);
50         System.out.println(opt.toString());
51         res += "Training "+this.trainingCount+"\n";
52         res += opt.toString()+"\n";
```

```
53          return res;
54      }
55
56      /**
57       * Menjalankan proses pelatihan dari algoritma ant colony hingga kondisi suatu berhenti tercapai.
58       * Kondisi berhenti yang digunakan adalah kondisi dihasilkannya nilai makespan yang sama pada 10
             proses pelatihan terakhir.
59       *
60       * @return String informasi hasil optimisasi pada tiap proses pelatihan.
61       */
62      public String trainFull(){
63          String res = "";
64          int makespan = 0;
65          int count = 0;
66          while(count<10){
67              try{
68                  Ant opt = this.spreadAnt();
69                  System.out.println("Training "+this.trainingCount);
70                  System.out.println(opt.toString());
71                  res += "Training "+this.trainingCount+"\n";
72                  res += opt.toString()+"\n";
73                  if(opt.getMakespan()==makespan){
74                      count++;
75                  }
76                  else{
77                      makespan = opt.getMakespan();
78                      count = 0;
79                  }
80              }
81              catch(OutOfMemoryError error){
82                  break;
83              }
84              catch (InterruptedException ex) {
85
86              }
87          }
88          return res;
89      }
90
91      /**
92       * melakukan pelatihan untuk mencari solusi optimal dari permasalahan pada atribut problem.
93       * pelatihan dilakukan dengan menyebarkan semut dengan solusi lokalnya masing-masing.
94       * semut yang disebarkan sejumlah parameter trainingCount.
95       *
96       * \@return Ant semut yang menyimpan solusi(urutan pengerjaan dan makespan) yang dianggap paling
             optimal saat ini.
97       */
98      public Ant spreadAnt() throws InterruptedException{
99          //memulai proses pelatihan oleh masing-masing kelompok semut
100         int countStart = 0;
101         while(countStart<this.antThread.length){
102             this.antThread[countStart] = new Thread(this.antColonySub[countStart]);
103             this.antThread[countStart].start();
104             countStart++;
105         }
106         //menunggu hingga semua semut menyelesaikan proses latihannya
107         int countWait = 0;
108         while(countWait<this.antThread.length){
109             this.antThread[countWait].join();
110             countWait++;
111         }
112         //mengambil hasil solusi-solusi dari masing-masing kelompok semut dan melakukan proses update
                feromon
113         int countUpdate = 0;
114         while(countUpdate<this.antThread.length){
115             this.updatePheromone(this.antColonySub[countUpdate].getAntParty());
116             countUpdate++;
117         }
118         //mencari tahu apakah proses pelatihan mendapatkan solusi optimal yang baru/yang lebih baik
119         int countGetMax = 0;
120         while(countGetMax<this.antColonySub.length){
121             if(this.optimalSolution.getMakespan()>this.antColonySub[countGetMax].getAntOpt().getMakespan()
                    ){
122                 this.optimalSolution = this.antColonySub[countGetMax].getAntOpt();
123             }
124             countGetMax++;
125         }
126         //menginformasikan solusi optimal yang baru pada masing-masing kelompok semut
127         int countUpdateOpt = 0;
128         while(countUpdateOpt<this.antColonySub.length){
129             this.antColonySub[countUpdateOpt].setAntOpt(this.optimalSolution);
130             countUpdateOpt++;
131         }
132         //evaporasi feromon dan menambah jumlah hitungan proses pelatihan
133         this.pheromone.evaporate();
134         this.trainingCount++;
135         return this.optimalSolution;
136     }
137
138     /**
139      * melakukan update pada atribut pheromone, berdasarkan solusi dan makespan yang dihasilkan oleh suatu
            semut.
140      *
141      * @param ant : semut yang telah menyimpan nilai makespan dari salah satu solusi
142      */
143     public void updatePheromone(Ant ant){
144         Job[] jobSequence = ant.getJobSequence();
145         int makespan      = ant.getMakespan();
146         int count = 1;
```

```
147         while(count<jobSequence.length){
148             this.pheromone.addPheromone(jobSequence[count-1].getJobNumber(), jobSequence[count].
                    getJobNumber(), this.pheromoneGap-makespan);
149             count++;
150         }
151     }
152 
153     /**
154      * melakukan update pada atribut pheromone, berdasarkan solusi dan makespan yang dihasilkan oleh suatu
|         kumpulan semut.
155      *
156      * @param ants : array kumpulan semut
157      */
158     public void updatePheromone(Ant[] ants){
159         int count = 0;
160         while(count<ants.length){
161             this.updatePheromone(ants[count]);
162             count++;
163         }
164     }
165 
166     /**
167      * mendapatkan informasi mengenai berapa kali pelatihan telah dilakukan.
168      *
169      * @return int banyaknya pelatihan yang telah dilakukan
170      */
171     public int getTrainingCount(){
172         return this.trainingCount;
173     }
174 
175     /**
176      * mendapatkan informasi mengenai urutan pengerjaan dan makespan yang saat ini dianggap paling optimal
|         .
177      * informasi disimpan dalam bentuk kelas Ant.
178      *
179      * @return Ant yang berisi informasi mengenai solusi yang dianggap optimal saat ini
180      */
181     public Ant getOptimalSolution(){
182         return this.optimalSolution;
183     }
184 
185     /**
186      * mengembalikan informasi mengenai banyaknya semut yang disebar pada setiap proses pelatihan
187      *
188      * @return int : banyaknya semut yang disebar
189      */
190     public int getAntNumber(){
191         return this.antNumber;
192     }
193 
194     /**
195      * mengembalikan informasi mengenai nilai feromon yang disimpan dalam bentuk string matriks.
196      *
197      * @return String : informasi nilai feromon
198      */
199     public String getPheromoneData(){
200         return this.pheromone.toString();
201     }
202 }
```

Listing E.9: TesterGUI.java

```
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.util.Scanner;
4  import java.util.logging.Level;
5  import java.util.logging.Logger;
6  import javax.swing.JFileChooser;
7  import javax.swing.JOptionPane;
8  import javax.swing.filechooser.FileNameExtensionFilter;
9 
10 /*
11  * To change this template, choose Tools | Templates
12  * and open the template in the editor.
13  */
14 
15 /**
16  *
17  * @author ALEX
18  */
19 public class TesterGUI extends javax.swing.JFrame {
20     private AntColonyAlgorithm aco;
21     private Problem problem;
22 
23     /**
24      * Creates new form TesterGUI
25      */
26     public TesterGUI() {
27         this.setTitle("Ant␣Colony␣Optimization␣(Multi-Thread)");
28         initComponents();
29     }
30 
31     /**
32      * This method is called from within the constructor to initialize the form.
33      * WARNING: Do NOT modify this code. The content of this method is always
34      * regenerated by the Form Editor.
35      */
36     @SuppressWarnings("unchecked")
```

```
37      // <editor-fold defaultstate="collapsed" desc="Generated Code">
38      private void initComponents() {
39
40          buttInput = new javax.swing.JButton();
41          labJob = new javax.swing.JLabel();
42          labPro = new javax.swing.JLabel();
43          jScrollPane1 = new javax.swing.JScrollPane();
44          taResult = new javax.swing.JTextArea();
45          buttTrain = new javax.swing.JButton();
46          tfDir = new javax.swing.JTextField();
47          buttTrainFull = new javax.swing.JButton();
48          buttReset = new javax.swing.JButton();
49          labInfo = new javax.swing.JLabel();
50          labBound = new javax.swing.JLabel();
51
52          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
53
54          buttInput.setText("Browse");
55          buttInput.addActionListener(new java.awt.event.ActionListener() {
56              public void actionPerformed(java.awt.event.ActionEvent evt) {
57                  buttInputActionPerformed(evt);
58              }
59          });
60
61          labJob.setText("Jumlah Job : ");
62
63          labPro.setText("Jumlah Proses : ");
64
65          taResult.setEditable(false);
66          taResult.setColumns(20);
67          taResult.setRows(5);
68          jScrollPane1.setViewportView(taResult);
69
70          buttTrain.setText("Train Once");
71          buttTrain.setEnabled(false);
72          buttTrain.addActionListener(new java.awt.event.ActionListener() {
73              public void actionPerformed(java.awt.event.ActionEvent evt) {
74                  buttTrainActionPerformed(evt);
75              }
76          });
77
78          tfDir.setEditable(false);
79
80          buttTrainFull.setText("Train Full");
81          buttTrainFull.setEnabled(false);
82          buttTrainFull.addActionListener(new java.awt.event.ActionListener() {
83              public void actionPerformed(java.awt.event.ActionEvent evt) {
84                  buttTrainFullActionPerformed(evt);
85              }
86          });
87
88          buttReset.setText("Reset");
89          buttReset.setEnabled(false);
90          buttReset.addActionListener(new java.awt.event.ActionListener() {
91              public void actionPerformed(java.awt.event.ActionEvent evt) {
92                  buttResetActionPerformed(evt);
93              }
94          });
95
96          labInfo.setForeground(new java.awt.Color(255, 0, 0));
97
98          labBound.setText("Upper/Lower Bounds : ");
99
100         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
101         getContentPane().setLayout(layout);
102         layout.setHorizontalGroup(
103             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
104             .addGroup(layout.createSequentialGroup()
105                 .addContainerGap()
106                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
107                     .addComponent(labInfo, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                            DEFAULT_SIZE, Short.MAX_VALUE)
108                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
109                         .addComponent(tfDir, javax.swing.GroupLayout.DEFAULT_SIZE, 307, Short.MAX_VALUE)
110                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
111                         .addComponent(buttInput))
112                     .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 380, Short.MAX_VALUE
                            )
113                     .addGroup(layout.createSequentialGroup()
114                         .addComponent(labJob)
115                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 148, Short.
                                MAX_VALUE)
116                         .addComponent(labPro)
117                         .addGap(91, 91, 91))
118                     .addGroup(layout.createSequentialGroup()
119                         .addGap(0, 0, Short.MAX_VALUE)
120                         .addComponent(buttReset)
121                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
122                         .addComponent(buttTrain)
123                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
124                         .addComponent(buttTrainFull))
125                     .addGroup(layout.createSequentialGroup()
126                         .addComponent(labBound)
127                         .addGap(0, 0, Short.MAX_VALUE)))
128                 .addContainerGap())
129         );
130         layout.setVerticalGroup(
131             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
132             .addGroup(layout.createSequentialGroup()
```

```
133                 .addContainerGap()
134                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
135                     .addComponent(buttInput)
136                     .addComponent(tfDir, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                            DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
137                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
138                 .addComponent(labInfo, javax.swing.GroupLayout.PREFERRED_SIZE, 15, javax.swing.GroupLayout
                        .PREFERRED_SIZE)
139                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
140                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
141                     .addComponent(buttTrainFull)
142                     .addComponent(buttTrain)
143                     .addComponent(buttReset))
144                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
145                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
146                     .addComponent(labJob)
147                     .addComponent(labPro))
148                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
149                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 209, javax.swing.
                        GroupLayout.PREFERRED_SIZE)
150                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
151                 .addComponent(labBound)
152                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
153         );
154 
155         pack();
156     }// </editor-fold>
157 
158     private void buttInputActionPerformed(java.awt.event.ActionEvent evt) {
159         File directory = new File(System.getProperty("user.dir"));
160 
161         JFileChooser chooser = new JFileChooser();
162         chooser.setCurrentDirectory(directory);
163 
164         FileNameExtensionFilter filter = new FileNameExtensionFilter("Txt Files", "txt");
165         chooser.setFileFilter(filter);
166 
167         int returnVal = chooser.showOpenDialog(this);
168 
169         if (returnVal == JFileChooser.APPROVE_OPTION){
170             File file = chooser.getSelectedFile();
171             //This is where a real application would open the file.
172             try {
173                 Scanner sc = new Scanner(file);
174                 int   jumlahJob    = sc.nextInt();
175                 int   jumlahProses = sc.nextInt();
176                 int[] jumlahMesin  = new int[jumlahProses];
177                 boolean boolOk = true;
178                 int countMesin = 0;
179                 while(countMesin<jumlahProses){
180                     jumlahMesin[countMesin] = sc.nextInt();
181                     if(jumlahMesin[countMesin]<=0){
182                         boolOk = false;
183                         break;
184                     }
185                     countMesin++;
186                 }
187                 if(boolOk){
188                     Job[] jobDetail    = new Job[jumlahJob];
189                     int count = 0;
190                     while(count<jumlahJob){
191                         int count2 = 0;
192                         int[]proses = new int[jumlahProses];
193                         while(count2<jumlahProses){
194                             proses[count2] = sc.nextInt();
195                             count2++;
196                         }
197                         jobDetail[count] = new Job(count,proses);
198                         count++;
199                     }
200                     int upperBound = sc.nextInt();
201                     int lowerBound = sc.nextInt();
202                     Problem problem = new Problem(jumlahJob, jumlahProses , jobDetail, jumlahMesin,
                            upperBound, lowerBound);
203                     this.problem = problem;
204                     this.aco = new AntColonyAlgorithm(problem);
205 
206                     this.tfDir.setText(file.getName());
207                     this.labInfo.setText("");
208                     this.labJob.setText("Jumlah Job : "+jumlahJob);
209                     this.labPro.setText("Jumlah Proses : "+jumlahProses);
210                     this.labBound.setText("Upper/Lower Bounds : "+upperBound+" / "+lowerBound);
211                     this.buttTrain.setEnabled(true);
212                     this.buttTrainFull.setEnabled(true);
213                     this.buttReset.setEnabled(true);
214                     this.taResult.setText("");
215                 }
216                 else{
217                     this.tfDir.setText(file.getName());
218                     this.labInfo.setText("Input error, banyaknya mesin harus lebih besar dari 0.");
219                     this.labJob.setText("Jumlah Job : "+jumlahJob);
220                     this.labPro.setText("Jumlah Proses : "+jumlahProses);
221                     this.labBound.setText("Upper/Lower Bounds : ");
222                     this.buttTrain.setEnabled(false);
223                     this.buttTrainFull.setEnabled(false);
224                     this.buttReset.setEnabled(false);
225                     this.taResult.setText("");
226                 }
227 
```

```
228             } catch (FileNotFoundException ex) {
229                 Logger.getLogger(TesterGUI.class.getName()).log(Level.SEVERE, null, ex);
230             }
231         }
232     }
233 
234     private void buttTrainActionPerformed(java.awt.event.ActionEvent evt) {
235         try {
236             this.taResult.append(this.aco.trainOnce());
237             System.out.println(this.aco.getPheromoneData());
238         } catch (InterruptedException ex) {
239             Logger.getLogger(TesterGUI.class.getName()).log(Level.SEVERE, null, ex);
240         }
241     }
242 
243     private void buttResetActionPerformed(java.awt.event.ActionEvent evt) {
244         this.aco = new AntColonyAlgorithm(this.problem);
245         this.taResult.setText("");
246     }
247 
248     private void buttTrainFullActionPerformed(java.awt.event.ActionEvent evt) {
249         this.taResult.append(this.aco.trainFull());
250         JOptionPane.showMessageDialog(null, "Train Complete");
251         System.out.println(this.aco.getPheromoneData());
252     }
253 
254     /**
255      * @param args the command line arguments
256      */
257     public static void main(String args[]) {
258         /* Set the Nimbus look and feel */
259         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
260         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
261          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
262          */
263         try {
264             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
                  getInstalledLookAndFeels()) {
265                 if ("Nimbus".equals(info.getName())) {
266                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
267                     break;
268                 }
269             }
270         } catch (ClassNotFoundException ex) {
271             java.util.logging.Logger.getLogger(TesterGUI.class.getName()).log(java.util.logging.Level.
                  SEVERE, null, ex);
272         } catch (InstantiationException ex) {
273             java.util.logging.Logger.getLogger(TesterGUI.class.getName()).log(java.util.logging.Level.
                  SEVERE, null, ex);
274         } catch (IllegalAccessException ex) {
275             java.util.logging.Logger.getLogger(TesterGUI.class.getName()).log(java.util.logging.Level.
                  SEVERE, null, ex);
276         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
277             java.util.logging.Logger.getLogger(TesterGUI.class.getName()).log(java.util.logging.Level.
                  SEVERE, null, ex);
278         }
279         //</editor-fold>
280 
281         /* Create and display the form */
282         java.awt.EventQueue.invokeLater(new Runnable() {
283             public void run() {
284                 new TesterGUI().setVisible(true);
285             }
286         });
287     }
288         // Variables declaration - do not modify
289     private javax.swing.JButton buttInput;
290     private javax.swing.JButton buttReset;
291     private javax.swing.JButton buttTrain;
292     private javax.swing.JButton buttTrainFull;
293     private javax.swing.JScrollPane jScrollPane1;
294     private javax.swing.JLabel labBound;
295     private javax.swing.JLabel labInfo;
296     private javax.swing.JLabel labJob;
297     private javax.swing.JLabel labPro;
298     private javax.swing.JTextArea taResult;
299     private javax.swing.JTextField tfDir;
300     // End of variables declaration
301 }
```