



# *SWARM INTELLIGENCE*

## (Teori & Case Study)

*"PSO, ACO, ABC, ACO-SVR, etc"*

Oleh:  
Imam Cholissodin  
Efi Riyandani

## **PENGANTAR**

Buku ini memberi pemahaman konsep lanjut dan detail penyelesaian untuk pengembangan *Swarm Intelligence* mulai dari dasar-dasar sebagai cara yang paling mudah untuk awalan dalam pemahaman, sampai pemecahan kasus yang komplek yang membutuhkan satu atau beberapa penggabungan dua metode atau lebih. Materi yang tersedia selain memudahkan bagi para pembaca, juga untuk mendukung materi perkuliahan yang dapat membantu pengayaan mahasiswa.

**Imam Cholissodin**

Dosen Pengampu MK Swarm Intelligence FILKOM UB

**2016**

## Kata Pengantar

Alhamdulillahhi robbil alamin, puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya dengan terselesaikannya penulisan buku ini dengan judul “**Swarm Intelligence**”. Buku ini merupakan uraian dari pemahaman konsep tingkat lanjut dalam sistem cerdas dan penerapannya, dengan mengedepankan keterampilan dalam pembuatan dan hasil implementasi dengan berbagai kombinasi algoritma berbasis sistem cerdas. Mata kuliah Swarm Intelligence adalah disiplin keilmuan dari sistem cerdas yang berhubungan dengan sistem alami dan buatan, yang terdiri dari banyak individu yang mengkoordinasikan menggunakan konsep kontrol desentralisasi (kecerdasan sosial dalam berkelompok) dan self-organisasi (kecerdasan personal). Secara khusus, Swarm Intelligence berfokus pada perilaku kolektif yang dihasilkan dari interaksi lokal dari individu dengan satu sama lain dan dengan lingkungan mereka. Swarm Intelligence mempelajari perilaku di alam dari sekawan koloni burung, semut, lebah, dan lainnya dalam penyelesaian kasus pada berbagai bidang. Hal yang paling utama dalam Swarm Intelligence adalah bagaimana membuat representasi individu sebagai solusi yang sesuai dengan kasus yang diselesaikan.

Penulis mengucapkan terimakasih yang sebesar-besarnya kepada beberapa pihak terkait yang telah membantu dalam penyelesaian buku ini:

1. Para penulis artikel tentang *Swarm Intelligence* (SI) di forum, web, blog dan buku yang menjadi referensi buku ini untuk memberikan masukan yang sangat berharga sekali untuk perbaikan dan penyelesaian buku ini.
2. Mbak Efi Riyandani, yang telah banyak membantu penulisan buku ini. Semoga kontribusinya menjadi ilmu yang barokah dan bermanfaat. Amiin. :)

Tidak ada gading yang tak retak, begitulah ungkapan yang tepat terkait dengan buku ini. Maka penulis memohon kritik dan saran untuk perbaikan dan penyempurnaan buku ini. Selamat membaca buku ini dan semoga bermanfaat.

Malang, 8 Juni 2016

Penulis

### **Kontributor Mahasiswa**

Beberapa mahasiswa terbaik saya pada MK Swarm Intelligence di Semester Ganjil 2016-2017 adalah sebagai berikut:

1. Muhammad Fhadli, Daneswara Jauhari, Dhimas Anjar Prabowo, Anang Hanafi, Aryeswara Sunaryo.
2. Reiza Adi Cahya, Nanda Agung Putra, Vera Rusmalawati, Muthia Azzahra, Dyan Dyanmita Putri.
3. Brigitta Ayu Kusuma Wardhany, Istiana Rachmi, Nur Firra Hasjidla, Zulianur Khaqiqiyah, Idham Triatmaja.
4. Husin Muhamad, Cahyo Adi Prasojo, Nur Afifah Sugianto, Listiya Surtiningsih.

Semoga kontribusi kalian menjadi ilmu yang barokah dan bermanfaat. Amiiin. :)

Malang, 8 Juni 2016

Penulis

## Daftar Isi

Judul .....	i
Kata Pengantar.....	ii
Kontributor Mahasiswa .....	iii
Daftar Isi .....	iv
Daftar Tabel.....	viii
Daftar Gambar.....	xii
Daftar Source Code.....	xii
BAB 1 Konsep Swarm Intelligence .....	1
1.1 Pengantar.....	1
1.2 Hirarki Swarm Intelligence .....	3
1.3 Swarm Intelligence.....	4
1.4 Prinsip Kerja Swarm Intelligence .....	5
1.5 Tugas Kelompok .....	6
BAB 2 Dasar-Dasar Algoritma Particle Swarm Optimization.....	7
2.1 Pengantar.....	7
2.2 Struktur Algoritma PSO.....	8
2.3 Studi Kasus : Maksimasi Fungsi Sederhana .....	10
2.3.1 Representasi Partikel.....	11
2.3.2 Inisialisasi.....	11
2.3.3 Update Kecepatan .....	13
2.3.4 Update Posisi.....	14
2.3.5 Update Pbest dan Gbest.....	15
2.4 Kondisi Berhenti.....	16
2.5 Tugas Kelompok .....	18
BAB 3 Real-Code PSO .....	19
3.1 Siklus RCPSO.....	19
3.1.1 Representasi Partikel.....	20
3.1.2 Inisialisasi.....	20
3.1.3 Update Kecepatan .....	22

3.1.4	Update Posisi dan Hitung Fitness.....	23
3.1.5	Update Pbest dan Gbest.....	24
3.2	Time Variant Algoritma PSO.....	26
3.3	Global Best Vs Local Best PSO.....	29
3.4	Penanganan Konvergensi Dini .....	30
3.5	Tugas Kelompok .....	31
BAB 4	Optimasi Masalah Kombinatorial .....	33
4.1	Pengantar.....	33
4.2	Algoritma Hybrid Discrete Particle Swarm Optimization (HDPSO) .....	33
4.2.1	Struktur algoritma HDPSO.....	33
4.2.2	Rumus Update Kecepatan dan Posisi .....	37
4.3	Travelling Salesman Problem (TSP) .....	38
4.3.1	Inisialisasi Kecepatan awal Partikel.....	39
4.3.2	Inisialisasi Posisi awal Partikel .....	39
4.3.3	Inisialisasi Pbest dan Gbest.....	40
4.3.4	Update Kecepatan .....	40
4.3.5	Update Posisi .....	42
4.3.6	Update Pbest dan Gbest.....	42
4.4	Scheduling Problem.....	43
4.4.1	Flow-Shop Scheduling Problem (FSP) .....	43
4.4.2	Two-Stage Assembly Flow-Shop Scheduling Problem .....	44
4.4.3	Job-Shop Scheduling Problem (JSP) .....	45
4.4.4	Flexible Job-Shop Scheduling Problem (FJSP). ....	46
4.5	Tugas Kelompok .....	48
BAB 5	Artificial Bee Colony (ABC).....	50
5.1	Pengantar.....	50
5.2	Real Code Artificial Bee Colony (ABC) .....	50
5.3	Discrete ABC.....	52
5.4	Case Study: Travelling Salesman Problem (TSP).....	56
5.4.1	Inisialisasi Parameter.....	57

5.4.2	Fase Initial.....	57
5.4.3	Improvement Solution (Fase Employeeed Bee) ..	58
5.4.4	Improvement Solution (Fase Onlooker Bee) ....	61
5.4.5	Fase Scout Bee .....	65
5.5	Tugas Kelompok .....	66
BAB 6	Algoritma Ant Colony Optimization (ACO).....	68
6.1	Pengantar.....	68
6.2	Discrete Ant Colony Optimization (ACO).....	68
6.3	Case Study: Travelling Salesman Problem (TSP).....	69
6.3.1	Inisialisasi.....	70
6.3.2	Penyusunan Jalur Kunjungan Setiap Semut .....	70
6.3.3	Cek kondisi berhenti .....	74
6.4	Tugas Kelompok .....	78
BAB 7	Topik Lanjut Pada Swarm Intelligence .....	80
7.1	Pengantar.....	80
7.2	Hybrid ACO dengan SVR .....	82
7.3	Support Vector Regression (SVR).....	83
7.4	Training dan Testing SVR.....	84
7.5	Penyelesaian ACO-SVR .....	92
7.6	Tugas Kelompok .....	95
BAB 8	Project Pilihan Swarm Intelligence .....	97
8.1	Optimasi Penjadwalan Pengajaran Software .....	97
8.1.1	Formulasi Permasalahan.....	97
8.1.2	Implementasi.....	110
8.2	Optimasi Komposisi Menu Makanan .....	126
8.2.1	Formulasi Permasalahan.....	127
8.2.2	Implementasi.....	137
8.3	Optimasi Penjadwalan Praktikum .....	140
8.3.1	Formulasi Permasalahan.....	141
8.3.2	Implementasi.....	151
8.4	Optimasi Naïve Bayes Classifier dengan PSO.....	162

8.4.1	Formulasi Permasalahan .....	162
8.4.2	Implementasi.....	167
Daftar Pustaka.....		176
Indeks .....		184
Biografi Penulis.....		186

## Daftar Tabel

Tabel 4.1 Tabel Jarak Antar Simpul .....	39
Tabel 7.1 Data Nilai Tukar IDR Terhadap USD Juli 2015.....	85
Tabel 7.2 Dataset dengan 4 fitur .....	85
Tabel 7.3 Hasil Normalisasi Data Latih .....	86
Tabel 7.4 Hasil Normalisasi Data Uji.....	86
Tabel 7.5 Data 1 dan 2 .....	86
Tabel 7.6 Jarak Data Latih .....	87
Tabel 7.7 Matrik Hessian.....	87
Tabel 7.8 Nilai $E_i$ .....	88
Tabel 7.9 Nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$ .....	88
Tabel 7.10 Nilai $\alpha_i^*$ dan $\alpha_i$ .....	89
Tabel 7.11 Nilai $f(x)$ Data Latih.....	90
Tabel 7.12 Nilai $f(x)$ Data Uji.....	90
Tabel 7.13 Hasil Denormalisasi Data Latih .....	91
Tabel 7.14 Hasil Denormalisasi Data Uji.....	91
Tabel 7.15 Nilai MAPE .....	91
Tabel 8.1 Data yang Digunakan.....	98
Tabel 8.2 Inisialisasi bee awal pada fase <i>employeed bee</i> .....	98
Tabel 8.3 Gantt-chart bee ke-1 .....	99
Tabel 8.4 Gantt-chart bee ke-2 .....	99
Tabel 8.5 Gantt-chart bee ke-3 .....	99
Tabel 8.6 <i>Improvement</i> SO pada tahap <i>employeed bee</i> .....	100
Tabel 8.7 Gantt-chart bee dengan fitness terbaik .....	100
Tabel 8.8 Hasil <i>improvement</i> SO pada tahap <i>employeed bee</i> .....	101
Tabel 8.9 <i>Improvement</i> SS pada tahap bee ke-1 .....	101
Tabel 8.10 <i>Gantt-chart</i> SS bee ke-1 dengan <i>fitness</i> terbaik .....	102
Tabel 8.11 <i>Improvement</i> SS pada tahap bee ke-2 .....	102
Tabel 8.12 <i>Gantt-chart</i> SS bee ke-2 dengan <i>fitness</i> terbaik .....	102
Tabel 8.13 <i>Improvement</i> SS pada tahap bee ke-3 .....	103

Tabel 8.14 Gantt-chart SS bee ke-3 dengan <i>fitness</i> terbaik .....	103
Tabel 8.15 Hasil <i>improvement</i> SS pada tahap <i>employeed bee</i> .....	104
Tabel 8.16 Nilai probabilitas keseluruhan <i>employeed bee</i> hasil SS	104
Tabel 8.17 Nilai probabilitas kumulatif dan pengecekan kondisi r ...	105
Tabel 8.18 Bee baru hasil seleksi RWS .....	105
Tabel 8.19 <i>Improvement</i> IO pada tahap <i>onlooker bee</i> .....	105
Tabel 8.20 Gantt-chart bee dengan <i>fitness</i> terbaik .....	105
Tabel 8.21 Hasil <i>improvement</i> IO pada tahap <i>onlooker bee</i> .....	106
Tabel 8.22 <i>Improvement</i> IS pada tahap <i>bee</i> ke-1 .....	106
Tabel 8.23 Gantt-chart IS bee ke-1 dengan <i>fitness</i> terbaik.....	107
Tabel 8.24 <i>Improvement</i> IS pada tahap <i>bee</i> ke-2 .....	107
Tabel 8.25 Gantt-chart IS bee ke-2 dengan <i>fitness</i> terbaik.....	107
Tabel 8.26 <i>Improvement</i> IS pada tahap <i>bee</i> ke-3.....	108
Tabel 8.27 Gantt-chart IS bee ke-3 dengan <i>fitness</i> terbaik.....	108
Tabel 8.28 Hasil <i>improvement</i> IS pada tahap <i>onlooker bee</i> .....	109
Tabel 8.29 Global best terpilih.....	109
Tabel 8.30 Bee hasil inisialisasi awal .....	109
Tabel 8.31 Bee hasil <i>improvement</i> terakhir.....	110
Tabel 8.32 Bee baru untuk iterasi selanjutnya .....	110
Tabel 8.33. Daftar Komposisi Zat Gizi Makanan per 100 gram .....	127
Tabel 8.34. Hasil Konversi Komposisi Gizi Data Makanan .....	130
Tabel 8.35. Fitnes tiap partikel .....	133
Tabel 8.36 Data Kelas Praktikum.....	141
Tabel 8.37 Data Sesi Praktikum.....	141
Tabel 8.38 Data Dosen Pengampu .....	142
Tabel 8.39 Data Asisten Praktikum .....	142
Tabel 8.40 Inisialisasi Posisi Awal.....	143
Tabel 8.41 Inisialisasi Kecepatan Awal Partikel .....	144
Tabel 8.42 Pembulatan Menjadi Bilangan Integer .....	144
Tabel 8.43 Repair dan Cek Constraint .....	145
Tabel 8.44 Hasil Repair .....	146

Tabel 8.45 Pbest Partikel .....	147
Tabel 8.46 Gbest Partikel.....	147
Tabel 8.47 Update Kecepatan.....	147
Tabel 8.48 Perbaikan Kecepatan .....	148
Tabel 8.49 Update Posisi .....	148
Tabel 8.50 Representasi Partikel Baru.....	149
Tabel 8.51 Hasil Repair .....	149
Tabel 8.52 Update Pbest.....	150
Tabel 8.53 Update Gbest .....	150

## Daftar Gambar

Gambar 1.1 Ilustrasi .....	1
Gambar 1.2 Konsep <i>Swarm Intelligence</i> .....	2
Gambar 1.3 Macam-Macam <i>Swarm Intelligence</i> .....	3
Gambar 1.4 Simulasi Pergerakan <i>Swarm Intelligence</i> .....	3
Gambar 2.1 <i>Pseudocode</i> Struktur Umum Algoritma PSO .....	9
Gambar 2.2 Ilustrasi Gambaran Algoritma PSO .....	10
Gambar 2.3 Grafik Fungsi Contoh (2.5) .....	10
Gambar 2.4 Contoh Pengujian Konvergensi Berdasarkan Perubahan Nilai Fitness .....	17
Gambar 3.1 Plotting 2D Contoh Fungsi .....	19
Gambar 3.2 Grafik Bobot Inertia w .....	28
Gambar 3.3 Grafik c1 dan c2 .....	28
Gambar 3.4 Ilustrasi Local Best – Initial Swarm dan Local Best – Second Swarm .....	29
Gambar 3.5 Star dan Ring.....	30
Gambar 3.6 Four Clusters dan Von Neumann.....	30
Gambar 3.7 wheel dan pyramid .....	30
Gambar 3.8 Pseudo code algoritma PSO dengan teknik penanganan konvergensi dini dengan <i>random injection</i> .....	31
Gambar 4.1 Contoh Permasalahan TSP.....	38
Gambar 5.1 Contoh Permasalahan TSP.....	56
Gambar 7.1 Nilai Tukar IDR terhadap USD Periode 2006-2015 .....	84
Gambar 7.2 Visualisasi Hasil Peramalan Iterasi SVR 100000.....	92

## **Daftar Source Code**

Source Code 8.1 FSPABC .....	110
Source Code 8.2 Perhitungan Kalori.....	137
Source Code 8.3 Inisialisasi Partikel Bagian 1.....	138
Source Code 8.4 Inisialisasi Partikel Bagian 2.....	138
Source Code 8.5 Update Kecepatan dan Posisi Partikel.....	139
Source Code 8.6 Perhitungan Penalti.....	139
Source Code 8.7 Perhitungan Fitness .....	140
Source Code 8.8 MRCPSO.....	151
Source Code 8.9 NBCPSO .....	167

# BAB 1 Konsep Swarm Intelligence

## 1.1 Pengantar

Pengetahuan yang diberikan oleh sang pencipta (Allah SWT, Tuhan Yang Maha Esa), yang dimiliki oleh sekelompok makhluk hidup memiliki ciri khas yang unik-unik. Jika dilihat dari aktifitas dalam keseharian mereka, maka akan banyak diketahui bahwa perilaku-perilakunya ternyata mampu membentuk kecerdasan yang alami, yang mampu menyelesaikan tugas-tugas dalam kehidupannya. Kecerdasan alamiah inilah yang melahirkan dasar kecerdasan berkelompok (*Swarm Intelligence*). Dalam pengantar ini disajikan ilustrasi pada Gambar 1.1 beberapa kegiatan yang sering dilakukan oleh sekelompok hewan atau secara individu dengan proses atau langkah tertentu, sehingga mereka mampu untuk mencapai tujuannya.

Fakta Keajaiban Alami Tentang *Swarm Intelligence*:



Gambar 1.1 Ilustrasi

- Bagaimana mereka bisa berbuat sedemikian baik seperti hal-hal di atas?
- Kira-kira dimana dan bagaimana mereka belajar, apakah kehidupan mereka juga mirip seperti manusia, yang punya bangsa-bangsa yang berbeda?

Hal-hal yang dapat diambil untuk suatu sistem cerdas sistesis dari alam:

- Kecerdasan Alami (Sosial & Personal)
- Berkoloni (berpasang-pasang),
- Mencari Makanan,
- Regenerasi, dll.

Problem yang mendasar dalam kehidupan sehari-hari dan butuh optimasi dengan sistem cerdas dalam penyelesaiannya:

- Optimasi Penjadwalan (Mesin, Pendidikan, Proyek)
- Optimasi Model Persamaan Matematika yang sangat komplek
- Optimasi Penyusunan Menu Makanan Keluarga
- Optimasi Komposisi Bahan Makanan
- Optimasi Parameter Suatu Algoritma
- Optimasi Seleksi Fitur Dataset
- Solusi yang dapat digunakan untuk teknik optimasi:
- *Swarm Intelligence*
- Algoritma Evolusi

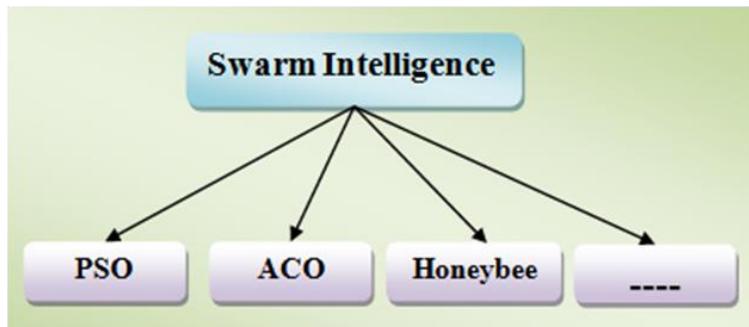
*Swarm Intelligence* adalah disiplin keilmuan dari sistem cerdas yang berhubungan dengan sistem alami dan buatan, yang terdiri dari banyak individu (populasi) yang berkoordinasi menggunakan konsep kontrol desentralisasi (kecerdasan sosial dalam berkelompok) dan *self-organized* (kecerdasan personal/ terorganisir secara mandiri). *Swarm Intelligence* secara singkat bisa juga disebut sebagai kecerdasan berkelompok. Secara umum konsep *swarm intelligence* dapat diperlakukan seperti Gambar 1.2.



Gambar 1.2 Konsep *Swarm Intelligence*

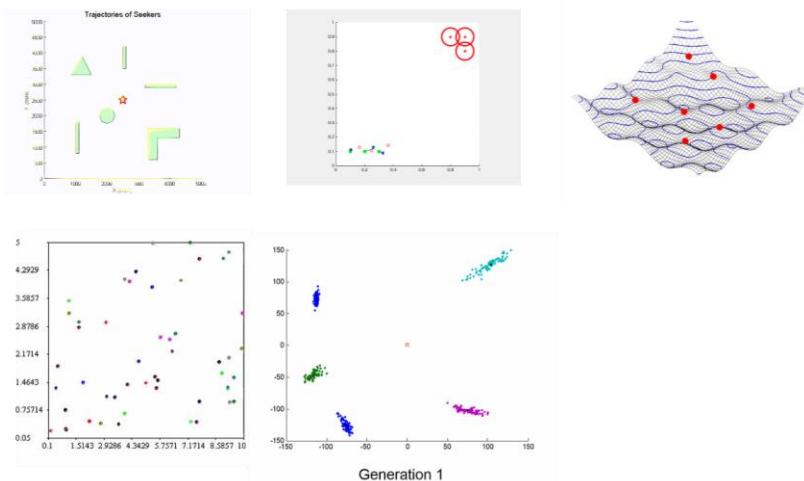
## 1.2 Hirarki Swarm Intelligence

Macam-Macam *Swarm Intelligence* dapat dilihat pada Gambar 1.3.



Gambar 1.3 Macam-Macam *Swarm Intelligence*

Masing-masing algoritma *Swarm Intelligence* memiliki keunikan tersendiri, dan bisa dipastikan setiap mekanisme yang ada pasti ada kelebihan dan keterbatasannya. Misal PSO yang diinspirasi dari sekawanan burung dan ACO dari sekawan semut, dan ABC dari lebah. Berikut simulasi pergerakan *Swarm Intelligence* yang ditunjukkan oleh Gambar 1.4.



Gambar 1.4 Simulasi Pergerakan *Swarm Intelligence*

## 1.3 Swarm Intelligence

Karakteristik dari ***Swarm Intelligence***:

- Mengharuskan untuk membuat *design* ulang representasi solusi yang sesuai untuk setiap *goal* atau *constraint* maupun kasus berbeda.
- Terkadang tidak mencapai solusi global optimal atau mengalami kegagalan (terjebak pada local optimal, konvergensi dini).
- Beberapa mekanisme alami belum dapat dipahami dengan baik.
- Mengharuskan untuk setiap problem optimasi yang akan diselesaikan telah terdefinisi dengan baik, mulai dari *input* datanya apa saja, bagaimana representasi solusinya, *constraints*-nya apa saja, bagaimana rancangan rumus *fitness*-nya atau *cost*-nya.

Representasi **solusi** merupakan Vektor atau matrik sederhana maupun komplek dalam bentuk tertentu dan spesifik, yang dapat berbeda bentuknya atau diubah secara dinamis sesuai dengan cara pandang seseorang terkait pemahamannya terhadap problem yang akan diselesaikan. Representasi solusi bisa disebut juga sebagai bentuk partikel atau individu dalam suatu populasi (kumpulan dari banyak individu) yang memiliki nilai posisi tertentu. Nilai-nilai dalam posisi tersebut menyatakan rangkaian solusi yang dibawa oleh masing-masing individu. Sebagai kandidat solusi optimal jika sudah mencapai kondisi konvergen.

Inti dari konsep algoritma pada *swarm intelligence* adalah terkait dengan pemecahan permasalahan-permasalahan yang sulit sekali untuk dimodelkan dalam bentuk matematika kompleks, dan jika dibuat model persamaan matematikanya akan membutuhkan waktu yang lama. Dan disinilah peranan algoritma *meta-heuristics*, yang sangat dibutuhkan penerapannya.

*Swarm intelligence* termasuk *nature-inspired meta-heuristics*, yaitu *Framework* algoritma yang diinspirasi dari alam dan memiliki kemampuan yang lebih dari banyak sekali pilihan teknik didalamnya yang dapat digunakan untuk mengoptimalkan pada saat pencarian atau penelusuran solusi pada berbagai macam permasalahan yang berbeda (hanya dengan sedikit modifikasi), yang didasarkan pada intuisi maupun aturan-aturan yang ada melalui pendekatan empiris.

## 1.4 Prinsip Kerja Swarm Intelligence

Organizing Principles Swarm In., berikut adalah fitur-fitur utamanya:

- **Autonomy:** Individu dalam sistem mampu bekerja secara mandiri (*otonom*)/ **self-organized**, dalam mengendalikan perilaku mereka sendiri baik di tingkat detektor dan efektor.
- **Adaptability:** Interaksi antara individu dapat timbul melalui komunikasi langsung atau tidak langsung.
- **Scalability:** Mampu men-generate kelompok-kelompok solusi yang terdiri dari beberapa, ribuan individu, atau lebih dengan arsitektur kontrol yang sama.
- **Flexibility:** Tidak ada individu tunggal yang paling utama dalam populasi, jadi setiap individu mendapat perlakuan yang sama, yaitu dapat ditambahkan secara dinamis, dihapus, atau diganti.
- **Robustness:** Tidak ada koordinasi yang terpusat secara pasti pada suatu titik solusi (individu), sehingga memungkinkan tidak ada satu individu pun yang dianggap mengalami kegagalan, karena secara desentralisasi terbantu oleh individu lainnya.
- **Massively parallel:** Tugas yang dilakukan oleh masing-masing individu dalam populasi adalah sama.
- **Self-organization:** Kecerdasan sistem tumbuh tidak hanya bertumpu pada suatu individu, melainkan bertumpu juga dari keseluruhan individu dalam kelompok atau kawanan.

Penjelasan *Direct* dan *Indirect Communication*:

- **Direct Communication:** komunikasi eksplisit yang juga dapat terjadi antara individu. Contoh: *Update Pbest*, *Update Gbest*, *Update Kecepatan Partikel*, *Update Posisi Partikel*, *Waggle dance* di Algoritma *Bee Colony*.
- **Indirect Communication:** Komunikasi implisit yang terjadi antara individu melalui lingkungan sekitarnya. Dikenal sebagai komunikasi *stigmergy* = stigma (*sting*) + ergon (*work*). Artinya *percept* yang diterima individu dari lingkungan ketika individu tersebut memberikan aksi tertentu ke lingkungan.

## 1.5 Tugas Kelompok

1. Jelaskan perbedaan antara *heuristik* sederhana dan *meta-heuristik*?
2. Jelaskan apa yang dimaksud dengan Representasi solusi, populasi, generasi dalam *Swarm Intelligence*!
3. Jelaskan perbedaan antara fungsi *fitness* dan *cost*, yang memang keduanya sama-sama digunakan untuk mengukur kualitas hasil dari suatu solusi?
4. Apa algoritma dalam *Swarm Intelligence* bersifat *stochastic*, jelaskan?
5. Jelaskan perbedaan antara *Constraints* dan *Goal*, dan berikan contohnya!
6. Jelaskan apa yang dimaksud dengan Local Optimal, Local Minimal, Global Optimal, Global Minimal?

## BAB 2 Dasar-Dasar Algoritma Particle Swarm Optimization

### 2.1 Pengantar

*Particle swarm optimization*, disingkat sebagai PSO, didasarkan pada perilaku sebuah kawanan serangga, seperti semut, rayap, lebah atau burung. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata “partikel” menunjukkan individu, misalnya seekor burung dalam kawanan burung. Setiap individu atau partikel berperilaku saling terhubung dengan cara menggunakan kecerdasannya (*intelligence*) sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh di kelompok tersebut.

Metode optimasi yang didasarkan pada *swarm intelligence* ini disebut algoritma *behaviorally inspired* sebagai alternatif dari algoritma genetika, yang sering disebut *evolution-based procedures*. Algoritma PSO ini awalnya diusulkan oleh J. Kennedy dan R. C. Eberhart (Kennedy, 1995). Dalam konteks optimasi multi-variabel, kawanan diasumsikan mempunyai ukuran tertentu atau tetap dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik posisi dan kecepatan.

PSO memiliki tiga komponen utama diantaranya: **partikel**, komponen **kognitif** dan komponen **sosial**, serta **kecepatan** partikel. Dan setiap partikel merepresentasikan solusi penyelesaian. Pembelajaran partikel terdiri dari dua faktor yaitu pengalaman partikel (disebut *cognitive learning*) dan kombinasi pembelajaran dari keseluruhan *swarm* (disebut *social learning*). *Cognitive learning* sebagai *pBest* yaitu posisi terbaik yang pernah dicapai sebuah partikel, sedangkan *social learning* sebagai *gBest* yaitu posisi terbaik dari keseluruhan partikel dalam *swarm*. *pBest* dan *gBest* untuk menghitung kecepatan partikel, kecepatan untuk menghitung posisi selanjutnya.

## 2.2 Struktur Algoritma PSO

Terdapat beberapa komponen dalam Algoritma PSO diantaranya adalah sebagai berikut:

- **Swarm**, merupakan jumlah partikel dalam populasi pada suatu algoritma. Ukuran *swarm* bergantung pada seberapa kompleks masalah yang dihadapi. Secara umum, ukuran *swarm* pada algoritma PSO cenderung lebih kecil jika dibandingkan dengan algoritma *evolusioner* yang lain dalam mencari solusi terbaik.
- **Partikel**, merupakan individu dalam suatu *swarm* yang merepresentasikan solusi penyelesaian masalah. Setiap partikel memiliki posisi dan kecepatan yang ditentukan oleh representasi solusi pada saat itu.
- **Personal best (*pBest*)**, merupakan posisi terbaik yang pernah dicapai partikel dengan membandingkan *fitness* pada posisi partikel sekarang dengan sebelumnya. *Personal best* dipersiapkan untuk mendapatkan solusi terbaik.
- **Global Best (*gBest*)**, merupakan posisi terbaik partikel yang diperoleh dengan membandingkan nilai *fitness* terbaik dari keseluruhan partikel dalam *swarm*.
- **Kecepatan (*velocity*)**,  $v$  merupakan vektor yang menentukan arah perpindahan posisi partikel. Perubahan *velocity* dilakukan setiap iterasi dengan tujuan memperbaiki posisi partikel semula.
- **Bobot inersia (*inertia weight*)**,  $w$  digunakan untuk mengontrol dampak dari perubahan *velocity* yang diberikan oleh partikel.
- **Koefisien akselerasi**, merupakan faktor pengontrol sejauh mana partikel berpindah dalam satu iterasi. Secara umum nilai koefisien akselerasi  $c_1$  dan  $c_2$  adalah sama yaitu dalam rentang 0 sampai 4. Namun demikian, nilai tersebut dapat ditentukan sendiri untuk setiap penelitian berbeda.

*Pseudo-code* sebagai struktur umum algoritma PSO disajikan pada Gambar 2.1 sebagai berikut:

```
procedure AlgoritmaPSO
begin
    t = 0
    inisialisasi posisi partikel ( $x_{i,j}^t$ ) , kecepatan ( $v_{i,j}^t$ ) ,  $Pbest_{i,j}^t = x_{i,j}^t$  ,
        hitung fitness tiap partikel, dan  $Gbest_{g,j}^t$ 
    do
        t = t + 1
        update kecepatan  $v_{i,j}(t)$ 
        update posisi  $x_{i,j}(t)$ 
        hitung fitness tiap partikel
        update  $Pbest_{i,j}(t)$  dan  $Gbest_{g,j}(t)$ 
    while (bukan kondisi berhenti)
end
```

Gambar 2.1 *Pseudocode Struktur Umum Algoritma PSO*

Untuk rumus yang digunakan pada algoritma PSO dapat dilihat pada persamaan 2.1 -2.4 berikut:

**Rumus update kecepatan (*velocity*):**

$$v_{i,j}^{t+1} = w.v_{i,j}^t + c_1.r_1(Pbest_{i,j}^t - x_{i,j}^t) + c_2.r_2(Gbest_{g,j}^t - x_{i,j}^t) \quad (2.1)$$

**Rumus update posisi:**

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1} \quad (2.2)$$

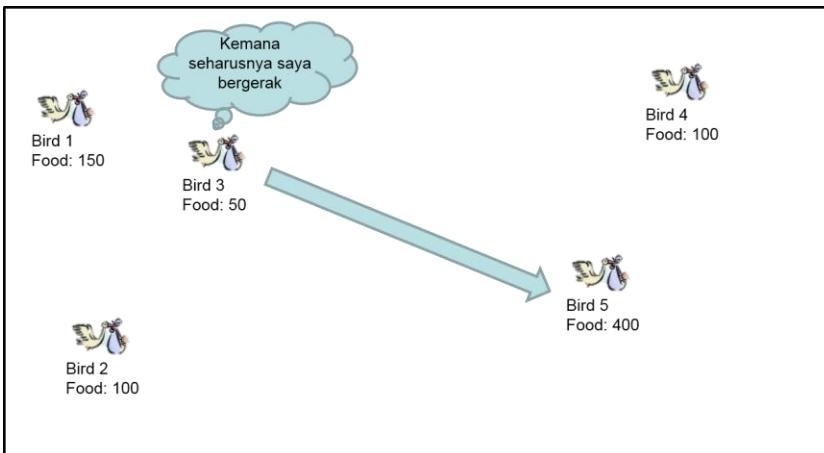
**Rumus Sigmoid dan update posisi (untuk *binary code*):**

$$sig(v_{i,j}^t) = \frac{1}{1 + e^{-v_{i,j}^t}}, \quad j = 1, 2, \dots, d \quad (2.3)$$



$$x_{i,j}^{t+1} = \begin{cases} 1, & \text{if } rand[0,1] < sig(v_{i,j}^{t+1}) \\ 0, & \text{otherwise} \end{cases}, \quad j = 1, 2, \dots, d \quad (2.4)$$

Ilustrasi gambaran algoritma PSO ditunjukkan pada Gambar 2.2 sebagai berikut:



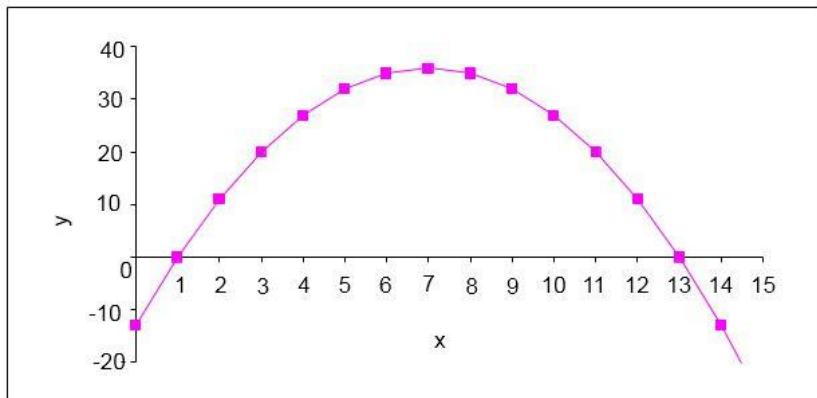
Gambar 2.2 Ilustrasi Gambaran Algoritma PSO

## 2.3 Studi Kasus : Maksimasi Fungsi Sederhana

Untuk menjelaskan siklus PSO maka diberikan contoh sederhana masalah maksimasi (mencari nilai maksimum) dari sebuah fungsi sebagai berikut:

$$\max, y = f(x) = -x^2 + 14x - 13, \quad 0 \leq x \leq 15 \quad (2.5)$$

Grafik dari fungsi tersebut ditunjukkan pada Gambar 2.3. Nilai maksimum fungsi adalah  $y=36$  pada  $x=7$



Gambar 2.3 Grafik Fungsi Contoh (2.5)

### 2.3.1 Representasi Partikel

Dalam siklus perkembangan algoritma *Particle Swarm Optimization* mencari solusi (partikel dengan ukuran dimensi tertentu) ‘terbaik’ terdapat beberapa proses sebagai berikut:

1. **Inisialisasi**
  - Inisialisasi Kecepatan Awal
  - Inisialisasi Posisi awal Partikel
  - Inisialisasi Pbest dan Gbest
2. **Update Kecepatan**
3. **Update Posisi dan Hitung Fitness** (Seperti pada algoritma evolusi, fungsi objektif mengukur seberapa dekat solusi dengan optimum, contohnya fungsi objektif mengukur performansi atau kualitas partikel, dalam hal ini adalah  $f(x)$ )
4. **Update Pbest dan Gbest**

### 2.3.2 Inisialisasi

Inisialisasi dilakukan untuk membangkitkan himpunan solusi baru secara acak/*random* yang terdiri atas sejumlah *string* dimensi partikel dan ditempatkan pada penampungan yang disebut populasi. Dalam tahap ini harus ditentukan ukuran populasi (*popSize*). Nilai ini menyatakan banyaknya individu/partikel yang ditampung dalam populasi. Panjang setiap string dimensi partikel (*stringLen*) dihitung berdasarkan presisi variabel solusi yang kita cari.

Misalkan kita tentukan *popSize*=4 dan kita gunakan representasi partikel biner (bilangan basis 2).

Nilai  $x$  ditentukan antara 0 sampai 15 dan bilangan biner dengan panjang 4 sudah dapat menjangkau nilai  $x$  (ingat  $1111_2 = 15$ ). Jadi *stringLen(d)*=4,  $d$  adalah banyaknya dimensi partikel.

$$(1 \times 2^{4-1}) + (1 \times 2^{4-2}) + (1 \times 2^{4-3}) + (1 \times 2^{4-4}) = 8 + 4 + 2 + 1 = 15$$

#### 1. Inisialisasi Kecepatan awal Partikel

Pada iterasi ke-0 ( $t=0$ ), dapat dipastikan bahwa nilai kecepatan awal semua partikel di ( $v_{i,j}(t)=0$ ) dan konversi partikelnya menjadi  $x$  sebagai berikut:

$$popSize=4, y = f(x) = -x^2 + 14x - 13, 0 \leq x \leq 15$$

kecepatan		[0000]
$v_1(0)$	[0 0 0 0]	
$v_2(0)$	[0 0 0 0]	
$v_3(0)$	[0 0 0 0]	
$v_4(0)$	[0 0 0 0]	

[ $v_{i=1,j=1}^{t=0}$     $v_{1,2}^0$     $v_{1,3}^0$     $v_{1,4}^0$ ]  
atau dapat ditulis dengan  
 $[v_{i=1,j=1}(t=0) \quad v_{1,2}(0) \quad v_{1,3}(0) \quad v_{1,4}(0)]$

## 2. Inisialisasi Posisi awal Partikel

Nilai  $x$  ditentukan antara 0 sampai 15 dan bilangan biner dengan panjang 4 sudah dapat menjangkau nilai  $x$  (ingat  $1111_2 = 15$ ). Jadi  $stringLen=4$ .

$$(1 \times 2^{4-1}) + (1 \times 2^{4-2}) + (1 \times 2^{4-3}) + (1 \times 2^{4-4}) = 8 + 4 + 2 + 1 = 15$$

Misalkan dari hasil *random* didapatkan populasi inisial pada iterasi ke-0 ( $t=0$ ) dan konversinya menjadi  $x$  sebagai berikut:

$$popSize=4, y = f(x) = -x^2 + 14x - 13, 0 \leq x \leq 15$$

partikel		$x$	$y=f(x)$	[0000]
$x_1(0)$	[0 0 1 1]	3	20	
$x_2(0)$	[0 1 0 0]	4	27	
$x_3(0)$	[1 0 0 1]	9	32	
$x_4(0)$	[0 1 0 1]	5	32	

[ $x_{i=1,j=1}^{t=0}$     $x_{1,2}^0$     $x_{1,3}^0$     $x_{1,4}^0$ ]  
atau dapat ditulis dengan  
 $[x_{i=1,j=1}(t=0) \quad x_{1,2}(0) \quad x_{1,3}(0) \quad x_{1,4}(0)]$

## 3. Inisialisasi *Pbest* dan *Gbest*

***Pbest***, karena masih iterasi ke-0 ( $t=0$ ), maka nilai *Pbest* akan disamakan dengan nilai posisi awal partikel, yaitu ( $Pbest_{i,j}(t)=x_{i,j}(t)$ ) dan konversinya menjadi  $x$  sebagai berikut:

$$popSize=4, y = f(x) = -x^2 + 14x - 13, 0 \leq x \leq 15$$

Pbest	x	y=f(x)	
$Pbest_{1,0}$	[0011]	3	20
$Pbest_{2,0}$	[0100]	4	27
$Pbest_{3,0}$	[1001]	9	32
$Pbest_{4,0}$	[0101]	5	32

[0011]

↓

atau dapat ditulis dengan

$[Pbest_{i=1,j=1}^{t=0}, Pbest_{1,2}^0, Pbest_{1,3}^0, Pbest_{1,4}^0]$

$[Pbest_{i=1,j=1}(t=0), Pbest_{1,2}(0), Pbest_{1,3}(0), Pbest_{1,4}(0)]$

**Gbest**, dicari dengan memilih satu  $Pbest$  yang *fitness*-nya tertinggi (yang nilai  $y$  nya paling tinggi), ( $k=\arg\max_i \{ fitness_{Pbest_{i,j}}(t) \}$ ), karena ada 2 nilai  $y$  yang tertinggi, maka cukup diambil salah satu, misal  $Gbest_{g=1,j}(t=0)=Pbest_{k=3,j}(t=0)$ .

Masuk pada iterasi ke-1, ( $t = t + 1 = 0 + 1 = 1$ )

### 2.3.3 Update Kecepatan

Update Kecepatan dilakukan untuk menentukan arah perpindahan posisi partikel yang ada di populasi. Batasan *lower* dan *upper* kecepatan yang digunakan dalam proses ini berdasarkan nilai maksimum dari posisi partikel ( $x_{max}$ ).

Diketahui nilai *lower* dan *upper* untuk  $x_{i,j}=[xmin, xmax]=[0,1]$ , karena memang nilai  $x_{i,j}$  tersebut biner, dan untuk  $v_{i,j}=[vmin, vmax]$ , dimana  $vmin=-vmax$  dan  $vmax=60\% * xmax$ . Dalam menentukan interval ini, baik untuk posisi maupun kecepatan partikel, sebaiknya dilakukan pada proses inisialisasi.

Cara untuk menghitung interval kecepatan adalah sebagai berikut:

$$vmax = 60\% * xmax = 60\% * 1 = 0.6 \text{ dan } vmin = -vmax = -0.6$$

$$\text{sehingga } v_{i,j} = [-0.6, 0.6]$$

Jika diketahui  $x_{i,j} = [0,1]$ ,  $v_{i,j} = [-0.6, 0.6]$ , dan misalkan diketahui  $w=0.5$ ,  $c_1=1$ ,  $c_2=1$ ,  $r_1=\text{rand}[0,1]$ , dan  $r_2=\text{rand}[0,1]$ . Dimana  $r_1$  dan  $r_2$  adalah bilangan acak dalam interval  $[0, 1]$ . Maka untuk mendapatkan hasil update kecepatannya dihitung sebagai berikut dengan menggunakan persamaan 2.1, misal menghitung  $v_{1,1}(1)$ .

$$v_{i,j}^{t+1} = w.v_{i,j}^t + c_1.r_1(Pbest_{i,j}^t - x_{i,j}^t) + c_2.r_2(Gbest_{g,j}^t - x_{i,j}^t)$$

$$\begin{aligned} v_{1,1}^{0+1} &= w.v_{1,1}^0 + c_1.r_1(Pbest_{1,1}^0 - x_{1,1}^0) + c_2.r_2(Gbest_{1,1}^0 - x_{1,1}^0) \\ &= (0.5)(0) + (1)(0.1)(0 - 0) + (1)(0.4)(1 - 0) = 0 + 0 + 0.4 = 0.4 \end{aligned}$$

Pbest	$x$	$y=f(x)$	kecepatan
$Pbest_1(0)$	[ 0 0 1 1 ]	3 20	$v_1(1)$ [ 0.4 0.0 -0.4 0.0 ]
$Pbest_2(0)$	[ 0 1 0 0 ]	4 27	$v_2(1)$ [ 0.4 -0.4 0.0 0.4 ]
$Pbest_3(0)$	[ 1 0 0 1 ]	9 32	$v_3(1)$ [ 0.0 0.0 0.0 0.0 ]
$Pbest_4(0)$	[ 0 1 0 1 ]	5 32	$v_4(1)$ [ 0.4 -0.4 0.0 0.0 ]

## 2.3.4 Update Posisi

$$x_{i,j}^0 \quad \Rightarrow \quad x_{i,j}^{0+1}$$

Setelah menghitung kecepatan, kemudian hitung  $Sig(v_{i,j}(t))$ . Berikut cara menghitungnya.

$$v_{i,j}^{t+1} = w.v_{i,j}^t + c_1.r_1(Pbest_{i,j}^t - x_{i,j}^t) + c_2.r_2(Gbest_{g,j}^t - x_{i,j}^t)$$

$$sig(v_{i,j}^t) = \frac{1}{1 + e^{-v_{i,j}^t}}, \quad j = 1, 2, \dots, d$$

Misal hanya diambil 1 angka dibelakang titik desimal

kecepatan		$Sig(\text{kecepatan})$	
$v_1(1)$	[ 0.4 0.0 -0.4 0.0 ]	$Sig(v_1(1))$	[ 0.6 0.5 0.4 0.5 ]
$v_2(1)$	[ 0.4 -0.4 0.0 0.4 ]	$Sig(v_2(1))$	[ 0.6 0.4 0.5 0.6 ]
$v_3(1)$	[ 0.0 0.0 0.0 0.0 ]	$Sig(v_3(1))$	[ 0.5 0.5 0.5 0.5 ]
$v_4(1)$	[ 0.4 -0.4 0.0 0.0 ]	$Sig(v_4(1))$	[ 0.6 0.4 0.5 0.5 ]

Membuat rand[0,1], lalu hitung hasil update posisi dan *fitness*

rand[0,1]	partikel x	y=f(x)
[ 0.3 0.8 0.1 0.7 ]	x <sub>1</sub> (1) [ 1 0 1 0 ]	10 27
[ 0.5 0.6 0.3 0.8 ]	x <sub>2</sub> (1) [ 1 0 1 0 ]	10 27
[ 0.8 0.6 0.3 0.6 ]	x <sub>3</sub> (1) [ 0 0 1 0 ]	2 11
[ 0.9 0.1 0.5 0.8 ]	x <sub>4</sub> (1) [ 0 1 1 0 ]	6 35

$$x_{i,j}^{t+1} = \begin{cases} 1, & \text{if } \text{rand}[0,1] < \text{sig}(v_{i,j}^{t+1}) \\ 0, & \text{otherwise} \end{cases}$$



### 2.3.5 Update Pbest dan Gbest

$$Pbest_{i,j}^0 \Rightarrow Pbest_{i,j}^{0+1}$$

- Update *Pbest*, disini kita harus membandingkan antara *Pbest* pada iterasi sebelumnya dengan hasil dari Update Posisi dengan menggunakan persamaan 2.6 berikut.

$$Pbest_{i,j}^{t+1} = \begin{cases} Pbest_{i,j}^t & \text{jika } fitness(x_{i,j}^{t+1}) \leq fitness(Pbest_{i,j}^t) \\ x_{i,j}^{t+1} & \text{jika } fitness(x_{i,j}^{t+1}) > fitness(Pbest_{i,j}^t) \end{cases} \quad (2.6)$$

Pbest	x	y=f(x)	partikel	x	y=f(x)
Pbest <sub>1</sub> (0) [ 0 0 1 1 ]	3	20	x <sub>1</sub> (1) [ 1 0 1 0 ]	10	27
Pbest <sub>2</sub> (0) [ 0 1 0 0 ]	4	27	x <sub>2</sub> (1) [ 1 0 1 0 ]	10	27
Pbest <sub>3</sub> (0) [ 1 0 0 1 ]	9	32	x <sub>3</sub> (1) [ 0 0 1 0 ]	2	11
Pbest <sub>4</sub> (0) [ 0 1 0 1 ]	5	32	x <sub>4</sub> (1) [ 0 1 1 0 ]	6	35

VS

- Dari 2 tabel di atas, cek dari urutan baris yang sama, kemudian dibandingkan *fitness*-nya, manakah yang lebih tinggi nilainya akan menjadi *Pbest* terbaru.

	<i>Pbest</i>	<i>x</i>	<i>y=f(x)</i>
<i>Pbest<sub>1</sub>(1)</i>	[ 1 0 1 0 ]	10	27
<i>Pbest<sub>2</sub>(1)</i>	[ 0 1 0 0 ]	4	27
<i>Pbest<sub>3</sub>(1)</i>	[ 1 0 0 1 ]	9	32
<b><i>Pbest<sub>4</sub>(1)</i></b>	<b>[ 0 1 1 0 ]</b>	<b>6</b>	<b>35</b>

3. Dan *Pbest* terbaru dengan nilai *fitness* tertinggi akan menjadi *Gbest*.

	<i>Gbest</i>	<i>X</i>	<i>y=f(x)</i>
<i>Gbest<sub>1</sub>(1)</i>	[ 0 1 1 0 ]	6	35

4. Dari hasil *Gbest* tersebut dapat digunakan untuk kesimpulan sebagai hasil terbaik pada iterasi ke-1.

	<i>Gbest</i>	<i>X</i>	<i>y=f(x)</i>
<i>Gbest<sub>1</sub>(1)</i>	[ 0 1 1 0 ]	6	35

5. Kemudian, jika dilanjutkan iterasi berikutnya ( $t = t + 1$ ), maka langkah di bawah ini akan diulang terus-menerus sampai iterasi Maksimum atau telah mencapai konvergen.

1. **Update Kecepatan**
2. **Update Posisi dan Hitung Fitness**
3. **Update *Pbest* dan *Gbest***

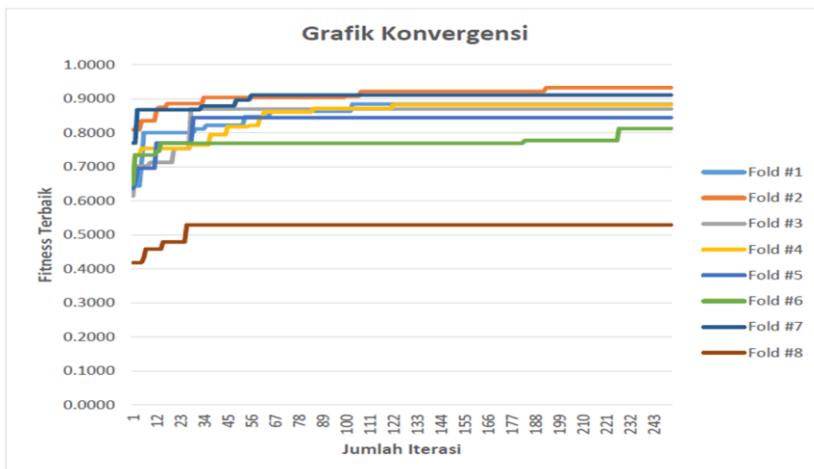
## 2.4 Kondisi Berhenti

Kondisi pemberhentian pencarian solusi optimal, dalam beberapa literatur umumnya algoritma PSO menggunakan beberapa hal berikut:

1. Iterasi berhenti sampai Iterasi Maksimum yang sudah ditentukan berdasarkan beberapa kali pengujian.

2. Iterasi berhenti setelah diketahui tidak ada perubahan yang signifikan “search stagnation” (dari nilai selisih absolut solusi terbaik saat ini dan sebelumnya  $< \text{epsilon}$ ) dari hasil nilai evaluasi (misal nilai fitness) yang mengindikasikan proses pencarian solusinya sudah mencapai konvergen.
3. Iterasi berhenti setelah  $t$  satuan waktu tercapai ( $t \geq t_{\max}$ ) atau dengan dikombinasikan dari banyaknya evaluasi fungsi yang sudah dijalankan.

Dalam implementasi, umumnya kondisi (1) sering dipakai, tetapi memang harus tetap selektif, artinya pada kasus yang berbeda akan membutuhkan proses iterasi yang berbeda pula, sesuai dengan kompleksitas permasalahannya. Berikut contoh pengujian konvergensi berdasarkan perubahan nilai fitness:



Gambar 2.4 Contoh Pengujian Konvergensi Berdasarkan Perubahan Nilai Fitness

Hal diatas digunakan untuk memastikan bahwa nilai Iterasi maksimum yang digunakan pada saat proses pencarian sudah dilakukan uji tersebut. Sehingga dengan besar Iterasi maksimum tersebut nilainya dapat dikatakan ideal sesuai kasus, dan yang lebih penting lagi adalah kondisi pemberhentiannya adalah berhenti setelah konvergen, bukan sebaliknya.

## 2.5 Tugas Kelompok

1. Jelaskan perbedaan antara *Swarm Intelligence* dan Algoritma Evolusi?
2. Jelaskan konsep desentralisasi pada PSO!
3. Jelaskan pengertian tentang istilah Partikel, pBest, gBest, Pembaruan Kecepatan, dan Posisi dalam PSO!
4. Buatlah komparasi Algoritma PSO dengan Algoritma Genetika!
5. Jelaskan indikator suatu permasalahan optimasi dikatakan sangat kompleks!
6. Jelaskan pentingnya kondisi pemberhentian iterasi harus diupayakan berhenti setelah konvergen, bukan sebaliknya!
7. Berdasarkan hasil  $Gbest_1(1)$  pada sub bab 2.3.5, lanjutkan penghitungan iterasinya untuk mendapatkan  $Gbest_1(2)$ !
8. Sebutkan dan jelaskan beberapa kelebihan dan keterbatasan dari algoritma PSO?

## BAB 3 Real-Code PSO

### 3.1 Siklus RCPSO

Kelemahan algoritma PSO dengan pengkodean biner jika digunakan pada optimasi fungsi adalah tidak bisa menjangkau beberapa titik solusi jika range solusi berada dalam daerah kontinyu. Selain itu, pada optimasi fungsi yang kompleks dan membutuhkan banyak generasi, operasi transformasi biner ke bilangan desimal (real) dan sebaliknya sangat menyita waktu.

**Solusi** : Pengkodean real (*real-coded particle swarm optimization, RCPSO*). Contoh fungsi sederhana yang dibuat lebih banyak variabel, dan dalam bentuk bilangan desimal, yaitu:

$$\max, f(x_1, x_2) = 19 + x_1 \sin(x_1\pi) + (10 - x_2) \sin(x_2\pi),$$

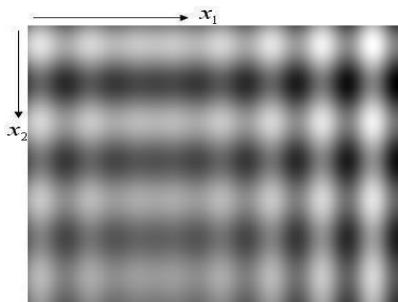
$$-5,0 \leq x_1 \leq 9,8 \quad 0,0 \leq x_2 \leq 7,3$$

Untuk memperjelas uraian pada bagian sebelumnya, sekarang bagaimana menangani angka pecahan (desimal) serta penggunaan Nilai Parameter Algoritma PSO, yaitu TVIW dan TVAC, maka diberikan lagi contoh sederhana masalah maksimasi (mencari nilai maksimum) dari sebuah fungsi sebagai berikut:

$$\max, f(x_1, x_2) = 19 + x_1 \sin(x_1\pi) + (10 - x_2) \sin(x_2\pi),$$

$$-5,0 \leq x_1 \leq 9,8 \quad 0,0 \leq x_2 \leq 7,3$$

Plotting dua dimensi dari fungsi diatas ditunjukkan pada Gambar 3.1. Warna putih menunjukkan nilai fungsi yang lebih besar. Perhatikan bahwa fungsi ini mempunyai banyak nilai maksimum lokal.



Gambar 3.1 Plotting 2D Contoh Fungsi

### 3.1.1 Representasi Partikel

Dalam kasus ini variabel keputusan ( $x_1$  dan  $x_2$ ) langsung menjadi dimensi pada partikel, sehingga panjang dimensi partikel adalah 2. Dalam siklus Real-Code algoritma *Particle Swarm Optimization* terdapat beberapa proses sebagai berikut:

1. Inisialisasi
  - Inisialisasi Kecepatan Awal
  - Inisialisasi Posisi awal Partikel
  - Inisialisasi Pbest dan Gbest
2. Update Kecepatan
3. Update Posisi dan Hitung Fitness (Seperti pada algoritma evolusi, fungsi objektif mengukur seberapa dekat solusi dengan optimum, contohnya fungsi objektif mengukur performansi atau kualitas partikel, dalam hal ini adalah  $f(x_1, x_2)$ )
4. Update Pbest dan Gbest

### 3.1.2 Inisialisasi

#### 1. Inisialisasi Kecepatan Awal Partikel

Pada iterasi ke-0 ( $t=0$ ), dapat dipastikan bahwa nilai kecepatan awal semua partikel di ( $v_{i,j}(t)=0$ ) dan misal kecepatan partikelnya sebagai berikut:

$$\text{popSize}=10, \max, f(x_1, x_2) = 19 + x_1 \sin(x_1\pi) + (10 - x_2) \sin(x_2\pi), \\ -5,0 \leq x_1 \leq 9,8 \quad 0,0 \leq x_2 \leq 7,3$$

Kecepatan	
j=1 untuk $x_1$	j=2 untuk $x_2$
$v_1(0)$	0
$v_2(0)$	0
$v_3(0)$	0
$v_4(0)$	0
$v_5(0)$	0
$v_6(0)$	0
$v_7(0)$	0
$v_8(0)$	0
$v_9(0)$	0
$v_{10}(0)$	0

[ 0 0 ]  
↓  
 $\begin{bmatrix} v_{i=1, j=1}^{t=0} & v_{1,2}^0 \end{bmatrix}$   
atau dapat ditulis dengan  
 $[v_{i=1, j=1}(t=0) \quad v_{1,2}(0)]$

## 2. Inisialisasi Posisi awal Partikel

Partikel inisial, yaitu pada iterasi ke-0 ( $t=0$ ) dibangkitkan secara random dengan persamaan.  $x = x_{\min} + \text{rand}[0,1] * (x_{\max} - x_{\min})$ , dan misal dihasilkan populasi sebagai berikut:

$$\text{popSize}=10, \text{ max}, f(x_1, x_2) = 19 + x_1 \sin(x_1 \pi) + (10 - x_2) \sin(x_2 \pi), \\ -5,0 \leq x_1 \leq 9,8 \quad 0,0 \leq x_2 \leq 7,3$$

partikel			$f(x_i, x_j)$
	$j=1$ untuk $x_1$	$j=2$ untuk $x_2$	
$x_1(0)$	1,4898	2,0944	19,8206
$x_2(0)$	8,4917	2,5754	34,7058
$x_3(0)$	1,4054	6,3035	20,6707
$x_4(0)$	5,8114	5,0779	14,5624
$x_5(0)$	-1,8461	1,7097	11,5858
$x_6(0)$	4,0206	4,4355	24,7106
$x_7(0)$	-0,1634	2,974	19,653
$x_8(0)$	5,2742	0,7183	22,1813
$x_9(0)$	9,4374	6,6919	12,4694
$x_{10}(0)$	-4,5575	0,1679	28,4324

[1,4898 2,0944]

$\left[ x_{i=1, j=1}^{t=0} \quad x_{1,2}^0 \right]$

atau dapat ditulis dengan

$\left[ x_{i=1, j=1}(t=0) \quad x_{1,2}(0) \right]$

## 3. Inisialisasi *Pbest* dan *Gbest*

**Pbest**, karena masih iterasi ke-0 ( $t=0$ ), maka nilai *Pbest* akan disamakan dengan nilai posisi awal partikel, yaitu ( $Pbest_{i,j}(t)=x_{i,j}(t)$ ) dan misal *Pbest*-nya sebagai berikut:

$$\text{popSize}=10, \text{ max}, f(x_1, x_2) = 19 + x_1 \sin(x_1 \pi) + (10 - x_2) \sin(x_2 \pi),$$

$$-5,0 \leq x_1 \leq 9,8 \quad 0,0 \leq x_2 \leq 7,3$$

Pbest			$f(x_i, x_j)$
	$j=1$	$j=2$	
$Pbest_1(0)$	1,4898	2,0944	19,8206
<b><math>Pbest_2(0)</math></b>	<b>8,4917</b>	<b>2,5754</b>	<b>34,7058</b>
$Pbest_3(0)$	1,4054	6,3035	20,6707
$Pbest_4(0)$	5,8114	5,0779	14,5624
$Pbest_5(0)$	-1,8461	1,7097	11,5858
$Pbest_6(0)$	4,0206	4,4355	24,7106
$Pbest_7(0)$	-0,1634	2,974	19,653
$Pbest_8(0)$	5,2742	0,7183	22,1813
$Pbest_9(0)$	9,4374	6,6919	12,4694
$Pbest_{10}(0)$	-4,5575	0,1679	28,4324

[1,4898 2,0944]

$\downarrow$

$\left[ Pbest_{i=1, j=1}^{t=0} \quad Pbest_{1,2}^0 \right]$

atau dapat ditulis dengan

$\left[ Pbest_{i=1, j=1}(t=0) \quad Pbest_{1,2}(0) \right]$

**Gbest**, memilih satu *Pbest* yang *fitness*-nya tertinggi, ( $k=\text{argMax}_i \{ \text{fitness}_{Pbest_{i,j}(t)} \} = 2$ ), maka  $Gbest_{g=1,j}(t=0) = Pbest_{k=2,j}(t=0)$ .

Masuk pada iterasi ke-1, ( $t = t + 1 = 0 + 1 = 1$ )

### 3.1.3 Update Kecepatan

Dalam implementasi PSO, terkadang ditemukan bahwa kecepatan partikel bergerak ke nilai yang besar dengan cepat, terutama untuk posisi partikel yang jauh dari posisi tetangga dan individu terbaik. Akibatnya, partikel tersebut memiliki kecenderungan untuk meninggalkan (keluar) dari ruang batas pencarian atau posisi baru partikel berada di luar range solusi. Oleh karena itu, untuk mengontrol eksplorasi global partikel, perlu adanya pembatasan kecepatan minimum dan maksimum. Teknik pembatasan ini disebut *velocity clamping* untuk mencegah partikel bergerak terlalu jauh melampaui ruang pencarinya.

Batasan lower dan upper kecepatan yang digunakan dalam proses ini berdasarkan nilai minimum dan maksimum pada setiap dimensi dari representasi solusi partikel ( $x_{min,d}$  dan  $x_{max,d}$ ), dimana  $d$  menyatakan dimensi, atau bisa dituliskan  $x_{i,j}=[x_{min_j} ; x_{max_j}]$

Diketahui nilai lower dan upper pada  $x_1$ , maka jika dibentuk intervalnya menjadi  $x_{i,j=1}=[x_{min_1} ; x_{max_1}]=[-5 ; 9,8]$ ,  $x_{i,2}=[x_{min_2} ; x_{max_2}]=[0 ; 7,3]$ , dimana  $v_{min_j}=-v_{max_j}$  dan

$$v_{max_j} = k \frac{(x_{max_j} - x_{min_j})}{2} \quad k \in (0,1] \quad (\text{Marini \& Walczak, 2015})$$

misal  $k=0,6$  maka

$$v_{max_{j=1}} = 0,6 \frac{(9,8 - (-5))}{2} = 4,44 \quad v_{max_2} = 0,6 \frac{(7,3 - 0)}{2} = 2,19$$

sehingga untuk  $v_{i,j=1}=[v_{min_1} ; v_{max_1}] = [-4,44 ; 4,44]$  dan

$$v_{i,j=2}=[v_{min_2} ; v_{max_2}] = [-2,19 ; 2,19]$$

Dalam menentukan interval pada langkah sebelumnya, baik untuk posisi maupun kecepatan partikel untuk setiap dimensi, sebaiknya dilakukan pada proses inisialisasi. Batasan kecepatan atau threshold yang digunakan adalah sebagai berikut (Marini & Walczak, 2015):

jika  $v_{ij}^{t+1} > v_{max_j}$  maka  $v_{ij}^{t+1} = v_{max_j}$

jika  $v_{ij}^{t+1} < -v_{max_j}$  maka  $v_{ij}^{t+1} = -v_{max_j}$

Jika diketahui  $x_{i,1} = [x_{min_1} ; x_{max_1}] = [-5 ; 9,8]$ ,  $x_{i,2} = [x_{min_2} ; x_{max_2}] = [0 ; 7,3]$ , serta telah didapatkan  $v_{i,1} = [-4,44 ; 4,44]$  dan  $v_{i,2} = [-2,19 ; 2,19]$

dan misalkan diketahui  $w = 0.5$ ,  $c_1 = 1$ ,  $c_2 = 1$ ,  $r_1 = \text{rand}[0,1]$ , dan  $r_2 = \text{rand}[0,1]$ . Dimana  $r_1$  dan  $r_2$  adalah bilangan acak dalam interval  $[0, 1]$ . Maka untuk mendapatkan hasil update kecepatannya dihitung sebagai berikut, misal menghitung  $v_{1,1}(1)$ .

$j=1$ untuk $x_1$	$j=1$ untuk $x_1$	$P_{\text{best}}$	$G_{\text{best}}$
$v_1(0)$	$x_1(0)$	$P_{\text{best},1}(0)$	$G_{\text{best},1}(0)$

$$v_{i,j}^{t+1} = w.v_{i,j}^t + c_1.r_1(P_{\text{best}}_{i,j}^t - x_{i,j}^t) + c_2.r_2(G_{\text{best}}_{g,j}^t - x_{i,j}^t)$$

$$\begin{aligned} v_{1,1}^{0+1} &= w.v_{1,1}^0 + c_1.r_1(P_{\text{best}}_{1,1}^0 - x_{1,1}^0) + c_2.r_2(G_{\text{best}}_{1,1}^0 - x_{1,1}^0) \\ &= (0.5)(0) + (1)(0.1)(1.4898 - 1.4898) + (1)(0.4)(8.4917 - 1.4898) \\ &= 0 + 0 + 2.8001 = 2.8001 \end{aligned}$$

Berikut keseluruhan hasil update kecepatannya:

$$\begin{aligned} v_{1,1}^{0+1} &= w.v_{1,1}^0 + c_1.r_1(P_{\text{best}}_{1,1}^0 - x_{1,1}^0) + c_2.r_2(G_{\text{best}}_{1,1}^0 - x_{1,1}^0) \\ &= (0.5)(0) + (1)(0.1)(1.4898 - 1.4898) + (1)(0.4)(8.4917 - 1.4898) \\ &= 0 + 0 + 2.8001 = 2.8001 \end{aligned}$$

Kecepatan		Kecepatan perbaikan	
$j=1$ untuk $x_1$	$j=2$ untuk $x_2$	$j=1$ untuk $x_1$	$j=2$ untuk $x_2$
$v_1(1)$	2.80076	$v_1(1)$	2.80076
$v_2(1)$	0	$v_2(1)$	0
$v_3(1)$	2.83452	$v_3(1)$	2.83452
$v_4(1)$	-1.001	$v_4(1)$	-1.001
$v_5(1)$	4.13512	$v_5(1)$	4.13512
$v_6(1)$	0.34628	$v_6(1)$	0.34628
$v_7(1)$	1.78844	$v_7(1)$	1.78844
$v_8(1)$	-0.74404	$v_8(1)$	-0.74404
$v_9(1)$	3.46204	$v_9(1)$	3.46204
$v_{10}(1)$	-0.15944	$v_{10}(1)$	-0.15944
$v_1(1)$	1.287	$v_1(1)$	1.287
$v_2(1)$	0.74284	$v_2(1)$	0.74284
$v_3(1)$	-0.37828	$v_3(1)$	-0.37828
$v_4(1)$	-1.6466	$v_4(1)$	-1.6466
$v_5(1)$	5.21968	$v_5(1)$	5.21968
$v_6(1)$	0.963	$v_6(1)$	0.963

### 3.1.4 Update Posisi dan Hitung Fitness

- Update posisi dihitung sebagai berikut, misal hitung  $x_{1,1}(1)$ .

partikel		Kecepatan	
$j=1$	$j=1$ untuk $x_1$	$j=1$	$j=1$ untuk $x_1$
$x_1(0)$	1,4898	$v_1(1)$	2.80076

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}$$

$$x_{1,1}^{0+1} = x_{1,1}^0 + v_{1,1}^{0+1} = 1.4898 + 2.80076 = 4.2906$$

$$x_{i,1} = [xmin_1 ; xmax_1] = [-5 ; 9,8]$$

$$x_{i,2} = [xmin_2 ; xmax_2] = [0 ; 7,3]$$

2. Berikut keseluruhan hasil update posisinya:

	partikel		$f(x_1, x_2)$
	$j=1$	$j=2$	
$x_1(1)$	4.2906	2.2868	28.4696
$x_2(1)$	8.4917	2.5754	34.7023
$x_3(1)$	4.2399	4.8123	24.7773
$x_4(1)$	6.8835	4.0769	22.8537
$x_5(1)$	2.2890	2.0560	22.2186
$x_6(1)$	5.8090	3.6915	10.5710
$x_7(1)$	3.2986	2.8146	20.2634
$x_8(1)$	6.5612	1.4611	16.9525
$x_9(1)$	9.0591	5.0453	16.4912
$x_{10}(1)$	-0.1175	1.1309	15.4854

### 3.1.5 Update Pbest dan Gbest

1. Update Pbest, disini kita harus membandingkan antara Pbest pada iterasi sebelumnya dengan hasil dari Update Posisi.

Pbest			partikel				
$j=1$	$j=2$	$f(x_1, x_2)$	$j=1$	$j=2$	$f(x_1, x_2)$		
$Pbest_1(0)$	1,4898	2,0944	19,8206	$x_1(1)$	4.2906	2.2868	28.4696
$Pbest_2(0)$	8,4917	2,5754	34,7058	$x_2(1)$	8.4917	2.5754	34.7023
$Pbest_3(0)$	1,4054	6,3035	20,6707	$x_3(1)$	4.2399	4.8123	24.7773
$Pbest_4(0)$	5,8114	5,0779	14,5624	$x_4(1)$	6.8835	4.0769	22.8537

$Pbest_5(0)$	-1,8461	1,7097	11,5858	$x_5(1)$	2.2890	2.0560	22.2186
$Pbest_6(0)$	4,0206	4,4355	24,7106	$x_6(1)$	5.8090	3.6915	10.5710
$Pbest_7(0)$	-0,1634	2,974	19,653	$x_7(1)$	3.2986	2.8146	20.2634
$Pbest_8(0)$	5,2742	0,7183	22,1813	$x_8(1)$	6.5612	1.4611	16.9525
$Pbest_9(0)$	9,4374	6,6919	12,4694	$x_9(1)$	9.0591	5.0453	16.4912
$Pbest_{10}(0)$	-4,5575	0,1679	28,4324	$x_{10}(1)$	-0.1175	1.1309	15.4854

2. Dari 2 tabel di atas, cek dari urutan baris yang sama, kemudian dibandingkan *fitness*-nya, manakah yang lebih tinggi nilainya akan menjadi *Pbest* terbaru.

	Pbest		
	$j=1$	$j=2$	$f(x_1, x_2)$
$Pbest_1(1)$	4.2906	2.2868	28.4696
$Pbest_2(1)$	8,4917	2,5754	34,7058
$Pbest_3(1)$	4.2399	4.8123	24.7773
$Pbest_4(1)$	6.8835	4.0769	22.8537
$Pbest_5(1)$	2.2890	2.0560	22.2186
$Pbest_6(1)$	4,0206	4,4355	24,7106
$Pbest_7(1)$	3.2986	2.8146	20.2634
$Pbest_8(1)$	5,2742	0,7183	22,1813
$Pbest_9(1)$	9.0591	5.0453	16.4912
$Pbest_{10}(1)$	-4,5575	0,1679	28,4324

3. Dan *Pbest* terbaru dengan nilai *fitness* tertinggi akan menjadi *Gbest*.

Gbest

	$j=1$	$j=2$	$f(x_1, x_2)$
$Gbest_1(1)$	8,4917	2,5754	34,7058

4. Dan  $Pbest$  terbaru dengan nilai *fitness* tertinggi akan menjadi  $Gbest$ .

	Gbest		
	$j=1$	$j=2$	$f(x_1, x_2)$
$Gbest_1(1)$	8,4917	2,5754	34,7058

5. Kemudian, jika dilanjutkan iterasi berikutnya ( $t = t + 1$ ), maka langkah di bawah ini akan diulang terus-menerus sampai iterasi Maksimum atau telah mencapai konvergen.

1. **Update Kecepatan**
2. **Update Posisi dan Hitung Fitness**
3. **Update  $Pbest$  dan  $Gbest$**

## 3.2 Time Variant Algoritma PSO

Time variant yang digunakan adalah time varying acceleration coefficients (TVAC) dan time varying inertia weight (TVIW). Time variant inilah yang digunakan untuk mengontrol kemampuan PSO dalam lokal pencarian secara efisien dan konvergensi ke solusi optimum global. Adapun nilai minimum dan maksimum  $w$  (TVIW) yang digunakan adalah 0.4 dan 0.9 nilai ini terbukti dapat meningkatkan solusi optimum pada pencarian banyak masalah, sedangkan nilai range  $c_1$  dan  $c_2$  (TVAC) yang digunakan adalah [2.5, 0.5] dan [0.5, 2.5] karena terbukti optimal (Ratnaweera, 2004). Bobot inertia  $w$  diperbarui untuk mendapatkan nilai  $w$  yang adaptif untuk setiap iterasi, sehingga nilainya bisa dinamis dan mampu meningkatkan hasil optimasi yang diharapkan, semakin besar nilai iterasinya, maka nilai  $w$  akan semakin kecil, dan sebaiknya, jika iterasinya masih diawal-awal, maka nilai  $w$  akan cenderung lebih besar. Artinya, jika nilai  $w$  semakin besar, maka partikel lebih difokuskan ke arah eksplorasi, tetapi ketika semakin kecil, maka lebih difokuskan ke arah eksplotasi. Detail uraiannya seperti pada persamaan TVIW berikut (Chen at all, 2011):

$$w = w_{\min} + (w_{\max} - w_{\min}) \frac{(t_{\max} - t)}{t_{\max}} \quad (3.1)$$

Dimana:

$w_{\max}$  : nilai maksimum bobot *inertia w*

$w_{\min}$  : nilai minimum bobot *inertia w*

$t$  : iterasi awal dari algoritma

$t_{\max}$  : nilai maksimum iterasi.

Pada  $c_1$  dan  $c_2$  adalah koefisien percepatan untuk keseimbangan yang lebih baik antara global eksplorasi dan lokal eksplorasi. Konsep ini akan diadopsi untuk solusi pencarian yang lebih baik. Inti TVAC adalah  $c_1$  menurunkan dari nilai inisial  $c_{1i}$  sampai  $c_{1f}$ , saat  $c_2$  menaikkan dari  $c_{2i}$  sampai  $c_{2f}$  berdasarkan persamaan TVAC secara matematika sebagai berikut (Chen et all, 2011):

$$c_1 = (c_{1f} - c_{1i}) \frac{t}{t_{\max}} + c_{1i} \quad c_2 = (c_{2f} - c_{2i}) \frac{t}{t_{\max}} + c_{2i} \quad (3.2)$$

Dimana:

$c_1$  dan  $c_2$  adalah koefisien percepatan untuk keseimbangan yang lebih baik antara global eksplorasi yaitu *fitness* terbaik yang dicapai oleh semua partikel dalam topologi ketetanggaan dan lokal eksplorasi yaitu *fitness* terbaik yang dicapai oleh satu partikel saat ini, *Time varying acceleration coefficients* (TVAC).

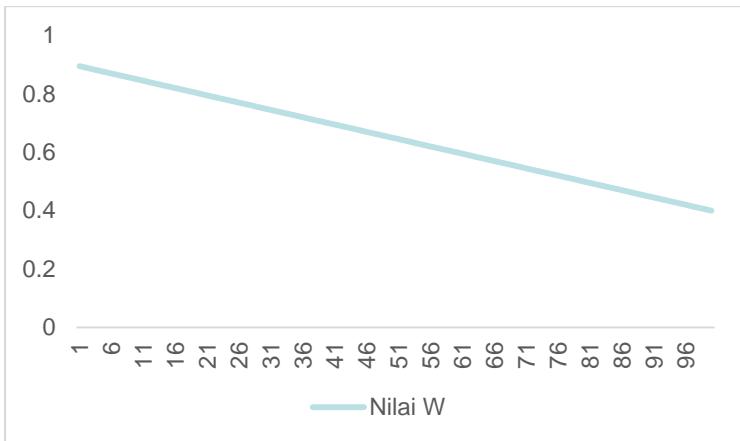
dimana  $c_{1f}, c_{1i}, c_{2f}, c_{2i}$  adalah konstanta, misalkan:

1. Bobot inertia  $w=[0.4, 0.9]$ :

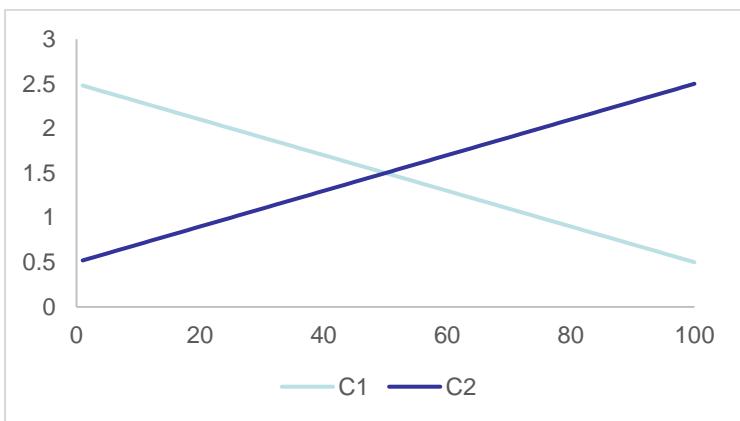
$$w = w_{\min} + (w_{\max} - w_{\min}) \frac{(t_{\max} - t)}{t_{\max}}$$

2. Pada  $c_1=[2.5, 0.5]$  dan  $c_2=[0.5, 2.5]$  :

$$c_1 = (c_{1f} - c_{1i}) \frac{t}{t_{\max}} + c_{1i} \quad c_2 = (c_{2f} - c_{2i}) \frac{t}{t_{\max}} + c_{2i}$$



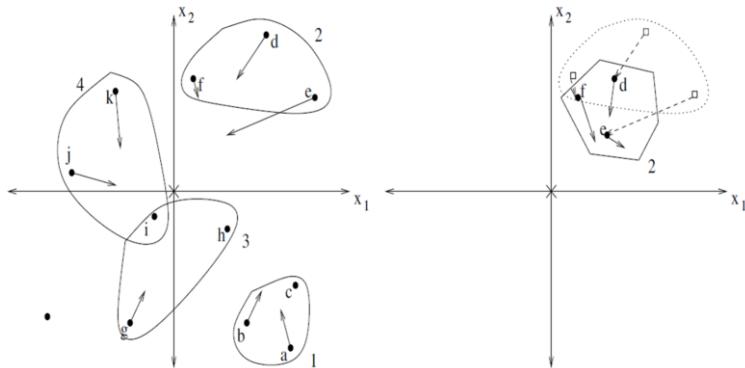
Gambar 3.2 Grafik Bobot Inertia w



Gambar 3.3 Grafik c1 dan c2

### 3.3 Global Best Vs Local Best PSO

Local best PSO, atau *lbest* PSO, menggunakan topologi jaringan dalam bentuk ring, dimana tetangga terdekat sebanyak  $nN_i$  harus ditentukan (dapat menggunakan jarak euclidean) untuk setiap partikel ke- $i$ .



Gambar 3.4 Ilustrasi Local Best – Initial Swarm dan Local Best – Second Swarm

Komponen sosial ini merefleksikan pertukaran informasi dalam ketetanggaan partikel, yang mengisyaratkan bahwa masing-masing partikel mengetahui keadaan lokal lingkungannya dengan jangkauan sebanyak  $nN_i$ . Berikut rumus untuk *update* kecepatannya:

#### Rumus Local best PSO, atau *lbest* PSO

$$v_{i,j}^{t+1} = w.v_{i,j}^t + c_1.r_1(Pbest_{i,j}^t - x_{i,j}^t) + c_2.r_2(lbest_{i,j}^t - x_{i,j}^t) \quad (3.3)$$

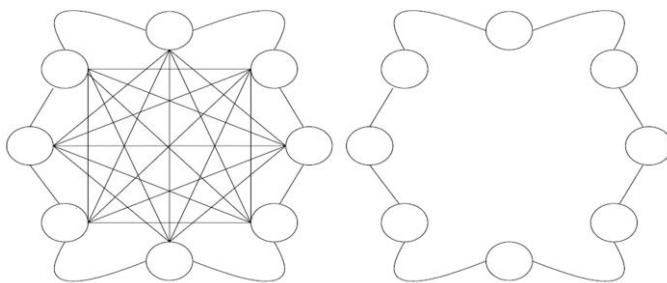
#### Rumus Global best PSO

$$v_{i,j}^{t+1} = w.v_{i,j}^t + c_1.r_1(Pbest_{i,j}^t - x_{i,j}^t) + c_2.r_2(Gbest_{g,j}^t - x_{i,j}^t) \quad (3.4)$$

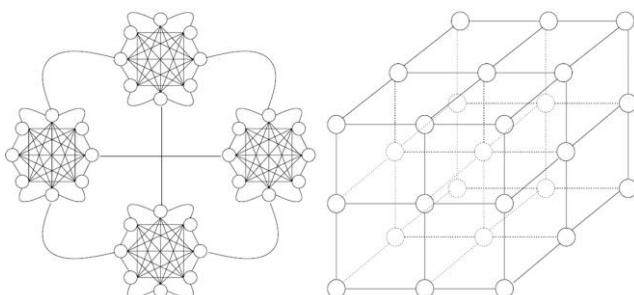
Pada Global best, besarnya nilai  $nN_i = popSize$ , sedangkan untuk update posisi masih menggunakan persamaan yang sama, yaitu:

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}$$

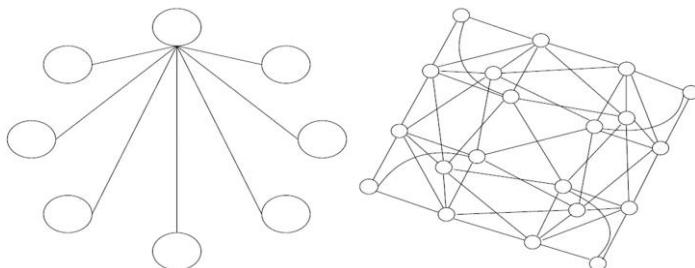
Macam-macam struktur topologi jaringan sosial pada PSO:



Gambar 3.5 Star dan Ring



Gambar 3.6 Four Clusters dan Von Neumann



Gambar 3.7 wheel dan pyramid

### 3.4 Penanganan Konvergensi Dini

Pada ruang pencarian yang tidak terlalu besar, sering dijumpai pencapaian konvergensi dini. Hal ini disebabkan karena partikel lebih cepat menemukan posisi terbaik global dalam ruang pencarian yang kecil dan disebabkan oleh kurangnya diversitas populasi setelah melewati sekian generasi (Mahmudy, 2014). Hal tersebut dapat diatasi dengan menerapkan sistem *random injection*. *Random injection*

dilakukan dengan menginisialisasi kembali posisi  $n$  partikel setiap  $g$  interval iterasi. Penentuan  $n$  dan  $g$  yang sesuai dilakukan berdasarkan beberapa percobaan sebelumnya pada PSO. Pseudo code algoritma PSO dengan teknik penanganan konvergensi dini dengan *random injection* ditunjukkan pada Gambar 3.8 berikut:

```
procedure AlgoritmaPSO
begin
    t = 0
    inisialisasi posisi partikel ( $x_{i,j}^t$ ), kecepatan ( $v_{i,j}^t$ ),  $Pbest_{i,j}^t = x_{i,j}^t$ ,
                                                hitung fitness tiap partikel, dan  $Gbest_{g,j}^t$ 
    do
        t = t + 1
        update kecepatan  $v_{i,j}(t)$ 
        update posisi  $x_{i,j}(t)$ 
        hitung fitness tiap partikel
        update  $Pbest_{i,j}(t)$  dan  $Gbest_{g,j}(t)$ 
        if (t mod g = 0)
            replace n individu
        end
    while (bukan kondisi berhenti)
end
```

Gambar 3.8 Pseudo code algoritma PSO dengan teknik penanganan konvergensi dini dengan *random injection*

- Misal, jumlah partikel yang diinjeksi ( $n$ ) = 30% dari ukuran *swarm size* atau jumlah partikel dan interval *injection* adalah setiap kelipatan 3 iterasi.
- Random injection dilakukan dengan mengevaluasi nilai *fitness* partikel saat memasuki interval *injection*.
- Nilai *fitness* akan diurutkan, kemudian 30% dari jumlah partikel dengan nilai *fitness-nya* terkecil akan digantikan dengan partikel yang baru.
- Untuk setiap partikel baru tersebut di-*generate* secara random, dan masing-masing partikel diberikan nilai kecepatan  $v = 0$  dan *pBest* sama dengan partikel itu sendiri. Hal ini sama seperti proses inisialisasi awal partikel.

## 3.5 Tugas Kelompok

1. Jelaskan perbedaan konsep binary code dan real code pada PSO!
2. Jelaskan perbedaan antara *Gbest* dan *Lbest* PSO! dan jelaskan beberapa kelebihan dan keterbatasan dari masing-masing! Dan

manakah dari kedua teknik tersebut yang lebih baik hasil optimasinya?

3. Jelaskan konsep  $w$ ,  $c_1$ ,  $c_2$  dalam TVIW dan TVAC! Dan jelaskan kenapa nilai  $w$  dan  $c_1$ , dengan bertambahnya iterasi dibuat monoton turun, tidak dibuat naik. Tetapi  $c_2$  dibuat monoton naik, tidak dibuat turun (berdasarkan pada grafik Time Variant PSO)!
4. Pada perhitungan  $v_{\min}$  dan  $v_{\max}$  pada slide ke-10, ubahlah rumus perhitungan  $v_{\max}$  dengan rumus berikut

$$v_{\max_j} = k(x_{\max_j} - x_{\min_j}) \quad k \in (0,1] \quad (\text{Engelbrecht, 2007})$$

Dengan  $k=0.6$  dan kemudian lanjutkan penghitungannya untuk mendapatkan  $v_{i,j}(1)$ !

5. Sebutkan dan jelaskan beberapa macam-macam struktur topologi jaringan sosial pada PSO?
6. Perhatikan sub bab 3.1.2 poin 3, jika Gbest diganti dengan Lbest PSO dan diketahui  $nN_i=2$  ( $nN_i$  adalah banyaknya Neigborhood terdekat pada partikel ke- $i$ ). Tentukan hasil Gbest<sub>1</sub>(1)! Dimana Gbest<sub>1</sub>(1) adalah Lbest <sub>$i,i$</sub> (1) yang memiliki nilai fitness tertinggi.

#### Local best PSO, atau *Ibest* PSO:

$$v_{i,j}^{t+1} = w.v_{i,j}^t + c_1.r_1(Pbest_{i,j}^t - x_{i,j}^t) + c_2.r_2(lbest_{i,j}^t - x_{i,j}^t)$$

7. Berdasarkan hasil Gbest<sub>1</sub>(1) pada sub bab 3.1.5, lanjutkan penghitungan iterasinya untuk mendapatkan Gbest<sub>1</sub>(2), tetapi dengan menggunakan konsep TVIW dan TVAC, seperti pada sub bab 3.2 (artinya nilai  $w$ ,  $c_1$ , dan  $c_2$  dibuat dinamis, misal diketahui nilai  $t_{\max}=100000$ )!

$$w = w_{\min} + (w_{\max} - w_{\min}) \frac{(t_{\max} - t)}{t_{\max}}$$

$$c_1 = (c_{1f} - c_{1i}) \frac{t}{t_{\max}} + c_{1i} \quad c_2 = (c_{2f} - c_{2i}) \frac{t}{t_{\max}} + c_{2i}$$

## BAB 4 Optimasi Masalah Kombinatorial

### 4.1 Pengantar

Optimasi masalah kombinatorial adalah suatu kasus yang penyelesaiannya dapat diperoleh dengan mengubah susunan dari posisi masing-masing nilai permutasi dalam bentuk integer dari representasi solusinya. Meskipun kasus kombinatorial bisa diselesaikan secara **brute force**, namun akan memerlukan waktu tidak cepat (efisiensinya kurang), jika kasusnya memang kompleks, sehingga tidak bisa diterima secara praktis. Dalam hal ini, algoritma **Hybrid Discrete Particle Swarm Optimization (HDPSO)** dapat diterapkan untuk mengatasi masalah kombinatorial dengan lebih praktis dan teratur (karena terdapat mekanisme pencarian dan evaluasi yang sangat sangat terstruktur).

### 4.2 Algoritma Hybrid Discrete Particle Swarm Optimization (HDPSO)

Pada tahun 2000, Clerc memodifikasi algoritma DPSO yang telah dirumuskan oleh Kennedy dan Eberhart. Clerc memodifikasi representasi posisi pada partikel, bentuk kecepatan yang dihasilkan oleh partikel serta pengaruh kecepatan terhadap posisi partikel. Harapan dari modifikasi tersebut adalah untuk dapat diaplikasikan pada permasalahan dengan model diskrit khususnya bertipe kombinatorial (Clerc, 2000).

#### 4.2.1 Struktur algoritma HDPSO

Berikut adalah struktur dari algoritma HDPSO yang dimodifikasi oleh Maurice Clerc pada tahun 2000:

1. Posisi Partikel

$$x_i^t = \begin{bmatrix} x_{i1}^t & x_{i2}^t & .. & x_{id}^t \end{bmatrix}$$

Dimana:

$x_i(t)$  merupakan posisi partikel ke- $i$  pada iterasi ke- $t$ , dan partikel tersebut memiliki sebanyak  $d$  dimensi.

## 2. Transposisi

Transposisi, yaitu cara untuk menukar dua buah nilai pada dimensi tertentu berdasarkan urutan index dari posisi partikel.

## 3. Velocity

Velocity, proses transposisi sebanyak  $\|v_i'\|$  antara dua posisi index. Velocity didefinisikan sebagai berikut:

$$v_i' = ((a_k, b_k)), a \in \{1, 2, \dots, d\}, b \in \{1, 2, \dots, d\}, k \uparrow_1^{\|v_i'\|}$$

dimana  $\|v_i'\|$  merupakan banyaknya daftar dari transposisi,  $a$  dan  $b$  adalah index dimensi partikel yang akan ditukar nilainya. Contoh, misal  $v_1 = ((1,3), (2,5))$  dan  $v_2 = ((2,5), (1,3))$ , maka  $v_1 @ v_2$  dikatakan tidak sama tetapi keduanya kongruen.

## 4. Opposite of a velocity

Berdasarkan pada point (3) di atas, dapat dikatakan bahwa  $v_1$  adalah opposite dari  $v_2$ , yaitu  $v_1 = -v_2$  dan dibisa dituliskan  $v_2 = -v_1$ . Sehingga berlaku bentuk umum bahwa  $-v = v$  serta  $v \oplus -v \cong \emptyset$

$$\text{jika } v_i' = ((a_k, b_k)), k \uparrow_1^{\|v_i'\|}$$

$$\text{maka } -v_i' = ((a_k, b_k)), k \downarrow_1^{\|v_i'\|} = \left( \left( a_{\|v_i'\|-k+1}, b_{\|v_i'\|-k+1} \right) \right), k \uparrow_1^{\|v_i'\|}$$

Contoh:

$$v_i' = ((1,3), (3,2), (4,5)) \xrightarrow{} v_i' = ((a_1, b_1), (a_2, b_2), (a_3, b_3)), \|v_i'\| = 3$$

$$-v_i' = ((4,5), (3,2), (1,3))$$

$$-v_i' = ((1,3), (3,2), (4,5))$$

## 5. Move (addition) "position plus velocity"

Misalkan Update posisi  $x_i^{t+1} = x_i^t + v_i^{t+1}$  dioperasikan dengan cara memproses mulai dari urutan pertama transposisi  $v$  ke posisi  $x$ , kemudian urutan selanjutnya, sampai urutan akhir dari  $v$ .

Contoh:

$$\begin{aligned}
 x_1^0 &= [2 \ 5 \ 1 \ 3 \ 4] \quad \xrightarrow{\hspace{1cm}} \quad x_i^t = [x_{i1}^t \ x_{i2}^t \ \dots \ x_{id}^t] \\
 v_1^1 &= ((1,3), (3,2), (4,5)) \quad \xrightarrow{\hspace{1cm}} \quad v_i^t = ((a_1, b_1), (a_2, b_2), (a_3, b_3)), \|v_i^t\| = 3 \\
 x_i^{t+1} &= x_i^t + v_i^{t+1} \quad \xrightarrow{\hspace{1cm}} \quad x_1^1 = [2 \ 5 \ 1 \ 3 \ 4] + ((1,3), (3,2), (4,5)) \\
 x_1^1 &= x_1^0 + v_1^1 \quad \xrightarrow{\hspace{1cm}} \quad x_1^1 = [1 \ 5 \ 2 \ 3 \ 4] + ((3,2), (4,5)) \\
 &\quad \quad \quad x_1^1 = [1 \ 2 \ 5 \ 3 \ 4] + ((4,5)) \\
 &\quad \quad \quad x_1^1 = [1 \ 2 \ 5 \ 4 \ 3]
 \end{aligned}$$

#### 6. Subtraction "position minus position"

Misal terdapat dua posisi  $x_i(t)$  dan  $x_i(t+1)$ . Pengurangan  $x_i(t+1) - x_i(t)$  didefinisikan sebagai sebuah kecepatan  $v_i(t+1)$ . Sehingga dengan mengaplikasikan kecepatan  $v_i(t+1)$  ke dalam posisi  $x_i(t)$  menghasilkan  $x_i(t+1)$ . Difference didefinisikan sebagai berikut.

$$\begin{aligned}
 x_i^{t+1} - x_i^t &= v_i^{t+1} \\
 x_i^{t+1} - x_i^t = v_i^{t+1} &\Leftrightarrow x_i^t + v_i^{t+1} = x_i^{t+1} \\
 \text{jika } x_i^{t+1} &= x_i^t, \text{ maka } x_i^{t+1} - x_i^t = v_i^{t+1} = \emptyset \\
 x_1^1 &= [2 \ 5 \ 1 \ 3 \ 4] + ((1,3), (3,2), (4,5))
 \end{aligned}$$

Contoh:

$$\begin{aligned}
 x_1^1 &= [1 \ 2 \ 5 \ 4 \ 3] \quad x_1^0 = [2 \ 5 \ 1 \ 3 \ 4] \\
 v_1^1 &= x_1^1 - x_1^0 = ((1,3), (3,2), (4,5))
 \end{aligned}$$

#### 7. Addition "velocity plus velocity"

Misalkan terdapat dua velocity  $v_1$  dan  $v_2$ , untuk menambah kecepatan  $v_1$  dengan  $v_2$  ( $v_1 \oplus v_2$ ), dilakukan operasi penjumlahan yaitu dengan cara urutan transposisi  $v_1$  kemudian diikuti dengan urutan transposisi  $v_2$ . Sehingga addition dapat didefinisikan sebagai berikut.

$$\begin{aligned}
 v &= v_1 \oplus v_2 \\
 v &= \left( (a_1, b_1)_1, (a_2, b_2)_1, \dots, \left( a_{\|v_1\|}, b_{\|v_1\|} \right)_1 \right) \oplus \left( (a_1, b_1)_2, (a_2, b_2)_2, \dots, \left( a_{\|v_2\|}, b_{\|v_2\|} \right)_2 \right) \\
 v &= \left( (a_1, b_1)_1, (a_2, b_2)_1, \dots, \left( a_{\|v_1\|}, b_{\|v_1\|} \right)_1, (a_1, b_1)_2, (a_2, b_2)_2, \dots, \left( a_{\|v_2\|}, b_{\|v_2\|} \right)_2 \right)
 \end{aligned}$$

Contoh:

$$v_1 = ((1,3),(3,2),(4,5)) \quad v_2 = ((3,4),(4,1))$$

$$v = v_1 \oplus v_2 = ((1,3),(3,2),(4,5)) \oplus ((3,4),(4,1))$$

$$v = ((1,3),(3,2),(4,5),(3,4),(4,1))$$

## 8. Multiplication "coefficient times velocity"

Misalkan  $c$  merupakan sebuah koefisien dan ( $c \in \mathbb{R}$ ), maka perkalian antara velocity  $v$  dan  $c$  dapat dilakukan operasi sebagai berikut sesuai dengan kondisi nilai koefisiennya:

a. Jika  $c = 0$

Maka,

$$v' = c \cdot v$$

$$\|v'\| = c \cdot \|v\|$$

$$\|v'\| = 0 \cdot \|v\|$$

$$\|v'\| = 0$$

$$v' = \emptyset$$

b. Jika  $0 < c \leq 1$

Maka,  $v$  dipotong

dengan panjang

$$\|v'\| = \lceil c \cdot \|v\| \rceil, \text{ contoh:}$$

$$v = ((1,3),(3,2),(4,5)),$$

$$\|v\| = 3 \quad c = 0.1$$

Penyelesaian:

$$v' = c \cdot v$$

$$\|v'\| = \lceil 0.1 * 3 \rceil = \lceil 0.3 \rceil = 1$$

$$v' = ((1,3))$$

c. Jika  $c > 1$

maka  $c$  dibentuk dari  $c = k + c'$  yang mana  $k = \lfloor c \rfloor$ ,  $k \in (\mathbb{N} > 0)$ ,  $c' = c - k$  dan  $0 < c' < 1$ . Sehingga kasus ini didefinisikan sebagai.

$$v' = \left( \sum_{1}^k v \right) \oplus (c' * v) = \underbrace{v \oplus v \oplus \dots \oplus v}_{k \text{ kali}} \oplus (c' * v)$$

$(c' \cdot v)$  dihitung menggunakan kondisi (b).

Contoh:

$$v = ((1,3),(3,2),(4,5)), \|v\| = 3 \quad c = 2.5$$

Penyelesaian:

$$k = \lfloor c \rfloor = \lfloor 2.5 \rfloor = 2 \text{ dan } c' = c - k = 2.5 - 2 = 0.5$$

$$\|v'\| = \|v\| + \|v\| + (\lceil c' * \|v\| \rceil) = 3 + 3 + \lceil 0.5 * 3 \rceil = 3 + 3 + \lceil 1.5 \rceil$$

$$\|v'\| = 3 + 3 + 2 = 8$$

$$v' = ((1,3),(3,2),(4,5),(1,3),(3,2),(4,5),(1,3),(3,2))$$

d. Jika  $c < 0$

maka velocity dibalik ( $v = -v$ ) menggunakan konsep pada slide ke-8 dan nilai  $c$  akan dipositifkan ( $c = |c|$ ). Sehingga velocity terbaru dapat dioperasikan menggunakan fungsi berikut.

$v' = c \cdot -v$ , dimana nilai  $c$  akan dioperasikan seperti pada kondisi (a), (b), dan (c) pada slide sebelumnya.

Contoh:

$$v = ((1,3),(3,2),(4,5)), \|v\| = 3 \quad c = -0.1$$

Penyelesaian:

$$v' = -c \cdot v = c \cdot -v$$

$$v' \| = [c' * \| -v \|] = [0.1 * 3] = [0.3] = 1$$

$$v' = ((4,5))$$

## 9. Distance between two positions

### 4.2.2 Rumus Update Kecepatan dan Posisi

Rumus update posisi yang telah dimodifikasi oleh Clerc didefinisikan sebagai berikut.

$$v_i^{t+1} = c_1 \cdot v_i^t \oplus c_2 \left( \left( Pbest_i^t + \frac{1}{2} (Gbest_g^t - Pbest_i^t) \right) - x_i^t \right)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Hoffmann et al. (2011) merubah representasi update posisi berdasarkan penemuan Clerc. Dengan menghilangkan koefisien inersia (set  $c_1 = 0$ ) dan menggantinya dengan kecepatan acak. Perubahan representasi tersebut dengan harapan bahwa partikel lebih baik dalam proses mendekati solusi optimal dan tidak cepat konvergen (konvergensi dini).

Rumus update posisi oleh Clerc (2000):

$$v_i^{t+1} = c_1 \cdot v_i^t \oplus c_2 \left( \left( Pbest_i^t + \frac{1}{2} (Gbest_g^t - Pbest_i^t) \right) - x_i^t \right)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Rumus update posisi oleh Hoffmann et al. (2011):

$$d_{loc} = x_i^t + r_{loc} \cdot b_{loc} (Pbest_i^t - x_i^t)$$

$$d_{glob} = x_i^t + r_{glob} \cdot b_{glob} (Gbest_g^t - x_i^t)$$

$$v_{rand} = r_{rand} \cdot b_{rand} (P_{rand}^t - x_i^t)$$

$$x_i^{t+1} = d_{glob} + \frac{1}{2} (d_{loc} - d_{glob}) + v_{rand}$$

Dimana  $r_{rand}, b_{loc}, b_{glob}, b_{rand} \in [0,1]$

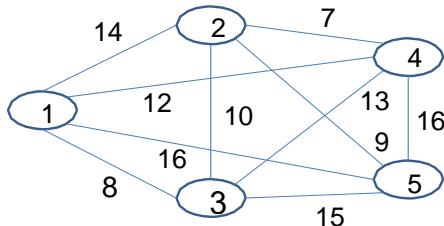
### 4.3 Travelling Salesman Problem (TSP)

Traveling salesman problem (TSP) merupakan suatu permasalahan sirkuit Hamilton (Hahsler, Michael dan Hornik, Kurt., 2007). Tujuan dari TSP adalah mencari suatu rute untuk mengunjungi semua kota yang ada tepat sebanyak satu kali dengan total jarak perjalanan seminimal mungkin dan kembali ke kota awal. Walaupun permasalahan ini mudah untuk dirumuskan, TSP sulit untuk dipecahkan, jika kasusnya komplek, mengingat TSP merupakan suatu permasalahan optimasi kombinatorial. **Nilai fitness** permasalahan TSP, yaitu minimalisasi jarak yang dapat didefinisikan sebagai berikut (Ahmed, Zakir H., 2010):

$$fitness = \frac{1}{f(x)}$$

Dimana:

$f(x)$  adalah fungsi obyektif dari TSP yaitu perhitungan total jarak tempuh untuk menyelesaikan satu rute perjalanan.



Gambar 4.1 Contoh Permasalahan TSP

Tabel 4.1 Tabel Jarak Antar Simpul

Node	1	2	3	4	5
1	-	14	8	12	16
2	14	-	10	7	9
3	8	10	-	13	15
4	12	7	13	-	16
5	16	9	15	16	-

Penyelesaian dengan algoritma Hybrid Discrete Particle Swarm Optimization (HDPSO) untuk masalah kombinatorial (Clerc, 2000)

### 4.3.1 Inisialisasi Kecepatan awal Partikel

Pada iterasi ke-0 ( $t=0$ ), nilai kecepatan awal semua partikel di ( $v_i(t) = \emptyset$ ) dan misal sebagai berikut ( $popSize=3$ ):

Kecepatan	
$v_1(0)$	$\emptyset$
$v_2(0)$	$\emptyset$
$v_3(0)$	$\emptyset$

### 4.3.2 Inisialisasi Posisi awal Partikel

Partikel inisial, pada iterasi ke-0 ( $t=0$ ) dibangkitkan secara random berupa angka integer yang menyatakan nomer simpul dan kombinasinya unik, dan misal sebagai berikut:

$x_i(t)$	partikel	Total Jarak ( $f(x)$ )	$fitness = \frac{1}{f(x)}$
$x_1(0)$	[ 1 2 3 4 5 ]	$14+10+13+16+16=69$	$1,449 \times 10^{-2}$
$x_2(0)$	[ 2 3 4 1 5 ]	$10+13+12+16+9=60$	$1,667 \times 10^{-2}$
$x_3(0)$	[ 4 1 2 5 3 ]	$12+14+9+15+13=63$	$1,587 \times 10^{-2}$

Misal, partikel [ 2 3 4 1 5 ] menyatakan perjalanan dimulai dari simpul 2 kemudian mengunjungi simpul 3, 4, 1, 5 dan kemudian kembali ke simpul 2.

### 4.3.3 Inisialisasi Pbest dan Gbest

**Pbest**, karena masih iterasi ke-0 ( $t=0$ ), maka nilai  $Pbest$  akan disamakan dengan nilai posisi awal partikel, yaitu ( $Pbest_i(t)=x_i(t)$ ) dan misal  $Pbest$ -nya sebagai berikut:

$Pbest_i(t)$	<b>Pbest</b>	Total Jarak ( $f(x)$ )	$fitness = \frac{1}{f(x)}$
$Pbest_1(0)$	[ 1 2 3 4 5 ]	$14+10+13+16+16=69$	$1,449 \times 10^{-2}$
$Pbest_2(0)$	[ 2 3 4 1 5 ]	$10+13+12+16+9=60$	$1,667 \times 10^{-2}$
$Pbest_3(0)$	[ 4 1 2 5 3 ]	$12+14+9+15+13=63$	$1,587 \times 10^{-2}$

**Gbest**, memilih satu  $Pbest$  yang *fitness*-nya tertinggi, ( $k=\arg\max_i \{ fitness_{Pbest_i(t)} \} = 2$ ), maka  $Gbest_{g=1}(t=0) = Pbest_{k=2}(t=0)$ .

$Gbest_i(t)$	<b>Gbest</b>
$Gbest_1(0)$	[ 2 3 4 1 5 ]

Masuk pada iterasi ke-1, ( $t = t + 1 = 0 + 1 = 1$ )

### 4.3.4 Update Kecepatan

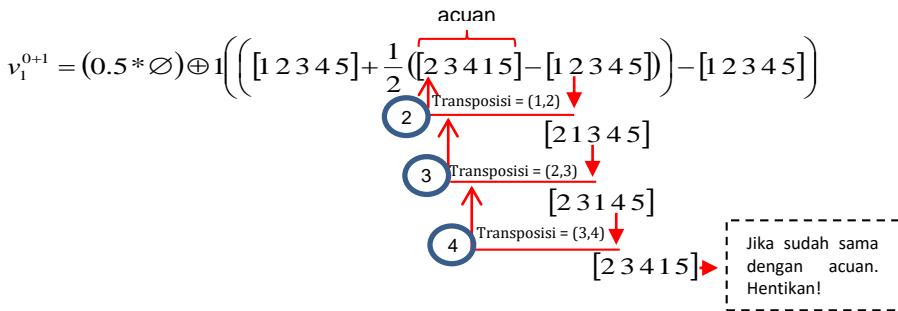
Misal menggunakan rumus update kecepatan berdasarkan Clerc (2000):

$$v_i^{t+1} = c_1 \cdot v_i^t \oplus c_2 \left( \left( Pbest_i^t + \frac{1}{2} (Gbest_g^t - Pbest_i^t) \right) - x_i^t \right)$$

Jika diketahui  $c_1 = 0.5$ ,  $c_2 = 1$ . Maka untuk mendapatkan hasil update kecepatannya dihitung sebagai berikut, misal menghitung  $v_1$  (1).

Kecepatan	$x_i(t)$	partikel	$Pbest_i(t)$	<b>Pbest</b>	$Gbest_i(t)$	<b>Gbest</b>
$v_1(0)$	$\emptyset$				$Gbest_1(0)$	[ 2 3 4 1 5 ]
	$x_1(0)$	[ 1 2 3 4 5 ]	$Pbest_1(0)$	[ 1 2 3 4 5 ]		

$$v_1^{0+1} = (0.5 * \emptyset) \oplus 1 \left( \left[ [1 2 3 4 5] + \frac{1}{2} ([2 3 4 1 5] - [1 2 3 4 5]) \right] - [1 2 3 4 5] \right)$$



$$v_1^{0+1} = \emptyset \oplus 1 \left( \left( [1 2 3 4 5] + \frac{1}{2} ((1,2), (2,3), (3,4)) \right) - [1 2 3 4 5] \right)$$

$$v_1^{0+1} = \left( \left( [1 2 3 4 5] + \frac{1}{2} ((1,2), (2,3), (3,4)) \right) - [1 2 3 4 5] \right)$$

$$v = ((1,2), (2,3), (3,4)), \quad \|v\| = 3 \quad c = 0.5$$

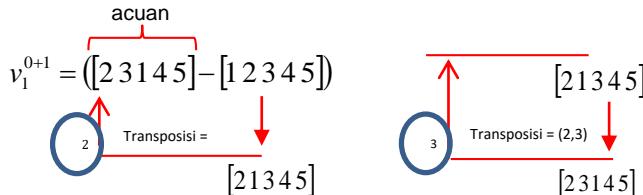
$$v' = c \cdot v$$

$$\|v'\| = [0.5 * 3] = [1.5] = 2$$

$$v' = ((1,2), (2,3))$$

$$v_1^{0+1} = (([1 2 3 4 5] + ((1,2), (2,3))) - [1 2 3 4 5])$$

$$v_1^{0+1} = (([2 1 3 4 5] + ((2,3))) - [1 2 3 4 5])$$



$$v_1^{0+1} = ((1,2), (2,3))$$



Kecepatan	
$v_1(1)$	((1,2), (2,3))
$v_2(1)$	.....
$v_3(1)$	.....

### 4.3.5 Update Posisi

Rumus update posisi berdasarkan Clerc (2000):

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Update posisi dihitung sebagai berikut, misal hitung  $x_{1,1}(1)$ .

$x_i(t)$	partikel	Kecepatan	
		$v_{1,1}(1)$	$((1,2),(2,3))$
$x_1(0)$	[ 1 2 3 4 5 ]		

Node	1	2	3	4	5
1	-	14	8	12	16
2	14	-	10	7	9
3	8	10	-	13	15
4	12	7	13	-	16
5	16	9	15	16	-

$$x_1^{0+1} = x_1^0 + v_1^{0+1}$$

$$x_1^1 = [1\ 2\ 3\ 4\ 5] + ((1,2), (2,3)) = [2\ 1\ 3\ 4\ 5] + ((2,3)) = [2\ 3\ 1\ 4\ 5]$$

Berikut keseluruhan hasil update posisinya:

$x_i(t)$	partikel	Total Jarak ( $f(x)$ )	$fitness = \frac{1}{f(x)}$
$x_1(1)$	[ 2 3 1 4 5 ]	$10+8+12+16+9=55$	$1,818 \times 10^{-2}$
$x_2(1)$	....	....	....
$x_3(1)$	....	....	....

### 4.3.6 Update Pbest dan Gbest

Update Pbest, disini kita harus membandingkan antara Pbest pada iterasi sebelumnya dengan hasil dari Update Posisi.

$Pbest_i(t)$	Pbest	Total Jarak ( $f(x)$ )	$fitness = \frac{1}{f(x)}$
$Pbest_1(0)$	[ 1 2 3 4 5 ]	$14+10+13+16+16=69$	$1,449 \times 10^{-2}$
$Pbest_2(0)$	[ 2 3 4 1 5 ]	$10+13+12+16+9=60$	$1,667 \times 10^{-2}$
$Pbest_3(0)$	[ 4 1 2 5 3 ]	$12+14+9+15+13=63$	$1,587 \times 10^{-2}$

$x_i(t)$	partikel	Total Jarak ( $f(x)$ )	$fitness = \frac{1}{f(x)}$
$x_1(1)$	[ 2 3 1 4 5 ]	$10+8+12+16+9=69$	$1,818 \times 10^{-2}$
$x_2(1)$	....	....	....
$x_3(1)$	....	....	....

Dari 2 tabel di atas, cek dari urutan baris yang sama, kemudian dibandingkan *fitness*-nya, manakah yang lebih tinggi nilainya akan menjadi *Pbest* terbaru. Dan *Pbest* terbaru dengan nilai *fitness* tertinggi akan menjadi *Gbest*.

Kemudian, jika dilanjutkan iterasi berikutnya ( $t = t + 1$ ), maka langkah di bawah ini akan diulang terus-menerus sampai iterasi Maksimum atau telah mencapai konvergen.

1. **Update Kecepatan**
2. **Update Posisi dan Hitung Fitness**
3. **Update Pbest dan Gbest**

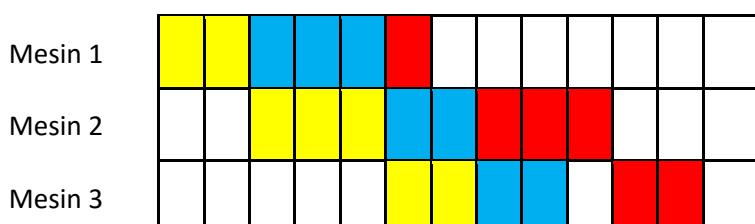
## 4.4 Scheduling Problem

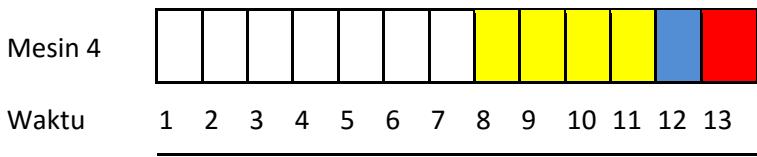
### 4.4.1 Flow-Shop Scheduling Problem (FSP)

**Flow-Shop Scheduling Problem (FSP)** : berkaitan dengan penjadwalan sejumlah  $j$  job pada sejumlah  $m$  mesin. Urutan operasi job pada mesin selalu sama, yaitu dari mesin 1, 2, 3, 4. dan semua job memerlukan semua mesin.

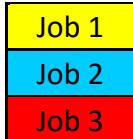
Job	Mesin			
	1	2	3	4
1	2	3	2	4
2	3	2	2	1
3	1	3	2	1

Jika urutan job  $J_1 \rightarrow J_2 \rightarrow J_3$  maka didapatkan *makespan* sebesar 13 pada Gantt-Chart berikut:





Keterangan:



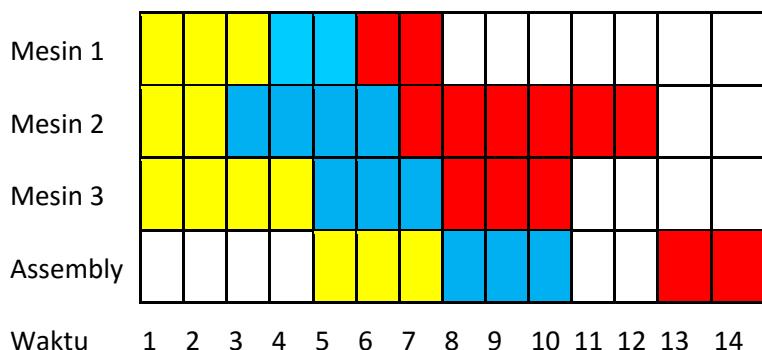
Representasi FSP seperti pada TSP. Setiap dimensi pada partikel menyatakan nomer job. Contoh: J1 → J2 → J3 menjadi  $X_1(0)=[1 \ 2 \ 3]$

#### 4.4.2 Two-Stage Assembly Flow-Shop Scheduling Problem

Penjadwalan 3 job pada 3 mesin, dan assembly (operasi tahap kedua).

job	mesin			assembly
	1	2	3	
1	3	2	4	3
2	2	4	3	3
3	2	6	3	2

Jika urutannya  $J1 \rightarrow J2 \rightarrow J3$  maka didapatkan *makespan* sebesar 14 pada Gantt-Chart berikut:

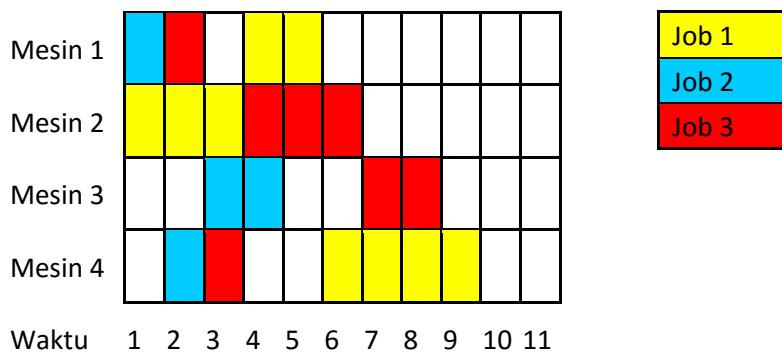


### 4.4.3 Job-Shop Scheduling Problem (JSP)

**Job-Shop (JSP):** Urutan operasi job pada mesin bisa berbeda-beda, dan sebagain job bisa hanya perlu beberapa mesin

Job	Waktu Operasi Pada Mesin				Urutan Operasi
	1	2	3	4	
1	2	3	-	4	$2 \rightarrow 1 \rightarrow 4$
2	1	-	2	1	$1 \rightarrow 4 \rightarrow 3$
3	1	3	2	1	$1 \rightarrow 4 \rightarrow 2 \rightarrow 3$

$O_{i,j}$  menyatakan **operasi ke- $j$  dari job  $i$** . Jika urutan job  $O_{1,1} \rightarrow O_{2,1} \rightarrow O_{1,2} \rightarrow O_{3,1} \rightarrow O_{3,2} \rightarrow O_{1,3} \rightarrow O_{2,2} \rightarrow O_{3,3} \rightarrow O_{3,4} \rightarrow O_{2,3}$  maka didapatkan makespan = 9



untuk solusi di atas bisa dinyatakan sebagai:

Job : [ 1 2 1 3 3 1 2 3 3 2 ]

Op : [ 1 1 2 1 2 3 2 3 4 3 ]

Angka permutasi

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Job

1	1	1	2	2	2	3	3	3	3
---	---	---	---	---	---	---	---	---	---

Representasi permutasi untuk JSP:  $O_{1,1} \rightarrow O_{2,1} \rightarrow O_{1,2} \rightarrow O_{3,1} \rightarrow O_{3,2} \rightarrow O_{1,3} \rightarrow O_{2,2} \rightarrow O_{3,3} \rightarrow O_{3,4} \rightarrow O_{2,3}$  maka menjadi

$X_1(0)=[1 4 2 7 8 3 5 9 10 6]$

#### 4.4.4 Flexible Job-Shop Scheduling Problem (FJSP)

**Flexible Job-Shop (FJSP):** merupakan bentuk umum (*generalized form*) dari JSP klasik. Sebuah job memiliki beberapa operasi. Setiap operasi bisa dikerjakan pada beberapa pilihan mesin.

job	operasi	machine	time
1	1	1	5
		2	6
	2	3	4
2	1	2	7
		3	8
	2	1	6
3	1	3	4
		2	3
	3	1	4
4	1	2	4
		3	5
	2	1	4
3	1	1	6
		2	4

Angka permutasi

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Job

1	1	2	2	2	3	3	4	4	4
---	---	---	---	---	---	---	---	---	---

Pilihan Mac

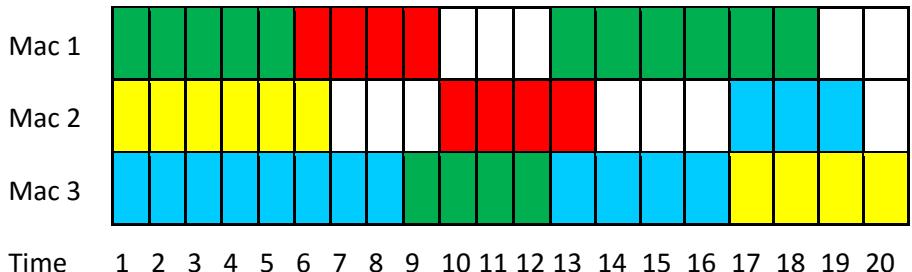
2	1	2	2	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---

Representasi permutasi untuk FJSP, dapat dibentuk dengan menggunakan gabungan dua cluster code partikel, yaitu kombinatorial code dan real code. Contoh:

$$X_1(0) = [5 \ 9 \ 8 \ 3 \ 1 \ 10 \ 4 \ 6 \ 7 \ 2] [2 \ 1 \ 1 \ 2 \ 2 \ 1 \ 1 \ 1 \ 1 \ 1]$$



Solusi  $X_1(0)$  bisa menghasilkan *makespan* 20 dengan Gantt-chart sebagai berikut:



Keterangan:



x	operasi	sorted x	operasi'	job,op,mac
143	1	75	5	2, 1, 3
209	2	83	9	4, 1, 1
115	3	96	8	4, 2, 3
173	4	115	3	2, 2, 3
75	5	143	1	1, 1, 2
179	6	144	10	4, 3, 1
193	7	173	4	2, 3, 2
96	8	179	6	3, 1, 1

83	9	193	7	3, 2, 2
144	10	209	2	1, 2, 3

---

## 4.5 Tugas Kelompok

1. Jelaskan konsep optimasi masalah kombinatorial pada algoritma DPSO, kenapa tidak cukup menggunakan PSO standar!
2. Jelaskan perbedaan dari representasi inisialisasi posisi partikel, inisialisasi kecepatan, update kecepatan, update posisi dari binary code vs real code vs dan kombinatorial code?
3. Jika diketahui  $x_1(t) = [1 \ 5 \ 2 \ 7 \ 3 \ 9 \ 4 \ 6 \ 8]$  dan  $Pbest_1(t) = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9]$ . Tentukan:
  - a.  $x_1(t) + ((2,5),(9,7))$
  - b.  $Pbest_1(t) - x_1(t)$
  - c.  $c * (Pbest_1(t) - x_1(t))$ , misal  $c = 3.5$
  - d.  $c * (Pbest_1(t) - x_1(t))$ , misal  $c = -1.5$
4. Pada sub bab 4.3.4, telah dihitung hasil dari  $v_1(1)$ , dengan cara yang sama. Tentukan hasil dari kecepatan partikel ke-2  $v_2(1)$  !

Kecepatan	
$v_1(1)$	$((1,2),(2,3))$
$v_2(1)$	.....
$v_3(1)$	.....

---

5. Perhatikan pada sub bab 4.3.5, misal menggunakan rumus update posisi berdasarkan usulan dari Hoffmann et al. (2011):

$$\begin{aligned}d_{loc} &= x_i^t + r_{loc} \cdot b_{loc} (Pbest_i^t - x_i^t) \\d_{glob} &= x_i^t + r_{glob} \cdot b_{glob} (Gbest_g^t - x_i^t) \\v_{rand} &= r_{rand} \cdot b_{rand} (P_{rand}^t - x_i^t)\end{aligned}$$

$$x_i^{t+1} = d_{glob} + \frac{1}{2}(d_{loc} - d_{glob}) + v_{rand}$$

Jika diketahui  $r_{loc} = 0.75$  ;  $r_{glob} = 0.5$  ;  $b_{loc} = b_{glob} = b_{grand} = 0.8$  ;  $r_{rand} = 0.1$  dan  $P_{rand}(0)=[5\ 4\ 1\ 3\ 2]$ . Tentukan hasil  $x_1(1)$ !

6. Berdasarkan sub bab 4.4.3 pada kasus JSP, jika diketahui  $X_1(0)=[1\ 9\ 8\ 5\ 2\ 3\ 7\ 4\ 10\ 6]$ . Hitunglah *makespan*-nya!

## BAB 5 Artificial Bee Colony (ABC)

### 5.1 Pengantar

**Artificial Bee Colony** (ABC) pada dasarnya memiliki model yang simple, jelas dan bisa diadaptasi (Mustafa Servet Kiran, 2012). Ada 2 jenis *foragers* (penjelajah) dalam sarang lebah (hive) ABC, yaitu *employeed foragers* dan *unemployeed foragers*. *Employeed foragers* mengeksplorasi mulai dari sumber makanan (berhubungan dengan solusi potensial dari masalah optimasi) secara terus menerus, dan lebah-lebah tersebut membawa informasi tentang posisi sumber makanan ke sarang lebah. Ada 2 jenis *unemployeed foragers*, yaitu onlooker bee dan scout bee.

**Onlooker** bee pergi ke sumber makanan agar bisa mengeksplorasi dengan cara mempertimbangkan informasi dari penjelajah pekerja (*employeed foragers*). Scout bee mencari sumber makanan sekitar sarang lebah. Dari populasi, rata-rata banyaknya scout bee adalah sekitar 5-10 persen. Persentase di dalam alam, walaupun ada 5-10 % scout bee dari ukuran populasi, tetapi hanya ada satu scout bee di dalam sarang ABC. Setengah dari ukuran populasi adalah *employeed bee*, dan setengah lainnya adalah onlooker di dalam ABC dasar.

### 5.2 Real Code Artificial Bee Colony (ABC)

#### 1. Inisialisasi:

- Buat solusi random tiap **employed bee** sebanyak *popSize* dan hitung fitness

$$x_i^j = x_{\min}^j + \text{rand}[0,1] \times (x_{\max}^j - x_{\min}^j)$$

Bee ke-*i* pada dimensi ke-*j*

Nilai batas bawah pada dimensi ke-*j*

Nilai batas atas pada dimensi ke-*j*

- Set nilai trial = 0 untuk tiap **employed bee**

- c. Tentukan global best berdasarkan fitness tertinggi dari (a)
2. While (bukan kondisi berhenti)

#### Fase Employed Bee

- a. Pilih neighbor **employed bee** secara random ( $k$ ),  $k \neq i$ , dimana  $k, i \in \{1, 2, \dots, popSize\}$
- b. Update posisi dengan persamaan berikut

$$v_{i,j} = x_{i,j} + rand[-1,1] \times (x_{i,j} - x_{k,j})$$

Jika nilai *fitness* yang baru memiliki nilai yang lebih tinggi, maka *trial* pada masing-masing individu direset lagi dengan nilai 0. Jika tidak, maka *trial* ditambahkan dengan 1.

#### Fase Onlooker Bee

- a. Pilih neighbor employed bee secara random ( $k$ ),  $k \neq i$ , dimana  $k, i \in \{1, 2, \dots, popSize\}$
- b. Seleksi employed bee dengan RW untuk dilakukan improvement dengan persamaan berikut.

$$Prob_i = \frac{fitness(x_i)}{\sum_{k=1}^{popSize} fitness(x_k)}$$

- c. Lalu, hasil seleksinya improve dengan persamaan berikut dan hitung fitness, dimana setelah update,  $x_i = v_i$ .

$$v_{i,j} = x_{i,j} + rand[-1,1] \times (x_{i,j} - x_{k,j})$$

Jika nilai *fitness* yang baru memiliki nilai yang lebih tinggi, maka *trial* pada masing-masing individu direset lagi dengan nilai 0. Jika tidak, maka *trial* ditambahkan dengan 1.

#### Fase Scout Bee

Simpan global best, lalu

hitung M, dimana  $M=\text{Max}(trial)$  dari keseluruhan bee.

Aturan: Jika  $M > \text{Limit}$ , maka hapus individu yang tidak ada perbaikan, ganti dengan individu baru hasil random, dengan persamaan,

$$x_i^j = x_{\min}^j + rand[0,1] \times (x_{\max}^j - x_{\min}^j)$$

kemudian reset trial = 0, jika  $M > \text{Limit}$  dan terdapat individu yang ada perbaikan, maka tidak perlu digantikan dengan individu baru. kemudian reset trial = 0. Jika  $M < \text{Limit}$ , maka tidak perlu digantikan dengan individu baru, dan nilai trial tidak perlu di-reset.

8. Selesai

## 5.3 Discrete ABC

Algoritma *Artificial Bee Colony* (ABC) adalah pendekatan population-based metaheuristic yang diusulkan oleh Karaboga dan Basturk (Karaboga, 2007). Pendekatan ini terinspirasi oleh perilaku cerdas kawanan lebah madu mencari makanan. Algoritma ABC memiliki 3 kelompok lebah:

- **Employeed Bee (Lebah Pekerja)** adalah lebah yang berhubungan dengan sumber makanan tertentu,
- **Onlooker Bee (Lebah Penjaga)** menyaksikan tarian lebah yang digunakan dalam sarang untuk memilih sumber makanan
- **Scout Bee (Lebah Pengintai)** mencari sumber makanan secara acak. Awalnya Scout Bee menemukan posisi semua sumber makanan, setelah itu tugas dari Employeed Bee dimulai.

Sebuah Employeed Bee secara probabilitas mengalami beberapa modifikasi posisi di memori untuk target sumber makanan baru dan menemukan jumlah nektar (madu bunga) atau nilai fitness dari sumber baru. Kemudian Scout Bee mengevaluasi informasi dari semua *Employeed Bee* buatan dan memilih sumber makanan akhir dengan nilai probabilitas tertinggi terkait dengan jumlah nektar tersebut. Jika fitness baru lebih tinggi dari sebelumnya, lebah akan melupakan yang lama dan menghafal posisi baru atau disebut *greedy selection*. Kemudian *Employeed Bee* yang sumber makanannya telah habis menjadi akan menjadi Scout Bee untuk mencari sumber makanan lebih lanjut sekali lagi. Berikut langkah-langkah Artificial Bee Colony (ABC):

1. **Inisialisasi** semua **parameter** yang diperlukan, yaitu data kasus, colony size, limit dan maksiterasi.
2. Fase initial:
  - Untuk tiap *population size*, *initial solution* dihasilkan secara *random*.
  - Hitung fitness
3. Iterasi = 1
4. Fase **Employeed Bee**:
  - Untuk tiap *Employeed Bee*, *Update Employeed Bee* dengan menggunakan *neighborhood operator*, yang terdiri dari *swap operator* dan *swap sequence*.

- Hitung fitness
  - Jika solusi yang baru hasilnya lebih baik dari sebelumnya gantikan solusi lama dengan solusi baru, jika tidak tambahkan *trial* dengan 1.
5. Hitung probabilitas tiap Employee Bee.
  6. Fase **Onlooker Bee**:
    - Untuk tiap *Onlooker Bee*, pilih solusi dari *Employee Bee* dengan nilai probabilitas dan gunakan teknik *roulette wheel selection*.
    - Tentukan solusi yang baru *Employee Bee* terpilih dengan menggunakan *neighborhood operator*, yang terdiri dari *insert operator* dan *insert sequence*.
    - Hitung fitness
    - Jika solusi yang baru hasilnya lebih baik dari sebelumnya gantikan solusi lama dengan solusi baru, jika tidak tambahkan *trial* dengan 1.
  7. Fase **Scout Bee**:
    - Hitung jumlah trial dan simpan jumlah maksimal dalam variabel M, untuk tiap bee yang tidak mengalami peningkatan solusi.
    - Jika  $M > \text{limit}$ , tinggalkan solusi yang tidak mengalami peningkatan, jika  $M < \text{limit}$ , scout bee memilih solusi sebelumnya.
  8. Tandai solusi terbaik yang dicapai saat ini.
  9. Iterasi = Iterasi + 1
  10. Stop ketika Iterasi sudah melebihi maks iterasi.

### Detail Artificial Bee Colony

1. Input Parameter:
  - Colony size, merupakan jumlah dari *Employee Bee* ditambah *Onlooker Bee* yang akan digunakan dalam sistem. Dimana jumlah *Employee Bee* = *Onlooker Bee* dalam hal ini disebut *Population Size*.
  - Limit, merupakan batasan dari population size yang tidak mengalami peningkatan kualitas untuk sejumlah iterasi.
  - Maksiterasi, merupakan banyaknya iterasi yang akan dilakukan atau dapat disebut juga sebagai kriteria berhenti.
2. Fase Initial:
  - Fase initial merupakan proses untuk mendapatkan *initial solution* untuk tiap *Employee Bee*
  - Menghitung fitness (untuk  $f(x_i) \geq 0$  dan untuk  $f(x_i) < 0$ )

$$fitness_i = \frac{1}{1 + f(x_i)} \quad fitness_i = 1 + |f(x_i)|$$

### 3. Improvement Solution

- *Improvement solution* berada pada fase *Employeed Bee* dan *Onlooker Bee*. *Improvement solution* digunakan untuk memperbarui solusi dengan metode *neighborhood operator*.
- *Neighborhood operator* yang akan digunakan
  - *Swap Operator* dan *Swap Sequence* pada fase *Employeed Bee*
  - *Insertion Operator* dan *Insertion Sequence* pada fase *Onlooker Bee*.
- *Neighborhood operator* digunakan untuk mencari kandidat solusi, dimana *neighborhood operator* terdiri dari 2 kelompok, yaitu operator poin ke poin dan operator subsekuen. Operator poin ke poin adalah *random swap* dan *random insertion*.
- Sementara, operator subsekuen adalah *random swap sequences*, *random insertion sequences*, *random reversing of sequences* (RRS), *random reversing swap of sequences* (RRSS), dan *random reversing insertion of sequences* (RRIS).

#### Neighbourhood Operator

**Random Reversing of Subsequence (RRS)**, operator ini membalikkan subsekuen yang terpilih secara acak dari sekuen.

2	8	7	5	10
2	5	7	8	10

**Random Reversing Insertion of Subsequences (RRIS)**, Operator ini terdiri dari random insertion sequences dan random insertion. Subsekuen yang terpilih secara acak dari sekuen, ditambahkan kepada titik yang dipilih secara acak. Sisa sekuen dirubah ke kanan sampai ukuran subsekuen. Sebelum menambahkan, subsekuen dibalikkan dengan kemungkinan 50%.

8	10	7	5	2
2	5	7	8	10

**Random Reversing Swap of Subsequences (RRSS)**, Operator ini adalah kombinasi dari dua operator, yaitu random swap dan random swap sequences. Dua Subsekuen pertama diseleksi untuk pertukaran.

Masing-masing dibalikkan dengan kemungkinan 50%. Tidak hanya bagian 1 dan 2 yang ditukar tetapi juga bagian 1 dibalikkan.

10	8	7	2	5
2	5	7	8	10

### Detail ABC

#### 1. Fase *Employeed Bee*

**Swap Operator (SO)** atau **Random Swap**. Misal bilangan randomnya adalah SO(1,3). Maka solusi baru yang dihasilkan dari initial solution dengan SO adalah sebagai berikut:

$$x_i = x_i + SO$$

$$x_i = (2 - 5 - 7 - 8 - 10) + SO(1,3)$$

2	5	7	8	10
7	5	2	8	10

#### Swap Sequences(SS) atau Random Swap Sequences

$$x_i = x_i + SS$$

$$= x_i + (SO_1, SO_2, SO_3, \dots, SO_n)$$

$$= (((((x_i + SO_1) + SO_2) + SO_3) + \dots + SO_n))$$

n = Banyaknya Swap Sequence, lalu hitung nilai probabilitas

$$Prob_i = \frac{fitness(x_i)}{\sum_{k=1}^S fitness(x_k)}$$

dimana

$Prob_i$  = Peluang memilih *Employeed Bee* ke-*i*

S = Jumlah *Employeed Bee*

#### 2. Fase *Onlooker Bee*

**Insert Operator (IO)** atau **Random Insertion**. Sebagai contoh bilangan random yang dihasilkan adalah IO(1,3). Maka solusi baru yang dihasilkan dari *initial tour* dengan IO adalah sebagai berikut:

$$x_i = x_i + IO$$

$$x_i = (2 - 5 - 7 - 8 - 10) + IO(1,3)$$

2	5	7	8	10
7	2	5	8	10

### **Insert Sequences (IS) atau Random Insertion Sequences**

$$\begin{aligned}x_i &= x_i + IS \\&= x_i + (IO_1, IO_2, IO_3, \dots, IO_n) \\&= (((((x_i + IO_1) + IO_2) + IO_3) + \dots) + IO_n)\end{aligned}$$

n = Banyaknya Insert Sequence

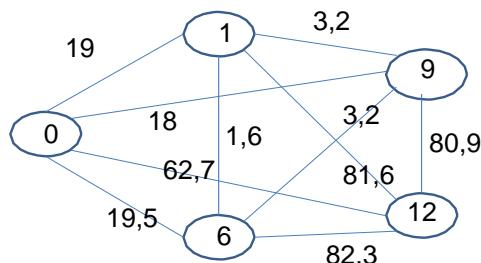
### **3. Fase Scout Bee**

Setelah melalui dua fase improvement solution, fase *Employeed Bee* dan fase *Onlooker Bee*, maka akan dilakukan perhitungan kualitas dari masing-masing *Employeed Bee*. Jumlah *Scout Bee* disini bersifat dinamis, tergantung pada jumlah *Employeed Bee* yang telah melebihi limit. Apabila limit dari bee yang melakukan improvement solution melebihi maximum limit yang ditetapkan, maka solusi dari bee tersebut akan dihilangkan dan diganti dengan solusi baru dengan menggunakan teknik random, memperbarui jarak yang dihasilkan, dan menyetel ulang limit kembali menjadi 0.

### **4. Kriteria Berhenti**

Kriteria berhenti dari sistem ini akan dilakukan sebanyak maksiterasi yang didefinisikan. Iterasi akan dilakukan sampai kriteria berhenti terpenuhi, dan selama belum terpenuhi, maka akan mengulang langkah ke-3.

## **5.4 Case Study: Travelling Salesman Problem (TSP)**



Gambar 5.1 Contoh Permasalahan TSP

Jarak	0	1	6	9	12
0	0	19	19,5	18	62,7
1	19	0	1,6	3,2	81,6
6	19,5	1,6	0	3,2	82,3
9	18	3,2	3,2	0	80,9
12	62,7	81,6	82,3	80,9	0

Langkah penyelesaian dengan algoritma Artificial Bee Colony (ABC) adalah sebagai berikut:

#### 5.4.1 Inisialisasi Parameter

*Colony Size = 4, Maksiter = 2, Size Problem = 5, Limit = 5, Colony Size = Population Size, Number of sequence (Nse) = 2\*Size Problem = 2\*5 =10*

#### 5.4.2 Fase Initial

Membangkitkan secara random untuk mendapatkan *initial solution* untuk tiap *Employeed Bee*, misal sebagai berikut:

Bee Ke -i	Rute					Jarak					Total Jarak	Fitness
x <sub>1</sub>	0	6	12	9	1	19,5	82,3	80,9	3,2	19	204,9	<b>0,00486</b>
x <sub>2</sub>	6	1	12	0	9	1,6	81,6	62,7	18	3,2	167,1	<b>0,00595</b>
x <sub>3</sub>	9	12	6	1	0	80,9	82,3	1,6	19	18	201,8	<b>0,00493</b>
x <sub>4</sub>	1	6	0	9	12	1,6	19,5	18	80,9	81,6	201,6	<b>0,00494</b>

$$fitness_i = \frac{1}{1 + f(x_i)}$$

Initial solution tiap employee bee

Bee Ke -i	Rute						Fitness
1	0	6	12	9	1		<b>0,00486</b>
2	6	1	12	0	9		<b>0,00595</b>
3	9	12	6	1	0		<b>0,00493</b>
4	1	6	0	9	12		<b>0,00494</b>

Masuk pada iterasi ke-1

### 5.4.3 Improvement Solution (Fase Employeed Bee)

- Perhitungan *swap operator*

$$fitness_i = \frac{1}{1 + f(x_i)}$$

Bee Ke - i	Rute					(SO)	Bee Ke - i	Rute					Jarak					Total Jarak	Fitness
x <sub>1</sub>	0	6	12	9	1	(1,3)	x <sub>1</sub>	12	6	0	9	1	82,3	19,5	18	3,2	81,6	204,6	0,00486
x <sub>2</sub>	6	1	12	0	9	(2,3)	x <sub>2</sub>	6	12	1	0	9	82,3	81,6	19	18	3,2	204,1	0,00488
x <sub>3</sub>	9	12	6	1	0	(1,4)	x <sub>3</sub>	1	12	6	9	0	81,6	82,3	3,2	18	19	204,1	0,00488
x <sub>4</sub>	1	6	0	9	12	(3,5)	x <sub>4</sub>	1	6	12	9	0	1,6	82,3	80,9	18	19	201,8	0,00493

→ SO dibangkitkan secara random

Perbandingan nilai *Fitness* (initial solution dan hasil swap operator)

Bee Ke - i	Awal			Hasil (SO)			Trial
	Total Jarak		Fitness	Total Jarak		Fitness	
x <sub>1</sub>	204,9		0,004857	204,6		0,004864	0
x <sub>2</sub>	167,1		0,005949	204,1		0,004876	1
x <sub>3</sub>	201,8		0,004931	204,1		0,004876	1
x <sub>4</sub>	201,6		0,004936	201,8		0,004931	1

Jika nilai *fitness* yang baru memiliki nilai yang lebih tinggi, maka *trial* pada masing-masing individu diberi nilai 0. Jika tidak, maka *trial* ditambahkan dengan 1

Individu yang dipilih

Bee Ke - i	Rute				
x <sub>1</sub>	12	6	0	9	1
x <sub>2</sub>	6	1	12	0	9
x <sub>3</sub>	9	12	6	1	0
x <sub>4</sub>	1	6	0	9	12

Individu yang dipilih merupakan hasil dari perbandingan nilai *fitness* individu awal dengan individu baru dengan nilai *fitness* yang lebih besar.

## 2. Perhitungan swap sequence

### Perhitungan swap sequences bee ke-1

Bee Ke- 1	12	6	0	9	1	Fitness	0,004864	Trial
(SO)	Rute					Jarak	Fitness	
(1,2)	6	12	0	9	1	167,8	0,005924	0
(...,)	..	..	..	..	..	170,4	0,005834	1
(...,)	..	..	..	..	..	167,9	0,005921	2
(...,)	1	12	0	9	6	167,1	0,005949	0
(...,)	..	..	..	..	..	167,4	0,005938	1
(...,)	..	..	..	..	..	167,8	0,005924	2
(...,)	..	..	..	..	..	201,6	0,004936	3
(...,)	..	..	..	..	..	201,6	0,004936	4
(...,)	..	..	..	..	..	204,1	0,004876	5
(...,)	1	12	0	6	9	170,2	0,005841	6
Terbaik	1	12	0	9	6	Fitness	0,005949	

Keterangan:

SO dibangkitkan secara random

### Perhitungan swap sequences bee ke-1

Bee Ke- 1	12	6	0	9	1	Fitness	0,004864	Trial
(SO)	Rute					Jarak	Fitness	
(1,2)	6	12	0	9	1	167,8	0,005924	0
(...,)	..	..	..	..	..	170,4	0,005834	1
(...,)	..	..	..	..	..	167,9	0,005921	2
(...,)	1	12	0	9	6	167,1	0,005949	0
(...,)	..	..	..	..	..	167,4	0,005938	1
(...,)	..	..	..	..	..	167,8	0,005924	2
(...,)	..	..	..	..	..	201,6	0,004936	3
(...,)	..	..	..	..	..	201,6	0,004936	4
(...,)	..	..	..	..	..	204,1	0,004876	5
(...,)	1	12	0	6	9	170,2	0,005841	6
Terbaik	1	12	0	9	6	Fitness	0,005949	

### Perhitungan swap sequences bee ke-3

Bee Ke- 3	9	12	6	1	0	Fitness	0,004931	Trial
(SO)	Rute					Jarak	Fitness	
(1,2)	12	9	6	1	0	167,4	0,005938	1
(...,)	..	..	..	..	..	204,2	0,004873	2
(...,)	..	..	..	..	..	170,4	0,005834	3
(...,)	0	9	6	1	12	167,1	0,005949	0
(...,)	..	..	..	..	..	170,2	0,005841	1
(...,)	..	..	..	..	..	167,4	0,005938	2
(...,)	..	..	..	..	..	167,8	0,005924	3
(...,)	..	..	..	..	..	167,8	0,005924	4
(...,)	..	..	..	..	..	201,6	0,004936	5
(...,)	0	9	6	12	1	204,1	0,004876	6
Terbaik	0	9	6	1	12	Fitness	0,005949	

### Perhitungan swap sequences bee ke-4

Bee Ke-4	1	6	0	9	12	Fitness	0,004936	Trial
(SO)	Rute					Jarak	Fitness	
(1,2)	6	1	0	9	12	201,8	0,004931	2
(...,)	..	..	..	..	..	167,9	0,005921	0
(...,)	..	..	..	..	..	167,4	0,005938	0
(...,)	..	..	..	..	..	201,8	0,004931	1
(...,)	..	..	..	..	..	167,9	0,005921	2
(...,)	..	..	..	..	..	201,8	0,004931	3
(...,)	..	..	..	..	..	204,2	0,004873	4
(...,)	..	..	..	..	..	204,2	0,004873	5
(...,)	..	..	..	..	..	170,4	0,005834	6
(...,)	9	6	1	12	0	167,1	0,005949	0
Terbaik	9	6	1	12	0	Fitness	0,005949	

Individu yang dipilih

Bee Ke - <i>i</i>	Rute					Fitness	Trial
x <sub>1</sub>	1	12	0	9	6	0,005949	6
x <sub>2</sub>	6	1	12	0	9	0,005949	11
x <sub>3</sub>	0	9	6	1	12	0,005949	6
x <sub>4</sub>	9	6	1	12	0	0,005949	0

Sebelum memasuki fase onlooker bee, individu terpilih akan diseleksi menggunakan metode roulette wheel. Pertama dihitung probabilitas dari setiap individu.

Bee Ke - <i>i</i>	Rute					Fitness	Prob
x <sub>1</sub>	1	12	0	9	6	0,005949	0,25
x <sub>2</sub>	6	1	12	0	9	0,005949	0,25
x <sub>3</sub>	0	9	6	1	12	0,005949	0,25
x <sub>4</sub>	9	6	1	12	0	0,005949	0,25
						0,023795	

$$Prob_i = \frac{fitness(x_i)}{\sum_{k=1}^S fitness(x_k)}$$

#### 5.4.4 Improvement Solution (Fase Onlooker Bee)

Setelah tiap individu dihitung probabilitasnya, dilakukan perhitungan probabilitas kumulatifnya untuk mendapatkan *range* pada tiap individu.

Bee Ke - <i>i</i>	Prob	ProbCum	Range
x <sub>1</sub>	0,25	0,25	25-Jan
x <sub>2</sub>	0,25	0,50	26 - 50
x <sub>3</sub>	0,25	0,75	51 - 75
x <sub>4</sub>	0,25	1	76 - 100

### Seleksi roulette wheel

Bee Ke - <i>i</i>	Rand	Terpilih	Rute					Fitness
x <sub>1</sub>	53	3	0	9	6	1	12	0,005949
x <sub>2</sub>	29	2	6	1	12	0	9	0,005949
x <sub>3</sub>	84	4	9	6	1	12	0	0,005949
x <sub>4</sub>	68	3	0	9	6	1	12	0,005949

#### a. Perhitungan *insert operator*

Bee Ke - <i>i</i>	Rute					(IO)	Bee Ke - <i>i</i>	Rute					Jarak	Fitness
x <sub>1</sub>	0	9	6	1	12	(1,3)	x <sub>1</sub>	6	0	9	1	12	204,6	0,004864
x <sub>2</sub>	6	1	12	0	9	(2,3)	x <sub>2</sub>	6	12	1	0	9	204,1	0,004876
x <sub>3</sub>	9	6	1	12	0	(1,4)	x <sub>3</sub>	12	9	6	1	0	167,4	0,005938
x <sub>4</sub>	0	9	6	1	12	(3,5)	x <sub>4</sub>	0	9	12	6	1	201,8	0,004931

Keterangan:

IO dibangkitkan secara random

Perbandingan nilai Fitness (initial solution dan hasil swap operator)

Bee Ke - <i>i</i>	Awal			Hasil (IO)			Trial
	Total Jarak		Fitness	Total Jarak		Fitness	
x <sub>1</sub>	167,1		0,005949	204,6		0,004864	7
x <sub>2</sub>	167,1		0,005949	204,1		0,004876	12
x <sub>3</sub>	167,1		0,005949	167,4		0,005938	1
x <sub>4</sub>	167,1		0,005949	201,8		0,004931	7

Jika nilai fitness yang baru memiliki nilai yang lebih tinggi, maka trial pada masing-masing individu diberi nilai 0. Jika tidak, maka trial ditambahkan dengan 1.

Individu yang dipilih

Bee Ke - <i>i</i>	Rute					Fitness
x <sub>1</sub>	0	9	6	1	12	0,005949
x <sub>2</sub>	6	1	12	0	9	0,005949
x <sub>3</sub>	9	6	1	12	0	0,005949
x <sub>4</sub>	0	9	6	1	12	0,005949

Individu yang dipilih merupakan hasil dari perbandingan nilai fitness individu awal dengan individu baru dengan nilai fitness yang lebih besar.

b. Perhitungan *insert sequence*

**Perhitungan swap sequences bee ke-1**

Bee Ke-1	0	9	6	1	12	Fitness	0,005949	Trial
(IO)	Rute					Jarak	Fitness	
(1,2)	9	0	6	1	12	201,6	0,004936	8
(...,)	..	..	..	..	..	204,6	0,004864	9
(...,)	..	..	..	..	..	204,1	0,004876	10
(...,)	..	..	..	..	..	167,1	0,005949	11
(...,)	..	..	..	..	..	170,2	0,005841	12
(...,)	..	..	..	..	..	170,4	0,005834	13
(...,)	..	..	..	..	..	177,4	0,005605	14
(...,)	..	..	..	..	..	167,8	0,005924	15
(...,)	..	..	..	..	..	201,8	0,004931	16
(...,)	..	..	..	..	..	204,1	0,004876	17
Terbaik	0	9	6	1	12	Fitness	0,005949	

**Perhitungan swap sequences bee ke-2**

Bee Ke-2	6	1	12	0	9	Fitness	0,005949	Trial
(IO)	Rute					Jarak	Fitness	
(1,2)	1	6	12	0	9	167,8	0,005924	13
(...,)	..	..	..	..	..	201,8	0,004931	14
(...,)	..	..	..	..	..	201,6	0,004936	15
(...,)	..	..	..	..	..	167,1	0,005949	16
(...,)	..	..	..	..	..	204,1	0,004876	17
(...,)	..	..	..	..	..	204,2	0,004873	18
(...,)	..	..	..	..	..	170,2	0,005841	19
(...,)	..	..	..	..	..	167,4	0,005938	20
(...,)	..	..	..	..	..	201,6	0,004936	21
(...,)	..	..	..	..	..	201,6	0,004936	22
Terbaik	6	1	12	0	9	Fitness	0,005949	

### Perhitungan swap sequences bee ke-3

Bee Ke-3	9	6	1	12	0	Fitness	0,005949	Trial
(IO)	Rute					Jarak	Fitness	
(1,2)	6	9	1	12	0	170,2	0,005841	2
(...,)	..	..	..	..	..	170,4	0,005834	3
(...,)	..	..	..	..	..	167,4	0,005938	4
(...,)	..	..	..	..	..	167,1	0,005949	5
(...,)	..	..	..	..	..	167,8	0,005924	6
(...,)	..	..	..	..	..	201,8	0,004931	7
(...,)	..	..	..	..	..	182,7	0,005444	8
(...,)	..	..	..	..	..	204,1	0,004876	9
(...,)	..	..	..	..	..	204,2	0,004873	10
(...,)	..	..	..	..	..	167,4	0,005938	11
Terbaik	9	6	1	12	0	<b>Fitness</b>	0,005949	

### Perhitungan swap sequences bee ke-4

Bee Ke-4	0	9	6	1	12	Fitness	0,005949	Trial
(IO)	Rute					Jarak	Fitness	
(1,2)	9	0	6	1	12	201,6	0,004936	8
(...,)	..	..	..	..	..	204,6	0,004864	9
(...,)	..	..	..	..	..	204,1	0,004876	10
(...,)	..	..	..	..	..	167,1	0,005949	11
(...,)	..	..	..	..	..	170,2	0,005841	12
(...,)	..	..	..	..	..	170,4	0,005834	13
(...,)	..	..	..	..	..	177,4	0,005605	14
(...,)	..	..	..	..	..	167,8	0,005924	15
(...,)	..	..	..	..	..	201,8	0,004931	16
(...,)	..	..	..	..	..	204,1	0,004876	17
Terbaik	0	9	6	1	12	<b>Fitness</b>	0,005949	

Pada fase *scout bee* ditampilkan individu awal yang pertama dimasukkan dan individu terpilih dari hasil perhitungan *Improvement Solution*.

#### Individu Awal

Bee Ke- <i>i</i>	Rute					Fitness
x <sub>1</sub>	0	6	12	9	1	0,00486
x <sub>2</sub>	6	1	12	0	9	0,00595
x <sub>3</sub>	9	12	6	1	0	0,00493
x <sub>4</sub>	1	6	0	9	12	0,00494

#### Individu Terpilih

Bee Ke- <i>i</i>	Rute					Fitness
x <sub>1</sub>	0	9	6	1	12	0,005949
x <sub>2</sub>	6	1	12	0	9	0,005949
x <sub>3</sub>	9	6	1	12	0	0,005949
x <sub>4</sub>	0	9	6	1	12	0,005949

#### Simpan solusi terbaik (global best)

Bee Ke- <i>i</i>	Rute					Fitness
x <sub>3</sub>	9	6	1	12	0	0,005949

jika nilai *fitness* tertinggi berjumlah lebih dari satu, maka akan dilakukan pemeriksaan *trial*. Salah satu dari hasil nilai *fitness* tertinggi dan memiliki nilai *trial* terendah, akan dipilih menjadi solusi yang terbaik.

### 5.4.5 Fase Scout Bee

Evaluasi, *Limit* = 5, (*M*=Max(*Trial*)=22)

Individu awal yang pertama dimasukkan dan hasil *Improvement Solution*

#### Individu awal

Bee Ke- <i>i</i>	Rute					Fitness
x <sub>1</sub>	0	6	12	9	1	0,004860
x <sub>2</sub>	6	1	12	0	9	0,005949
x <sub>3</sub>	9	12	6	1	0	0,004930
x <sub>4</sub>	1	6	0	9	12	0,004940

Individu terpilih

Bee Ke -i	Rute					Fitness	Trial	Ada Perbaikan
x <sub>1</sub>	0	9	6	1	12	0,005949	17	Ada
x <sub>2</sub>	6	1	12	0	9	0,005949	22	Tidak
x <sub>3</sub>	9	6	1	12	0	0,005949	11	Ada
x <sub>4</sub>	0	9	6	1	12	0,005949	17	Ada

Aturan: Jika  $M > \text{Limit}$ , maka hapus individu yang tidak ada perbaikan, ganti dengan individu baru hasil random, kemudian reset trial = 0, jika  $M > \text{Limit}$  dan terdapat individu yang ada perbaikan, maka tidak perlu digantikan dengan individu baru, kemudian reset trial = 0. Jika  $M < \text{Limit}$ , maka tidak perlu digantikan dengan individu baru, dan nilai trial tidak perlu di-reset.

Maka, Individu barunya adalah sebagai berikut:

Bee Ke-i	Rute					Trial
x <sub>1</sub>	0	9	6	1	12	0
x <sub>2</sub>	1	6	12	9	0	0
x <sub>3</sub>	9	6	1	12	0	0
x <sub>4</sub>	0	9	6	1	12	0

Bee Ke-2 di-generate secara random, karena tidak ada perbaikan

**Masuk pada iterasi ke-2**

- **Improvement Solution (Fase Employeed Bee)**
- **Improvement Solution (Fase Onlooker Bee)**
- **Fase Scout Bee**

Lakukan Fase-fase tersebut sampai iterasi Maksimum.

## 5.5 Tugas Kelompok

1. Jelaskan perbedaan konsep representasi solusi, serta fase iteratifnya dari optimasi real code dan masalah kombinatorial pada ABC?
2. Jelaskan perbedaan antara *Employeed Bee*, *Onlooker Bee*, dan *Scout Bee* pada ABC?
3. Jika diketahui  $x_1=[1\ 5\ 2\ 7\ 3\ 9\ 4\ 6\ 8]$  dan  $SO_1=(1,4)$ ,  $SO_2=(5,4)$ ,  $IO_1=(1,4)$ ,  $IO_2=(5,4)$ . Tentukan:

a.  $x_1 + SO_2$

b.  $x_1 + SS$

c.  $x_1 + IO_1$

d.  $x_1 + IS$

4. Tentukan hasil dari  $v_{1j}$  jika diketahui  $k=2$ , dan  $v_{3j}$  jika diketahui  $k=1$ !

---

bee	$j=1$	$j=2$	<i>fitness</i>
$x_1$	1,4898	2,0944	..
$x_2$	8,4917	2,5754	..
$x_3$	-4,5575	0,1679	..

# BAB 6 Algoritma Ant Colony Optimization (ACO)

## 6.1 Pengantar

Algoritma koloni semut ini diperkenalkan oleh Moysen dan Manderick pada tahun 1996, setelah itu dikembangkan oleh Marco Dorigo. Algoritma ini mendapat inspirasi dari perilaku semut yang mencari makanan dari sarangnya. Semut mencari makanan dengan melepaskan feromon sebagai alat penanda dalam jalur yang dilewatinya. Perilaku semut yang melepaskan feromon ini sangat berguna bagi kelompoknya untuk mendapatkan jalur optimal dalam menemukan tempat makanannya. Proses meninggalkan feromon ini disebut stigmergy, yaitu proses memodifikasi lingkungan untuk mengingat jalan kembali ke sarang dan merupakan alat berkomunikasi bagi semut.

## 6.2 Discrete Ant Colony Optimization (ACO)

Berikut langkah-langkah Ant Colony Optimization (ACO) Untuk TSP:

1. Inisialisasi
  - a. Set nilai Tetapan siklus-semut ( $Q$ ), Tetapan pengendali intensitas jejak feromon semut ( $\alpha > 0$ ), Tetapan pengendali visibilitas ( $\beta > 0$ ), Tetapan penguapan jejak feromon semut ( $0 < \rho \leq 1$ ), banyak semut( $m$ ), banyak iterasi ( $NC_{max}$ )
  - b. Set feromon awal (dengan menentukan satu solusi awal dengan algoritma greedy, lalu hitung  $cost$ -nya ( $c_{greedy}$ ) atau dengan langsung diset) dan hitung nilai visibilitas

$$\tau_0 = \frac{m}{C_{greedy}}$$

atau

$$\tau_0 = 0.021$$

$$\eta_{ij} = \frac{1}{d_{ij}}$$

$k = 1, 2, \dots, m$

$i, j = 1, 2, \dots, n$  (banyak kota)

2. Penyusunan jalur kunjungan setiap semut ( $t=1$ ):

$$\tau_{ij}(t=1) = \tau_0 \longrightarrow \boxed{\text{Khusus ketika } t=1}$$

- a. Inisialisasi kota pertama setiap semut

b. Memilih kota yang dikunjungi dari nilai Probabilitas tertinggi

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{k' \in allowed_k} [\tau_{ik'}(t)]^\alpha [\eta_{ik'}(t)]^\beta}, & \text{jika } j \in allowed_k \\ 0, & \text{lainnya} \end{cases}$$

Dimana:

$p_{ij}^k(t)$  : Peluang semut ke- $k$  untuk berkunjung dari kota  $i$  ke kota  $j$  pada iterasi ke- $t$

c. Mencatat setiap kota yang dikunjungi (Memori)

**Note:** Setelah semua kota dikunjungi dari setiap semut, lalu hitung *cost*, lalu simpan solusi terbaik (*global best*)

3. Cek kondisi berhenti ( $t = t + 1$  (apakah masih  $\leq NC_{max}$ )), jika belum memenuhi kondisi berhenti:

a. Hitung matrik perubahan intensitas feromon tiap semut, jika jalur (i,j) atau (j,i) ada dalam Memori, gunakan rumus berikut:

$$\Delta \tau_{ij}^k = \frac{Q}{Cost_k}$$

jika jalur (i,j) atau (j,i) tidak ada dalam Memori, maka  $\Delta \tau_{ij}^k = 0$

b. Hitung matrik perubahan intensitas feromon global dari tiap semut

$$\Delta \tau_{ij} = \sum_{k=1}^m \Delta \tau_{ij}^k$$

c. Update feromon dengan rumus

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta \tau_{ij}$$

d. Kosongkan tabel jalur kunjungan setiap semut

e. Kembali ke langkah (2)

### 6.3 Case Study: Travelling Salesman Problem (TSP)

Contoh penyelesaian masalah TSP:

Jarak ( $d_{ij}$ )	1	2	3	4	5
1	0	39.1	80.6	36.9	35.1

2	39.1	0	101	27	24.9
3	80.6	101	0	84.9	85.2
4	36.9	27	84.9	0	4
5	35.1	24.9	85.2	4	0

### 6.3.1 Inisialisasi

- Set nilai Tetapan siklus-semut ( $Q = 1$ ), Tetapan pengendali intensitas jejak feromon semut ( $\alpha = 1$ ), Tetapan pengendali visibilitas ( $\beta = 1$ ), Tetapan penguapan jejak feromon semut ( $\rho = 0.1$ ), banyak semut( $m = 5$ ), banyak iterasi ( $NC_{max} = 2$ )
- Set feromon awal dan hitung nilai visibilitas

$$\tau_0 = 0.021$$

$$\eta_{ij} = \frac{1}{d_{ij}}$$

Tabel Nilai visibilitas

	1	2	3	4	5
1	0	0.026	0.012	0.027	0.028
2	0.026	0	0.01	0.037	0.04
3	0.012	0.01	0	0.012	0.012
4	0.027	0.037	0.012	0	0.25
5	0.028	0.04	0.012	0.25	0

### 6.3.2 Penyusunan Jalur Kunjungan Setiap Semut

- Inisialisasi **kota pertama** setiap semut
- Memilih kota yang dikunjungi dari nilai **Probabilitas tertinggi**
- Mencatat setiap kota yang dikunjungi (**Memori**)

$$\tau_{ij}(t=1) = \tau_0 = 0.021$$

	1	2	3	4	5
1	0.021	0.021	0.021	0.021	0.021
2	0.021	0.021	0.021	0.021	0.021
3	0.021	0.021	0.021	0.021	0.021
4	0.021	0.021	0.021	0.021	0.021
5	0.021	0.021	0.021	0.021	0.021

$\eta_{ij}$  (Nilai visibilitas)

	1	2	3	4	5
1	0	0.026	0.012	0.027	0.028
2	0.026	0	0.01	0.037	0.04
3	0.012	0.01	0	0.012	0.012
4	0.027	0.037	0.012	0	0.25
5	0.028	0.04	0.012	0.25	0

Kota Awal	Probabilitas					Max Probabilitas	Tujuan	Memori
	kota 1	kota 2	kota 3	kota 4	kota 5			
1	0	0.28	0.129	0.29	<b>0.301</b>	0.301	5	[1 5]
2	0.23	0	0.088	0.327	<b>0.354</b>	0.354	5	[2 5]
3	<b>0.261</b>	0.217	0	0.261	0.261	0.261	1	[3 1]
4	0.083	0.113	0.037	0	<b>0.767</b>	0.767	5	[4 5]
5	0.085	0.121	0.036	<b>0.758</b>	0	0.758	4	[5 4]

Ket:

Memori = kota ke 2

$$\tau_{ij}(t=1) = \tau_0 = 0.021$$

	1	2	3	4	5
1	0.021	0.021	0.021	0.021	0.021
..	..	..	..	..	..
5	0.021	0.021	0.021	0.021	0.021

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{k' \in \text{allowed}_k} [\tau_{ik'}(t)]^\alpha [\eta_{ik'}(t)]^\beta}, & \text{jika } j \in \text{allowed}_k \\ 0, \text{ lainnya} & \end{cases}$$

$$p_{11}^1(1) = 0$$

$$p_{12}^1(1) = \frac{[\tau_{12}(1)]^\alpha [\eta_{12}(1)]^\beta}{[\tau_{12}(1)]^\alpha [\eta_{12}(1)]^\beta + [\tau_{13}(1)]^\alpha [\eta_{13}(1)]^\beta + [\tau_{14}(1)]^\alpha [\eta_{14}(1)]^\beta + [\tau_{15}(1)]^\alpha [\eta_{15}(1)]^\beta}$$

Kota Awal	Probabilitas					Max Probabilitas	Tujuan	Memori
	kota 1	kota 2	kota 3	kota 4	kota 5			
1	0	0.28	0.129	0.29	<b>0.301</b>	0.301	5	[1 5]
..	..	..	..	..	..	..	..	..
5	0.085	0.121	0.036	<b>0.758</b>	0	0.758	4	[5 4]

- Memilih kota yang dikunjungi dari nilai **Probabilitas tertinggi**
- Mencatat setiap kota yang dikunjungi (**Memori**)

	1	2	3	4	5
1	0.021	0.021	0.021	0.021	0.021
2	0.021	<b>0.021</b>	0.021	0.021	0.021
3	0.021	0.021	<b>0.021</b>	0.021	0.021
4	0.021	0.021	0.021	<b>0.021</b>	0.021
5	0.021	0.021	0.021	0.021	<b>0.021</b>

$\eta_{ij}$  (Nilai visibilitas)

	1	2	3	4	5
1	0	0.026	0.012	0.027	0.028
2	0.026	0	0.01	0.037	0.04
3	0.012	0.01	0	0.012	0.012
4	0.027	0.037	0.012	0	0.25
5	0.028	0.04	0.012	0.25	0

Kota Awal	Probabilitas					Max Probabilitas	Tujuan	Memori
	kota 1	kota 2	kota 3	kota 4	kota 5			
1	0	0.4	0.185	<b>0.415</b>	0	0.415	4	[1 5 4]
2	0.356	0	0.137	<b>0.507</b>	0	0.507	4	[2 5 4]
3	0	0.294	0	0.353	<b>0.353</b>	0.353	5	[3 1 5]
4	0.355	<b>0.487</b>	0.158	0	0	0.487	2	[4 5 2]
5	0.35	<b>0.5</b>	0.15	0	0	0.5	2	[5 4 2]

Ket:

Memori = kota ke 3

- Memilih kota yang dikunjungi dari nilai **Probabilitas tertinggi**
- Mencatat setiap kota yang dikunjungi (**Memori**)

$$\tau_{ij}(t=1) = \tau_0 = 0.021$$

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>1</b>	0.021	0.021	0.021	0.021	0.021
<b>2</b>	0.021	<b>0.021</b>	0.021	0.021	0.021
<b>3</b>	0.021	0.021	<b>0.021</b>	0.021	0.021
<b>4</b>	0.021	0.021	0.021	<b>0.021</b>	0.021
<b>5</b>	0.021	0.021	0.021	0.021	<b>0.021</b>

$\eta_{ij}$  (Nilai visibilitas)

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>1</b>	0	0.026	0.012	0.027	0.028
<b>2</b>	0.026	0	0.01	0.037	0.04
<b>3</b>	0.012	0.01	0	0.012	0.012
<b>4</b>	0.027	0.037	0.012	0	0.25
<b>5</b>	0.028	0.04	0.012	0.25	0

Kota Awal	Probabilitas					Max Probabilitas	Tujuan	Memori
	kota 1	kota 2	kota 3	kota 4	kota 5			
1	0	<b>0.684</b>	0.316	0	0	0.684	2	[1 5 4 2]
2	<b>0.722</b>	0	0.278	0	0	0.722	1	[2 5 4 1]
3	0	0.455	0	<b>0.545</b>	0	0.545	4	[3 1 5 4]
4	<b>0.692</b>	0	0.308	0	0	0.692	1	[4 5 2 1]
5	<b>0.7</b>	0	0.3	0	0	0.7	1	[5 4 2 1]

Ket:

Memori = kota keempat

- Memilih kota yang dikunjungi dari nilai **Probabilitas tertinggi**
- Mencatat setiap kota yang dikunjungi (**Memori**)

$$\tau_{ij}(t=1) = \tau_0 = 0.021$$

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>1</b>	0.021	0.021	0.021	0.021	0.021
<b>2</b>	0.021	<b>0.021</b>	0.021	0.021	0.021
<b>3</b>	0.021	0.021	<b>0.021</b>	0.021	0.021
<b>4</b>	0.021	0.021	0.021	<b>0.021</b>	0.021
<b>5</b>	0.021	0.021	0.021	0.021	<b>0.021</b>

$\eta_{ij}$  (Nilai visibilitas)

	1	2	3	4	5
1	0	0.026	0.012	0.027	0.028
2	0.026	0	0.01	0.037	0.04
3	0.012	0.01	0	0.012	0.012
4	0.027	0.037	0.012	0	0.25
5	0.028	0.04	0.012	0.25	0

Kota Awal	Probabilitas					Max Probabilitas	Tujuan	Memori
	kota 1	kota 2	kota 3	kota 4	kota 5			
1	0	0	1	0	0	1	3	[1 5 4 2 3]
2	0	0	1	0	0	1	3	[2 5 4 1 3]
3	0	1	0	0	0	1	2	[3 1 5 4 2]
4	0	0	1	0	0	1	3	[4 5 2 1 3]
5	0	0	1	0	0	1	3	[5 4 2 1 3]

Ket:

Memori = kota kelima

Setelah semua kota dikunjungi, hitung *cost*, lalu simpan solusi terbaik (*global best*)

Kota Awal	Memori	Cost (Total Jarak)
1	[1 5 4 2 3]	247.7
2	[2 5 4 1 3]	247.4
3	[3 1 5 4 2]	247.7
4	[4 5 2 1 3]	233.5
5	[5 4 2 1 3]	235.9

### 6.3.3 Cek kondisi berhenti

Cek kondisi berhenti ( $t = t + 1 = 1 + 1 = 2$  (masih  $\leq NC_{max}$ )), jika belum memenuhi kondisi berhenti:

- a. Hitung **matrik perubahan intensitas feromon tiap semut**, jika jalur (i,j) atau (j,i) ada dalam **Memori**, gunakan rumus berikut:

$$\Delta \tau_{ij}^k = \frac{Q}{Cost_k} \quad \Delta \tau_{ij}^1 = \frac{Q=1}{Cost_1} = \frac{1}{247.7} = 0.004037$$

jika jalur (i,j) atau (j,i) tidak ada dalam **Memori**, maka

$$\Delta \tau_{ij}^k = 0$$

Semut ke- <i>k</i>	Memori	Cost (Total Jarak)
1	[1 5 4 2 3]	247.7
2	[2 5 4 1 3]	247.4
3	[3 1 5 4 2]	247.7
4	[4 5 2 1 3]	233.5
5	[5 4 2 1 3]	235.9

Matrik Perubahan Intensitas Feromon Semut ke-1

$\Delta \tau_{ij}^1$	1	2	3	4	5
1	0	0	$\Delta \tau_{13}^1$	0	$\Delta \tau_{15}^1$
2	0	0	$\Delta \tau_{23}^1$	$\Delta \tau_{24}^1$	0
3	$\Delta \tau_{31}^1$	$\Delta \tau_{32}^1$	0	0	0
4	0	$\Delta \tau_{42}^1$	0	0	$\Delta \tau_{45}^1$
5	$\Delta \tau_{51}^1$	0	0	$\Delta \tau_{54}^1$	0

Matrik Perubahan Intensitas Feromon Semut ke-1

$\Delta \tau_{ij}^1$	1	2	3	4	5
1	0	0	0.004037	0	0.004037
2	0	0	0.004037	0.004037	0
3	0.004037	0.004037	0	0	0
4	0	0.004037	0	0	0.004037
5	0.004037	0	0	0.004037	0

Semut ke- <i>k</i>	Memori	Cost (Total Jarak)
1	[1 5 4 2 3]	247.7
2	[2 5 4 1 3]	247.4
3	[3 1 5 4 2]	247.7
4	[4 5 2 1 3]	233.5
5	[5 4 2 1 3]	235.9

$$\Delta \tau_{ij}^1 = \frac{Q=1}{Cost_1} = \frac{1}{247.7} = 0.004037$$

### Matrik Perubahan Intensitas Feromon Semut ke-2

$\Delta \tau_{ij}^2$	1	2	3	4	5
1	0	0	0.004042	0.004042	0
2	0	0	0.004042	0	0.004042
3	0.004042	0.004042	0	0	0
4	0.004042	0	0	0	0.004042
5	0	0.004042	0	0.004042	0

Semut ke- <i>k</i>	Memori	Cost (Total Jarak)
1	[1 5 4 2 3]	247.7
2	[2 5 4 1 3]	247.4
3	[3 1 5 4 2]	247.7
4	[4 5 2 1 3]	233.5
5	[5 4 2 1 3]	235.9

$$\Delta \tau_{ij}^3 = \frac{Q=1}{Cost_3} = \frac{1}{247.7} = 0.004037$$

$\Delta \tau_{ij}^3$	1	2	3	4	5
1	0	0	0.004037	0	0.004037
2	0	0	0.004037	0.004037	0
3	0.004037	0.004037	0	0	0
4	0	0.004037	0	0	0.004037
5	0.004037	0	0	0.004037	0

Semut ke- <i>k</i>	Memori	Cost (Total Jarak)
1	[1 5 4 2 3]	247.7
2	[2 5 4 1 3]	247.4
3	[3 1 5 4 2]	247.7
4	[4 5 2 1 3]	233.5
5	[5 4 2 1 3]	235.9

$$\Delta \tau_{ij}^4 = \frac{Q=1}{Cost_4} = \frac{1}{233.5} = 0.004283$$

$\Delta \tau_{ij}^4$	1	2	3	4	5

<b>1</b>	0	0.004283	0.004283	0	0
<b>2</b>	0.004283	0	0	0	0.004283
<b>3</b>	0.004283	0	0	0.004283	0
<b>4</b>	0	0	0.004283	0	0.004283
<b>5</b>	0	0.004283	0	0.004283	0

Semut ke- <i>k</i>	Memori	Cost (Total Jarak)
1	[1 5 4 2 3]	247.7
2	[2 5 4 1 3]	247.4
3	[3 1 5 4 2]	247.7
4	[4 5 2 1 3]	233.5
<b>5</b>	<b>[5 4 2 1 3]</b>	<b>235.9</b>

$$\Delta \tau_{ij}^5 = \frac{Q=1}{Cost_5} = \frac{1}{235.9} = 0.004239$$

Matrik Perubahan Intensitas Feromon Semut ke-5

$\Delta \tau_{ij}^5$	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>1</b>	0	0.004239	0.004239	0	0
<b>2</b>	0.004239	0	0	0.004239	0
<b>3</b>	0.004239	0	0	0	0.004239
<b>4</b>	0	0.004239	0	0	0.004239
<b>5</b>	0	0	0.004239	0.004239	0

- b. Hitung matrik perubahan intensitas feromon global dari tiap semut

$$\Delta \tau_{ij} = \sum_{k=1}^m \Delta \tau_{ij}^k = \sum_{k=1}^5 \Delta \tau_{ij}^k = \Delta \tau_{ij}^1 + \Delta \tau_{ij}^2 + \Delta \tau_{ij}^3 + \Delta \tau_{ij}^4 + \Delta \tau_{ij}^5$$

$\Delta \tau_{ij}$	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>1</b>	0	0.008522	0.020638	0.004042	0.008074
<b>2</b>	0.008522	0	0.012116	0.012313	0.008325
<b>3</b>	0.020638	0.012116	0	0.004283	0.004239
<b>4</b>	0.004042	0.012313	0.004283	0	0.020638
<b>5</b>	0.008074	0.008325	0.004239	0.020638	0

Contoh, hitung  $\Delta \tau_{12}$  :

$$\begin{aligned}\Delta \tau_{12} &= \Delta \tau_{12}^1 + \Delta \tau_{12}^2 + \\&\quad \Delta \tau_{12}^3 + \Delta \tau_{12}^4 + \Delta \tau_{12}^5 \\ \Delta \tau_{12} &= 0 + 0 + \\&\quad 0 + 0.004283 + \\&\quad 0.004239 \\ \Delta \tau_{12} &= 0.008522\end{aligned}$$

### c. Update Feromon

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta \tau_{ij}$$

$\tau_{ij}(t+1)$	1	2	3	4	5
1	0.0021	0.010622	0.022738	0.006142	0.010174
2	0.010622	0.0021	0.014216	0.014413	0.010425
3	0.022738	0.014216	0.0021	0.006383	0.006339
4	0.006142	0.014413	0.006383	0.0021	0.022738
5	0.010174	0.010425	0.006339	0.022738	0.0021

Contoh, hitung  $\tau_{12}(t+1)$  :

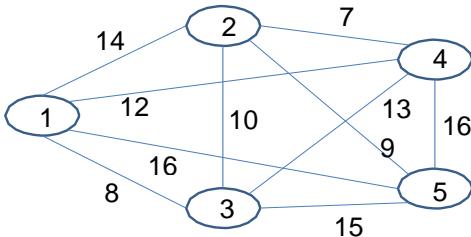
$$\begin{aligned}\Delta \tau_{12} &= \Delta \tau_{12}^1 + \Delta \tau_{12}^2 + \\&\quad \Delta \tau_{12}^3 + \Delta \tau_{12}^4 + \Delta \tau_{12}^5 \\ \Delta \tau_{12} &= 0 + 0 + \\&\quad 0 + 0.004283 + \\&\quad 0.004239 \\ \Delta \tau_{12} &= 0.008522\end{aligned}$$

- Kosongkan tabel jalur kunjungan setiap semut
- Kembali ke langkah (2)

## 6.4 Tugas Kelompok

- Jelaskan pengertian dari istilah-istilah berikut pada Ant Colony Optimization (ACO)!
  - Nilai alfa ( $\alpha$ ) dan Beta ( $\beta$ )
  - Nilai  $Q$

- c. Nilai rho ( $\rho$ )
  - d. Konsep nilai feromon dari:  $\tau_0$ ,  $\Delta\tau_{ij}^k$ ,  $\Delta\tau_{ij}$ ,  $\tau_{ij}$
2. Selesaikan permasalahan TSP berikut menggunakan Ant Colony Optimization (ACO)!



Tabel jarak antar simpul

Node	1	2	3	4	5
1	-	14	8	12	16
2	14	-	10	7	9
3	8	10	-	13	15
4	12	7	13	-	16
5	16	9	15	16	-

**Note:** diketahui nilai Tetapan siklus-semut ( $Q = 1$ ), Tetapan pengendali intensitas jejak feromon semut ( $\alpha = 2$ ), Tetapan pengendali visibilitas ( $\beta = 1$ ), Tetapan penguapan jejak feromon semut ( $\rho = 0.2$ ), banyak semut( $m = 5$ ), banyak iterasi ( $NC_{max} = 2$ )

# BAB 7 Topik Lanjut Pada Swarm Intelligence

## 7.1 Pengantar

Algoritma koloni semut ini diperkenalkan oleh Moyson dan Manderick pada tahun 1996, setelah itu dikembangkan oleh Marco Dorigo. Algoritma ini mendapat **inspirasi dari perilaku semut** yang **mencari makanan** dari sarangnya. Semut mencari makanan dengan **melepaskan feromon** sebagai alat penanda **dalam jalur yang dilewatinya**. Perilaku semut yang melepaskan feromon ini sangat berguna bagi kelompoknya untuk mendapatkan jalur optimal dalam menemukan tempat makanannya. Proses meninggalkan feromon ini disebut **stigmergy**, yaitu proses memodifikasi lingkungan untuk mengingat jalan kembali ke sarang dan merupakan **alat berkomunikasi bagi semut**.

Pheromone digunakan **sebagai komunikasi antar semut** pada saat membangun rute agar dapat mencapai titik target (Tyas, 2013; Martin, 2012). Setiap **semut** memiliki **kemampuan** untuk **mengeksplorasi** dan **mengeksplorasi** informasi pheromone yang telah ditinggalkan saat mereka melintasi jalur tersebut (Lixing, 2010). Dalam metode ACO, **perilaku semut** buatan dimanfaatkan **untuk mencari solusi** mulai dari penentuan awal node dan penentuan **perpindahan node ke node** tetangga yang layak dalam rangka proses penyelesaian masalah.

**Penguapan pheromone** adalah proses penurunan intensitas jalur pheromone karena telah melebihi batas waktu. Proses tersebut digunakan untuk **menghindari lokal konvergensi** dan untuk **mengeksplorasi** pecarian area yang **lebih luas**. **Setiap semut berulang kali membangun jalurnya** dengan menerapkan **stochastic greedy rule**, yang biasa dikenal dengan **transition rule** (Lixing, 2010). ACO merupakan algoritma yang sangat tepat jika digunakan untuk permasalahan optimasi, karena memiliki **natural self-learning** yang sangat kuat (Tianshi, 2014). Contoh konsep optimasi sebagai Topik Lanjut Pada Swarm Intelligence, misalnya:

- Algoritma Optimasi<sub>1</sub> + Algoritma Optimasi<sub>2</sub> + ... + Algoritma Optimasi<sub>n</sub>

- Algoritma Optimasi<sub>1</sub> + Algoritma Optimasi<sub>2</sub> + ... + Algoritma Optimasi<sub>n</sub> + Algoritma Klasifikasi
- Algoritma Optimasi<sub>1</sub> + Algoritma Optimasi<sub>2</sub> + ... + Algoritma Optimasi<sub>n</sub> + Algoritma Regresi (Peramalan dan prediksi)
- Algoritma Optimasi<sub>1</sub> + Algoritma Optimasi<sub>2</sub> + ... + Algoritma Optimasi<sub>n</sub> + Algoritma Clustering
- Algoritma Optimasi<sub>1</sub> + Algoritma Optimasi<sub>2</sub> + ... + Algoritma Optimasi<sub>n</sub> + Algoritma Rekomendasi (misal di MK Sistem Pendukung Keputusan)
- Dan lainnya.

Konsep hybrid sederhana dan tingkat lanjut, misal:

- **Hybrid PSO** dengan **Fungsi Tertentu (yang sederhana)**
- **Hybrid ACO** dengan Algoritma tertentu (misal **SVR**, yang didalam algoritma tersebut terdapat Fungsi tertentu, misal ketika menghitung nilai evaluasi dengan Fungsi MAPE)

$$MAPE = \frac{1}{N} \sum_{i=1}^N \left| \frac{\hat{y}_i - y_i}{y_i} \right| \times 100$$

Ilustrasi konsep hybrid sebagai perbandingan untuk mempermudah pemahaman:

### Hybrid PSO dengan Fungsi Tertentu

$$\max, f(x_1, x_2) = 19 + x_1 \sin(x_1 \pi) + (10 - x_2) \sin(x_2 \pi),$$
$$-5,0 \leq x_1 \leq 9,8 \quad 0,0 \leq x_2 \leq 7,3$$

Sudah diselesaikan di Bab ke-3

### Hybrid ACO dengan SVR

$$\min, f(\sigma, C, \epsilon) = MAPE$$

- $\sigma$  dapat dianggap sebagai  $x_1$
- $C$  dapat dianggap sebagai  $x_2$
- $\epsilon$  dapat dianggap sebagai  $x_3$

## 7.2 Hybrid ACO dengan SVR

Berikut langkah-langkah SVR-ACO (Lixing, 2010), yaitu menyelesaikan / Solve kasus Real Code dengan Combinatorial Code:

1. Inisialisasi:

- Tentukan parameter SVR dan batas minimum-maksimum parameter yang akan dioptimasi dengan ACO, misal  $\sigma$ , C dan  $\varepsilon$
- Set nilai Tetapan siklus-semut ( $0 \leq q_0 \leq 1$ ), Tetapan pengendali intensitas jejak feromon semut ( $\alpha > 0$ ), Tetapan pengendali visibilitas ( $\beta > 0$ ), Tetapan penguapan jejak feromon local dan global dari semut ( $0 < \rho < 1$ , dan  $0 < \delta < 1$ ), banyak semut( $m$ ), banyak iterasi ( $NC_{max}$ ), nilai feromon awal ( $\tau_0$ ).

$$\boxed{\begin{array}{l} k = 1, 2, \dots, m \\ i, j = 1, 2, \dots, n \text{ (banyak digit)} \end{array}}$$

2. Penyusunan jalur kunjungan setiap semut ( $t=1$ ):

$$\tau_{ij}(t=1) = \tau_0 \longrightarrow \boxed{\text{Khusus ketika } t=1}$$

Optimasi parameter (misal, parameter  $\sigma$ , C,  $\varepsilon$ ):

- Tentukan digit pertama secara acak sebagai nilai  $r=[0,9]$  untuk setiap semut.
- Tentukan digit berikutnya (nilai  $u$ ), dengan random  $q = [0,1]$  uniformly distributed, lalu bandingkan dengan  $q_0$ .

- Jika  $q \leq q_0$

$$P(r, u) = [\tau(r, u)]^\alpha \cdot [\tau(r, u)]^\beta, \quad u \in J(r)$$

$$u = \arg \max_{u \in J(r)} \left\{ [\tau(r, u)]^\alpha \cdot [\tau(r, u)]^\beta \right\}$$

- Jika  $q > q_0$ ,

- Jika  $s \in J(r)$ , maka

$$P(r, s) = \frac{[\tau(r, s)]^\alpha \cdot [\tau(r, s)]^\beta}{\sum_{s \in J(r)} [\tau(r, s)]^\alpha \cdot [\tau(r, s)]^\beta}$$

- Jika  $s \notin J(r)$ , maka

$$P(r, s) = 0$$

lalu, gunakan seleksi Roulette Wheel untuk menentukan  $u=s$ .

- Update feromon local

$$\tau(r,u) = (1-\rho)(\tau(r,u)) + \rho\tau_0$$

3. Cek kondisi berhenti ( $t = t + 1$  (apakah masih  $\leq NC_{max}$  ?)), jika belum memenuhi kondisi berhenti:

- a. Update Pheromone Global

$$\tau(r,u) = (1-\delta)\tau(r,u) + \delta.\Delta\tau(r,u)$$

dimana ada kondisi:

- Jika  $(r,u) \in \text{global best tour}$ , maka

$$\Delta\tau(r,u) = \frac{1}{L_{globalbest}} = MAPE_{globalbest}$$

- Jika  $(r,u) \notin \text{global best tour}$ , maka

$$\Delta\tau(r,u) = 0$$

- b. Kosongkan tabel jalur kunjungan setiap semut

- c. Kembali ke langkah (2)

## 7.3 Support Vector Regression (SVR)

Konsep SVR didasarkan pada *risk minimization*, yaitu untuk mengestimasi suatu fungsi dengan cara meminimalkan batas dari *generalization error*, sehingga SVR mampu mengatasi *overfitting*. Fungsi regresi dari metode SVR adalah sebagai berikut (Sethu Vijayakumar & Si Wu, 1999).

### ALGORITMA SEKUENSIAL TRAINING SVR :

1. Inisialisasi parameter SVR yang digunakan

2. Rumus memperoleh Matrik Hessian

$$R_{ij} = (K(x_i, x_j) + \lambda^2)$$

3. Untuk tiap training point lakukan:

$$E_i = y_i - \sum_{j=1}^l (\alpha_j^* - \alpha_j) R_{ij}$$

$$\delta\alpha_i^* = \min \left\{ \max \left\{ \gamma(E_i - \varepsilon), -\alpha_i^* \right\}, C - \alpha_i^* \right\}$$

$$\delta\alpha_i = \min \left\{ \max \left\{ \gamma(-E_i - \varepsilon), -\alpha_i \right\}, C - \alpha_i \right\}$$

$$\alpha_i^* = \alpha_i^* + \delta\alpha_i^*$$

$$\alpha_i = \alpha_i + \delta\alpha_i$$

4. Kembali pada langkah ketiga, sampai pada kondisi iterasi maksimum atau  $\max(|\delta\alpha_i|) < \varepsilon$  dan  $\max(|\delta\alpha_i^*|) < \varepsilon$

5. Dengan menggunakan fungsi peramalan sebagai berikut:

$$f(x) = \sum_{i=1}^l (\alpha_i^* - \alpha_i)(K(x_i, x) + \lambda^2)$$

### Metode Kernel

$$k(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)$$

Misal nilai  $\sigma = 0.7$

(Rajkumar et al., 2013) (Javed et al., 2009)

### Normalisasi Data

Persamaan yang digunakan untuk normalisasi data adalah (S Gopal et al., 2015) seperti yang ditunjukkan pada persamaan.

$$x' = \frac{(x - x_{\min})}{x_{\max} - x_{\min}}$$

Dimana:

- $x'$  = Hasil normalisasi data  
 $x$  = Nilai data yang akan dinormalisasi  
 $x_{\max}$  = Nilai maksimum dari dataset yang digunakan  
 $x_{\min}$  = Nilai minimum dari dataset yang digunakan

### Nilai Evaluasi

Persamaan yang digunakan untuk nilai evaluasi adalah MAPE.

## 7.4 Training dan Testing SVR

Berikut merupakan contoh proses training dan testing terhadap kasus peramalan nilai kurs. Diketahui Grafik Fluktuasi Nilai Tukar IDR terhadap USD Periode 2006-2015 yang ditunjukkan pada Gambar 6.1.



Gambar 7.1 Nilai Tukar IDR terhadap USD Periode 2006-2015

Grafik di atas adalah pergerakan nilai tukar mulai dari tahun 2006 hingga 2015(Nilai grafik tahunan). Dengan titik terkuat terjadi pada tahun 2011 dan titik terlemah terjadi pada tahun 2015. Berikut diketahui Data Nilai Tukar IDR Terhadap USD Juli 2015 yang ditunjukkan oleh Tabel 6.1.

Tabel 7.1 Data Nilai Tukar IDR Terhadap USD Juli 2015

TANGGAL	NILAI TUKAR
42190	13338
42191	13356
42192	13332
42193	13331
42194	13337
42195	13316
42196	13316
42197	13316
42198	13353
42199	13304
42200	13304
42201	13309

dan dataset dengan 4 fitur yang ditunjukkan oleh Tabel 6.2.

Tabel 7.2 Dataset dengan 4 fitur

No	Tgl/Bln/Thn	X1	X2	X3	X4	Y
1	9 Juli 2015	13338	13356	13332	13331	13337
2	10 Juli 2015	13356	13332	13331	13337	13316
3	11 Juli 2015	13332	13331	13337	13316	13316
4	12 Juli 2015	13331	13337	13316	13316	13316
5	13 Juli 2015	13337	13316	13316	13316	13353
6	14 Juli 2015	13313	13346	13347	13304	13304
7	15 Juli 2015	13346	13347	13304	13304	13304
8	16 Juli 2015	13347	13304	13304	13304	13309

Data Latih

Data Uji

### Proses Normalisasi

Setelah ditentukan data latih dan data uji maka selanjutnya adalah melakukan proses normalisasi data, dimana:

- $x'$  = Hasil normalisasi data  
 $x$  = Nilai data yang akan dinormalisasi  
 $x_{max}$  = Nilai maksimum dari keseluruhan data, misal mulai dari tahun 2006 – 2015  
 $x_{min}$  = Nilai minimum dari keseluruhan data, misal mulai dari tahun 2006 – 2015

Misal didapatkan,  $x_{min} = 9634$  dan  $x_{max} = 14728$

Dengan menggunakan persamaan 6.9 maka diperoleh hasil data normalisasi untuk data 1 fitur  $x_1$  adalah sebagai berikut:

$$x' = \frac{(x - x_{min})}{x_{max} - x_{min}}$$
$$x_1' = \frac{(13338 - 9634)}{14728 - 9634} = 0.727129$$

Hasil normalisasi dari data latih ditunjukkan oleh Tabel 6.3 dan hasil normalisasi data uji ditunjukkan oleh Tabel 6.4.

Tabel 7.3 Hasil Normalisasi Data Latih

No	Tgl/Bln/Thn	X1	X2	X3	X4	Y
1	9 Juli 2015	0.727130	0.730664	0.725952	0.725756	0.726934
2	10 Juli 2015	0.730664	0.725952	0.725756	0.726934	0.722811
3	11 Juli 2015	0.725952	0.725756	0.726934	0.722811	0.722811
4	12 Juli 2015	0.725756	0.726934	0.722811	0.722811	0.722811
5	13 Juli 2015	0.726934	0.722811	0.722811	0.722811	0.730075

Tabel 7.4 Hasil Normalisasi Data Uji

No	Tgl/Bln/Thn	X1	X2	X3	X4	Y
1	14 Juli 2015	0.722811	0.722811	0.722811	0.730075	0.722222
2	15 Juli 2015	0.722811	0.722811	0.730075	0.722222	0.728700
3	16 Juli 2015	0.722811	0.730075	0.722222	0.728700	0.728897

### Proses Perhitungan Jarak Data

Setelah dilakukan proses normalisasi maka langkah selanjutnya adalah menghitung jarak dari setiap data. Berikut merupakan contoh perhitungan jarak data 1 terhadap data 2.

Tabel 7.5 Data 1 dan 2

No	Tgl/Bln/Thn	X1	X2	X3	X4
1	9 Juli 2015	0.72712996	0.73066353	0.7259521	0.72575579
2	10 Juli 2015	0.73066353	0.7259521	0.72575579	0.72693365

$$\begin{aligned}x_{1,2} &= \|x_1 - x_2\|^2 = (0.7271299 - 0.7306635)^2 + (0.7306635 - 0.7259521)^2 \\&+ (0.7259521 - 0.7257557)^2 + (0.7257557 - 0.7269336)^2 \\&= 3.61095 \times 10^{-5}\end{aligned}$$

Keterangan :

X<sub>1</sub>, X<sub>2</sub>, X<sub>3</sub>, X<sub>4</sub> → menyatakan fitur data atau pola-pola data, sedangkan x<sub>1</sub> dan x<sub>2</sub> menyatakan data ke-1 dan data ke-2.

Hasil perhitungan jarak dari setiap data latih ditunjukan oleh Tabel 6.6.

Tabel 7.6 Jarak Data Latih

Data ke- <i>i</i>	1	2	...	5
1	0	3.61095x10 <sup>-5</sup>	...	8.02348 x10 <sup>-5</sup>
2	3.61095 x10 <sup>-5</sup>	0	...	4.94435 x10 <sup>-5</sup>
3	3.51075 x10 <sup>-5</sup>	4.06184 x10 <sup>-5</sup>	...	2.66293 x10 <sup>-5</sup>
4	3.43368 x10 <sup>-5</sup>	5.07152 x10 <sup>-5</sup>	...	1.83823 x10 <sup>-5</sup>
5	8.02348 x10 <sup>-5</sup>	4.94435 x10 <sup>-5</sup>	...	0

### Perhitungan Matriks Hessian

Setelah diperoleh jarak dari setiap data langkah selanjutnya adalah menghitung matrik hessian dengan menggunakan persamaan 6.1. berikut merupakan contoh perhitungan matrik hessian untuk data 1 terhadap data 2.

$$[R]_{1,2} = K(x_1, x_2) + \lambda^2 = \exp\left(-\frac{3.61095 \times 10^{-5}}{2 \times (0.7)^2}\right) + 4.32^2 = 19.662363154$$

Perhitungan matrik hessian tersebut terus dilakukan untuk setiap data terhadap data lainnya. Hasil perhitungan matrik hessian untuk seluruh data ditunjukkan pada Tabel 6.7.

Tabel 7.7 Matrik Hessian

R <sub>ij</sub>	1	2	3	4	5
1	19.6624	19.6623632	19.6623642	19.662365	19.6623181
2	19.6623632	19.6624	19.6623586	19.6623483	19.6623495
3	19.6623642	19.6623586	19.6624	19.6623812	19.6623728
4	19.662365	19.6623483	19.6623812	19.6624	19.6623812
5	19.6623181	19.6623495	19.6623728	19.6623812	19.6624

### Hitung Nilai Error, $\delta\alpha_i^*$ dan $\delta\alpha_i$ , serta $\alpha_i$ dan $\alpha_i^*$

Setelah diperoleh nilai matrik hessian maka selanjutnya dihitung nilai Error menggunakan persamaan 6.2. berikut merupakan contoh perhitungan nilai  $E_i$  untuk data 1.

$$E_i = y_i - \sum_{j=1}^l (\alpha_j^* - \alpha_j) R_{ij}$$

$$\begin{aligned} E_1 &= 0.72693 - ((0.0073646 - 0) * 19.6624 - (0.0072136 - 0) * 19.6623 \\ &\quad - (0.0072136 - 0) * 19.6623 - (0.0072136 - 0) * 19.6623 \\ &\quad - (0.0075296 - 0) * 19.6623 \\ &= 0.009548583120820 \end{aligned}$$

Hasil perhitungan  $E_i$  dari seluruh data ditunjukkan pada Tabel 6.8.

Tabel 7.8 Nilai  $E_i$

Data	$E_i$
1	0.009548583
2	0.005426086
3	0.005426086
4	0.005426086
5	0.012689533

Setelah diperoleh nilai  $E_i$  maka dilakukan perhitungan  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  dengan persamaan 6.3 dan 6.4. Berikut contoh perhitungan nilai  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  untuk data 1.

$$\delta\alpha_i^* = \min \left\{ \max \left\{ \gamma(E_i - \varepsilon), -\alpha_i^* \right\}, C - \alpha_i^* \right\}$$

$$\begin{aligned} \delta\alpha_1^* &= (\min(\max(0.00406 * (0.0095485 - 0.0004) \\ &\quad , 0.0073646), 100 - 0.0073646)) \\ &= 0.000037222650829 \end{aligned}$$

$$\delta\alpha_i = \min \left\{ \max \left\{ \gamma(-E_i - \varepsilon), -\alpha_i \right\}, C - \alpha_i \right\}$$

$$\delta\alpha_1 = (\min(\max(0.00406 * (0.0095485 - 0.0004), -0), 100 - 0)) = 0$$

Hasil perhitungan  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  untuk seluruh data latih ditunjukkan oleh Tabel 6.9.

Tabel 7.9 Nilai  $\delta\alpha_i^*$  dan  $\delta\alpha_i$

Data	$\delta\alpha_i^*$	$\delta\alpha_i$

<b>1</b>	<b>3.7223E-05</b>	0
<b>2</b>	<b>2.045E-05</b>	0
<b>3</b>	<b>2.045E-05</b>	0
<b>4</b>	<b>2.045E-05</b>	0
<b>5</b>	<b>5.0002E-05</b>	0

Setelah diperoleh nilai  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  maka dilakukan perhitungan terhadap  $\alpha_i$  dan  $\alpha_i^*$  dengan menggunakan persamaan 6.5 dan 6.6. Berikut merupakan contoh perhitungan  $\alpha_i$  dan  $\alpha_i^*$  untuk data 1.

$$\alpha_i^* = \alpha_i^* + \delta\alpha_i^*$$

$$\alpha_1^* = 0.0073646102 + 0.0000372226 = 0.0074018329$$

$$\alpha_i = \alpha_i + \delta\alpha_i$$

$$\alpha_1 = 0 + 0 = 0$$

Hasil perhitungan  $\alpha_i$  dan  $\alpha_i^*$  untuk seluruh data latih ditunjukkan oleh Tabel 6.10.

Tabel 7.10 Nilai  $\alpha_i^*$  dan  $\alpha_i$

Data	$\alpha_i^*$	$\alpha_i$
<b>1</b>	<b>0.00740183</b>	0
<b>2</b>	<b>0.0072341</b>	0
<b>3</b>	<b>0.0072341</b>	0
<b>4</b>	<b>0.0072341</b>	0
<b>5</b>	<b>0.00752963</b>	0

### Testing Data Latih Dan Data Uji

Dengan menggunakan fungsi peramalan pada persamaan 6.10 diperoleh nilai  $f(x)$  untuk data latih ke 1 adalah sebagai berikut:

$$f(x) = \sum_{i=1}^l (\alpha_i^* - \alpha_i)(K(x_i, x) + \lambda^2)$$

$$\begin{aligned}f(x) &= ((0.00740183290 - (0)) \times 19.6624) \\&+ ((0.00723410171 - (0)) \times 19.662363) \\&+ ((0.00723410171 - (0)) \times 19.662364) \\&+ ((0.00723410171 - (0)) \times 19.662364) \\&+ ((0.00752962809 - (0)) \times 19.662318) \\&= 0.720306367977400\end{aligned}$$

Hasil perhitungan nilai  $f(x)$  untuk seluruh data latih ditunjukan oleh Tabel 6.11.

Tabel 7.11 Nilai  $f(x)$  Data Latih

Data Latih ke- <i>i</i>	Nilai Aktual Kurs	F(x)
1	0.726933647	0.72030637
2	0.72281115	0.72030644
3	0.72281115	0.72030686
4	0.72281115	0.72030685
5	0.730074598	0.72030646

Hasil perhitungan nilai  $f(x)$  untuk seluruh data uji ditunjukan oleh Tabel 6.12.

Tabel 7.12 Nilai  $f(x)$  Data Uji

Data Uji ke- <i>i</i>	Nilai Aktual Kurs	F(x)
1	0.722222222	0.7203045
2	0.728700432	0.72030434
3	0.728896741	0.72030458

### Denormalisasi Peramalan

$$x' = \frac{(x - x_{\min})}{x_{\max} - x_{\min}}$$

$$x = x'(x_{\max} - x_{\min}) + x_{\min}$$

$$x = (0.720306367977 \times (14728 - 9634)) + 9634 = 13303.24057$$

Hasil denormalisasi untuk seluruh data latih ditunjukan oleh Tabel 6.13.

Tabel 7.13 Hasil Denormalisasi Data Latih

Data Latih ke- <i>i</i>	F(x)	F(x) Denormalisasi	Nilai Aktual
1	0.720306368	13303.24057	13337
2	0.720306437	13303.24309	13316
3	0.720306858	13303.24396	13316
4	0.720306853	13303.24388	13316
5	0.72030646	13303.2422	13353

Hasil denormalisasi untuk seluruh data latih ditunjukan oleh Tabel 6.14.

Tabel 7.14 Hasil Denormalisasi Data Uji

Data Uji ke- <i>i</i>	F(x)	F(x) Denormalisasi	Nilai Aktual
1	0.720304504	13303.23114	13304
2	0.720304342	13303.23032	13304
3	0.720304581	13303.23153	13309

### MAPE

Menghitung Mape dengan menggunakan persamaan 6.10, sehingga diperoleh nilai Mape sebagai berikut:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \times 100 \right|$$

$$\begin{aligned} MAPE_{\text{data latih}} &= \frac{1}{5} \left( \left| \frac{13303.24057 - 13337}{13337} \times 100 \right| + \left| \frac{13303.24309 - 13316}{13316} \times 100 \right| \right. \\ &\quad + \left| \frac{13303.24396 - 13316}{13316} \times 100 \right| + \left| \frac{13303.24388 - 13316}{13316} \times 100 \right| \\ &\quad \left. + \left| \frac{13303.2422 - 13353}{13353} \times 100 \right| \right) \\ &= 0.182630337 \end{aligned}$$

Hasil perhitungan nilai Mape untuk data latih dan data uji dapat dilihat pada Tabel 6.15.

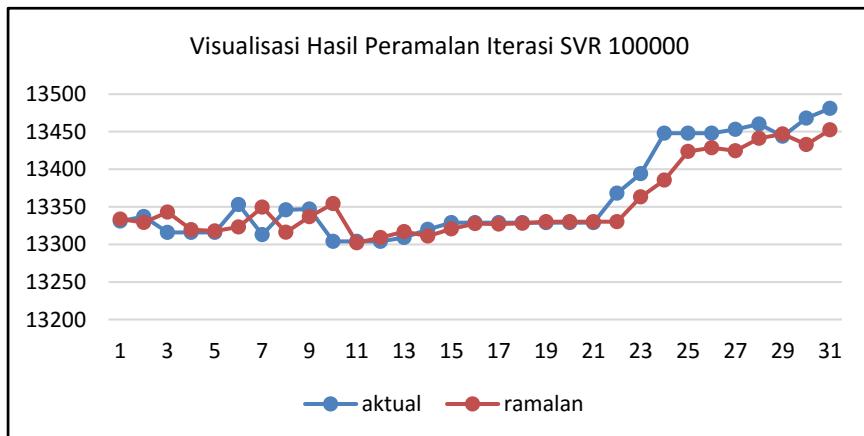
Tabel 7.15 Nilai MAPE

NILAI	Data Latih	Data Uji
	0.253126116	0.073378327
	0.095801356	0.320468153

	0.095794862	0.327927367
	0.095795468	
	0.372633884	
MAPE	0.182630337	0.240591282

### Contoh Grafik Hasil Peramalan Nilai Kurs

MAPE = 0.127551, berikut contoh grafik hasil peramalan nilai kurs yang ditunjukkan oleh Gambar 6.2.



Gambar 7.2 Visualisasi Hasil Peramalan Iterasi SVR 100000

## 7.5 Penyelesaian ACO-SVR

Berikut detail langkah-langkah SVR-ACO (Lixing, 2010), yaitu menyelesaikan optimasi nilai parameter SVR:

1. Inisialisasi:
  - a. Tentukan parameter SVR dan batas minimum-maksimum parameter yang akan dioptimasi dengan ACO, misal  $\sigma=[0,9.999]$ ,  $C=[0,9999]$  dan  $\epsilon=[0,9.999]$   
(Misal, panjang digit masing **parameter  $\sigma$ ,  $C$ ,  $\epsilon$**  diset sebesar 4)

Semut ke-k	Kota yang dikunjungi sebagai digit					Memori
	kota 1	kota 2	kota 3	..	kota 12	
1						[ ]
..						[ ]
5						[ ]

• kota 1 sampai 4 ( $\sigma$ )  
• kota 5 sampai 8 ( $C$ )  
• kota 9 sampai 12 ( $\epsilon$ )

- b. Set nilai Tetapan siklus-semut ( $q_0 = 0.5$ ), Tetapan pengendali intensitas jejak feromon semut ( $\alpha = 0.1$ ), Tetapan pengendali visibilitas ( $\beta = 0.9$ ), Tetapan penguapan jejak feromon local dan global dari semut ( $\rho = 0.9$ , dan  $\delta = 0.7$ ), banyak semut( $m = 5$ ), banyak iterasi ( $Nc_{max} = 1$ ), nilai feromon awal ( $\tau_0 = 0.5$  ).

$$\boxed{\begin{array}{l} k = 1, 2, \dots, m \\ i, j = 1, 2, \dots, n \text{ (banyak digit)} \end{array}}$$

2. Penyusunan jalur kunjungan setiap semut ( $t=1$ ):

Optimasi untuk parameter  $\sigma$ ,  $C$ ,  $\varepsilon$ :

- a. Tentukan digit pertama **secara acak** sebagai nilai  $r=[0,9]$  untuk setiap semut, misal untuk semut ke-( $k=1$ ) dengan  $r = 7$   
b. Tentukan digit berikutnya (nilai  $u$ ), dengan random  $q = [0,1]$  *uniformly distributed*, lalu bandingkan dengan  $q_0$ .

- Jika  $q \leq q_0$

$$P(r, u) = [\tau(r, u)]^\alpha \cdot [\tau(r, u)]^\beta, \quad u \in J(r)$$

$$u = \arg \max_{u \in J(r)} \left\{ [\tau(r, u)]^\alpha \cdot [\tau(r, u)]^\beta \right\}$$

- Jika  $q > q_0$ ,

- Jika  $s \in J(r)$ , maka

$$P(r, s) = \frac{[\tau(r, s)]^\alpha \cdot [\tau(r, s)]^\beta}{\sum_{s \in J(r)} [\tau(r, s)]^\alpha \cdot [\tau(r, s)]^\beta}$$

- Jika  $s \notin J(r)$ , maka

$$P(r, s) = 0$$

lalu, gunakan seleksi Roulette Wheel untuk menentukan  $u=s$ .

Keterangan: Pada kondisi jika  $q > q_0$ , strategi pemilihan yang digunakan di atas disebut dengan 'roulette wheel (RW)' karena mekanismenya adalah simulasi pengoperasian roda roulette. **Setiap kota (s) memiliki persentase di roda roulette** dan semakin besar lebar slot (setara dengan nilai persentase) di roda, maka probabilitas memilih kota tersebut juga semakin besar. Dimana  $s$  adalah variabel acak yang dipilih sesuai dengan distribusi probabilitas dengan RW.

misal  $q = 0.5$ ,  $q_0 = 0.5$ . Maka memenuhi kondisi  $q \leq q_0$ , maka langkah selanjutnya adalah menghitung nilai *argmax*, contoh menghitung  $P(7,5)$ .

$$P(r,u) = [\tau(r,u)]^\alpha [\tau(r,u)]^\beta, u \in J(r)$$

$$u = \arg \max_{u \in J(r)} \{[\tau(r,u)]^\alpha [\tau(r,u)]^\beta\}$$

$$u = \arg \max \{([\tau(7,0)]^{0.1} [\tau(7,0)]^{0.9}), ([\tau(7,1)]^{0.1} [\tau(7,1)]^{0.9}), \dots, ([\tau(7,9)]^{0.1} [\tau(7,9)]^{0.9})\}$$

karena semua nilai  $P(r,u)$ -nya sama, maka sebaiknya dipilih secara random, misal nilai  $u$  yang didapat adalah 5.

- Update feromon local  Setiap semut ketika mengunjungi kota berikutnya, wajib update feromon secara local.
- $$\tau(r,u) = (1 - \rho)(\tau(r,u)) + \rho\tau_0$$
- $$\tau(7,5) = (1 - 0.9) \times 0.5 + (0.9 \times 0.5) = 0.5$$

Semut ke-k	Kota yang dikunjungi sebagai digit					Memori (12 digit)
	kota 1	kota 2	kota 3	...	kota 12	
1	7	5	..	..	..	[7 5 .. .. .. .. .. .. .. .. .. .. .. ..]
2						[.. .. .. .. .. .. .. .. .. .. .. .. .. ..]
3						[.. .. .. .. .. .. .. .. .. .. .. .. .. ..]
4						[.. .. .. .. .. .. .. .. .. .. .. .. .. ..]
5						[.. .. .. .. .. .. .. .. .. .. .. .. .. ..]

- kota 1 sampai 4 atau digit 1 sampai 4 pada memori, merepresentasikan nilai  $\sigma$
- kota 5 sampai 8 atau digit 5 sampai 8 pada memori, merepresentasikan nilai  $C$
- kota 9 sampai 12 atau digit 9 sampai 12 pada memori, merepresentasikan nilai  $\epsilon$

(misalkan semua semut sudah mengunjungi sampai 12 kota), maka akan tampil seperti berikut:

Semut ke-k	Kota yang dikunjungi sebagai digit					Memori (12 digit)
	kota 1	kota 2	kota 3	...	kota 12	
1	7	5	4	..	9	[7 5 4 8 0 0 6 9 1 7 5 9]
2	1	3	9	..	4	[1 3 9 4 1 5 4 3 8 1 0 4]
3	2	2	4	..	2	[2 2 4 0 0 0 0 1 4 7 1 2]
4	9	0	3	..	8	[9 0 3 2 4 3 7 2 5 8 8 8]
5	6	2	1	..	5	[6 2 1 1 7 4 1 0 9 6 2 5]

- kota 1 sampai 4 atau digit 1 sampai 4 pada memori, merepresentasikan nilai  $\sigma$
- kota 5 sampai 8 atau digit 5 sampai 8 pada memori, merepresentasikan nilai  $C$
- kota 9 sampai 12 atau digit 9 sampai 12 pada memori, merepresentasikan nilai  $\epsilon$

Setelah semua kota dikunjungi, hitung *MAPE*, lalu simpan solusi terbaik (*global best*)

Semut ke-k	Hasil konversi			MAPE (misal didapat nilai MAPE berikut)
	$\sigma$	$C$	$\epsilon$	
1	7.548	69	1.759	20.29
2	1.394	1543	8.104	23.60
3	2.240	1	4.712	0.01
4	9.032	4372	5.888	5.54
5	6.211	7410	9.625	0.59

Batas bawah dan atas parameter  $\sigma=[0,9.999]$ ,  $C=[0,9999]$  dan  $\epsilon=[0,9.999]$

3. Cek kondisi berhenti ( $t = t + 1 = 1 + 1 = 2$  (masih  $\leq NC_{max}$ )), jika belum memenuhi kondisi berhenti:
  - a. Update Pheromone Global
$$\tau(r,u) = (1 - \delta)\tau(r,u) + \delta \cdot \Delta\tau(r,u)$$
dimana ada kondisi:
    - Jika  $(r,u) \in \text{global best tour}$ , maka
$$\Delta\tau(r,u) = \frac{1}{L_{globalbest}} = MAPE_{globalbest}$$
    - Jika  $(r,u) \notin \text{global best tour}$ , maka
$$\Delta\tau(r,u) = 0$$
  - b. Kosongkan tabel jalur kunjungan setiap semut
  - c. Kembali ke langkah (2)

## 7.6 Tugas Kelompok

1. Jelaskan pengertian dari istilah-istilah berikut dari sudut pandang topik lanjut pada Swarm Intelligence yang menggunakan lebih dari satu algoritma!
  - a. Compare
  - b. Parallel
  - c. Hybrid
  - d. Kombinasi
  - e. Integrasi
  - f. Improve
  - g. Enhanced
  - h. dari point (a) sampai (g), carilah paper internasional/ nasional dengan judul yang menggunakan kata-kata tersebut masing-masing 1 paper.
2. Jelaskan mengapa beberapa parameter dari algoritma SVR sebaiknya dioptimasi menggunakan algoritma Ant Colony Optimization (ACO)!

3. Berdasarkan dari slide ke-22, jika diset panjang digit masing parameter  $\sigma$ ,  $C$ ,  $\varepsilon$  berturut-turut sebesar 2, 3, 3. Lakukan proses penentuan optimasi parameter SVR tersebut dengan ACO
  - a. Tentukan batas bawah dan batas atas dari  $\sigma=[0,\dots]$ ,  $C=[0,\dots]$  dan  $\varepsilon=[0,\dots]$
  - b. Tentukan semut yang memiliki MAPE terbaik, jika diketahui Set nilai Tetapan siklus-semut ( $q_0 = 0.5$ ), Tetapan pengendali intensitas jejak feromon semut ( $\alpha = 0.1$ ), Tetapan pengendali visibilitas ( $\beta = 0.9$ ), Tetapan penguapan jejak feromon local dan global dari semut ( $\rho = 0.9$ , dan  $\delta = 0.7$ ), banyak semut( $m = 5$ ), banyak iterasi ( $N_{cmax} = 1$ ), nilai feromon awal ( ) dan gambarkan path-nya!
4. Dari No. 3, selesaikan optimasi parameter algoritma SVR tersebut menggunakan algoritma real-coded particle swarm optimization (RCPSO)!

## BAB 8 Project Pilihan Swarm Intelligence

### 8.1 Optimasi Penjadwalan Penggeraan Software

*Scheduling* atau penjadwalan merupakan pekerjaan untuk mengalokasikan sumber daya yang terbatas dari suatu pekerjaan untuk mengefisiensikan waktu penggeraan. *Job Shop Problem* merupakan suatu permasalahan dimana dibutuhkan suatu solusi untuk menyelesaikan suatu *job/task* dengan meminimalkan total *cost* yang dikeluarkan. Pada penelitian ini, terdapat *constraint* yang perlu diperhatikan yaitu ketidakpastian waktu proses penyelesaian suatu job. Ketidakpastian waktu proses penyelesaian tentunya berhubungan erat dengan bidang pekerjaan tertentu seperti *software house*, dsb. Untuk *constraint* pertama berupa masalah ketidakpastian dalam waktu pemrorsesan, dapat diselesaikan menggunakan algoritma kombinatorial *Artificial Bee Colony* (ABC). Penelitian ini mencoba menerapkan prinsip prinsip tersebut dengan merubah domain permasalahan menjadi penjadwalan penggeraan *software*. Penelitian ini diberi judul Optimasi Penjadwalan Penggeraan Software pada *Software House* dengan *Flow-Shop Problem* Menggunakan *Artificial Bee Colony*, diharapkan penelitian ini dapat digunakan sebagai rujukan dalam penelitian-penelitian lain dengan topik yang berkaitan.

#### 8.1.1 Formulasi Permasalahan

Terdapat sebuah studi kasus permasalahan kombinatorial yaitu penjadwalan penggeraan software dengan tipe *flow-shop scheduling problem* (FSP) menggunakan ABC. Dimana akan dilakukan optimasi untuk mendapatkan nilai *makespan* (waktu pergeraan tercepat) dari keseluruhan projek dengan waktu tahap penggeraan tertentu yang berbeda dengan satu sama lain. Proses optimasi FSP dengan ABC adalah berikut:

1. Inisialisasi bee awal pada fase *employeed bee*
2. Evaluasi nilai *fitness* keseluruhan bee dalam populasi
3. Melakukan *improvement* terhadap *employeed bee* dengan *neighborhood operator*:
  - a) *Improvement* dengan *swap operator* (SO)
  - b) *Improvement* dengan *swap sequence* (SS)

4. Melakukan seleksi *roulette wheel selection* untuk membentuk *bee* fase *onlooker bee*
5. Melakukan *improvement* terhadap *onlooker bee* dengan *neighborhood operator*:
  - a) *Improvement* dengan *insert operator* (IO)
  - b) *Improvement* dengan *insert sequence* (IS)
6. Melakukan evaluasi fase *scout bee* untuk mendapatkan *bee* dengan *fitness* terbaik (global best)
7. Melakukan pengecekan nilai *trial* terhadap *limit* guna membentuk *bee* baru untuk fase *employeed bee* pada iterasi selanjutnya

Sebagai contoh dalam studi kasus ini, digunakan data pengerjaan projek yang ditunjukkan pada Tabel 2.3:

Tabel 8.1 Data yang Digunakan

Projek	Task			
	1	2	3	4
Projek 1	5	5	2	3
Projek 2	5	4	2	4
Projek 3	3	4	2	2
Projek 4	5	2	1	1

Pada tabel 3.3, *Task* merupakan pekerjaan yang perlu dilakukan. Dengan keterangan *task* “1” merupakan Analisis, *task* “2” merupakan Perancangan, *task* “3” merupakan Implementasi, dan *task* “4” merupakan Pengujian. Sedangkan untuk nilai dari projek terhadap *task* merupakan nilai waktu yang dibutuhkan untuk penyelesaian setiap *task*.

Untuk parameter yang digunakan dalam perhitungan adalah jumlah *bee* dalam populasi (*colonySize*) sebesar 3, maksimum iterasi sebesar 1, *size problem* sebesar 4, limit sebesar 5, dan *number of sequence* (Nse) sebesar 2 kali *size problem* yaitu 8. Berikut ini perhitungan manualnya:

1. Inisialisasi *bee* awal pada fase *employeed bee*

Menginisialisasi *bee* awal secara acak sebesar *colonySize* yang telah ditentukan. Dimana *bee* ini merupakan representasi solusi dari permasalahan yaitu urutan pengerjaan dari keseluruhan projek yang ada. Inisialisasi dapat dilihat pada Tabel 2.4:

Tabel 8.2 Inisialisasi *bee* awal pada fase *employeed bee*

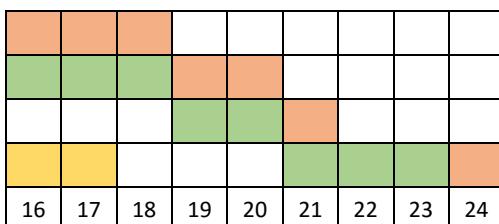
Bee ke-i	Urutan			
x1	2	3	1	4
x2	3	4	1	2
x3	1	2	3	4

2. Evaluasi nilai *fitness* keseluruhan bee dalam populasi

Melakukan perhitungan *fitness* dengan menghitung nilai *makespan* dengan menggunakan *gantt-chart*. Dimana nilai *fitness* terbaik adalah nilai *makespan* dengan total waktu terendah. Gantt-chart untuk bee ke-1 sampai 3 dapat dilihat pada Tabel 2.5, 2.6, dan 2.7.

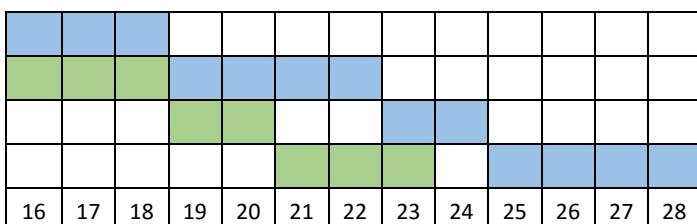
Tabel 8.3 Gantt-chart bee ke-1

Task 1															
Task 2															
Task 3															
Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Tabel 8.4 Gantt-chart bee ke-2

Task 1															
Task 2															
Task 3															
Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Tabel 8.5 Gantt-chart bee ke-3

Task 1															
Task 2															
Task 3															

Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

16	17	18	19	20	21	22	23							

3. Melakukan *improvement* terhadap *employeed bee* dengan *neighborhood operator*.

- a) *Improvement* dengan *swap operator* (SO)

Melakukan SO terhadap setiap bee dalam populasi, dengan indeks SO yang ditentukan secara random. Perlakuan SO dapat dilihat pada Tabel 2.8 dan *gantt-chart* bee dengan nilai *fitness* terbaik pada Tabel 2.9:

Tabel 8.6 *Improvement* SO pada tahap *employeed bee*

Bee ke-i	Urutan				Swap Operator		Urutan				Makespan	
x1	2	3	1	4	4	1	4	3	1	2		28
x2	3	4	1	2	1	3	1	4	3	2		28
x3	1	2	3	4	2	4	1	4	3	2		28

Tabel 8.7 Gantt-chart bee dengan fitness terbaik

Task 1															
Task 2															
Task 3															
Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

16	17	18	19	20	21	22	23	24	25	26	27	28	

*Gantt-chart* yang ditampilkan adalah milik bee ke-1. Dikarenakan terdapat *makespan* yang sama nilainya terhadap bee lainnya, maka bee pertama yang ditampilkan. Maka didapatkan hasil bee baru yang dapat dilihat pada Tabel 2.10:

Tabel 8.8 Hasil *improvement SO* pada tahap *employeed bee*

Bee ke-i	Urutan				Makespan	Trial
x1	4	3	1	2	28	1
x2	1	4	3	2	28	1
x3	1	4	3	2	28	1

Nilai *trial* adalah nilai percobaan untuk mendapatkan bee baru dengan nilai *fitness* yang lebih baik, dimana jika didapatkan nilai *fitness* yang tidak lebih baik maka nilai *trial* ditambahkan dengan 1, dan jika didapatkan nilai *fitness* yang lebih baik maka nilai *trial* di-set dengan 0.

- b) *Improvement* dengan *swap sequence* (SS)

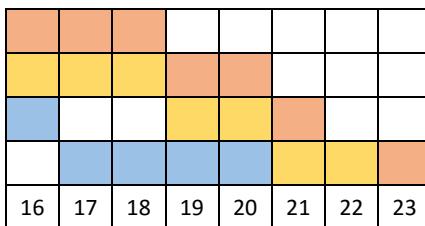
Melakukan SS terhadap setiap bee hasil *improvement SO*, dengan indeks SO yang ditentukan secara random. Perlakuan SS dapat dilihat pada Tabel 2.11, 2.13, 2.15 dan *gantt-chart* bee dengan nilai *fitness* terbaik pada Tabel 2.12, 2.14, 2.16:

Tabel 8.9 *Improvement SS* pada tahap bee ke-1

Bee ke-i	Urutan				Makespan	Trial
x1	4	3	1	2	28	1
SO	Urutan				Makespan	Trial
3	4	4	2	1	28	2
1	3	1	2	4	26	0
4	3	1	2	3	23	0
3	2	1	3	2	25	1
3	1	2	3	1	24	2
1	2	3	2	1	24	3
4	1	4	2	1	28	4
2	4	4	3	1	28	5
Terbaik	1	2	3	4	23	

Tabel 8.10 Gantt-chart SS bee ke-1 dengan *fitness* terbaik

Task 1															
Task 2															
Task 3															
Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Tabel 8.11 Improvement SS pada tahap bee ke-2

Bee ke-i		Urutan				Makespan	Trial
x2		1	4	3	2	28	1
SO		Urutan				Makespan	Trial
2	3	1	3	4	2	28	2
1	4	2	3	4	1	28	3
4	2	2	1	4	3	26	0
3	2	2	4	1	3	28	1
1	2	4	2	1	3	28	2
4	3	4	2	3	1	28	3
1	3	3	2	4	1	28	4
2	3	3	4	2	1	28	5
Terbaik		2	1	4	3	26	

Tabel 8.12 Gantt-chart SS bee ke-2 dengan *fitness* terbaik

Task 1															
Task 2															
Task 3															
Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

16	17		18	19	20	21	22	23	24	25	26														

Tabel 8.13 *Improvement SS* pada tahap bee ke-3

Bee ke-i		Urutan				Makespan		Trial
x3		1	4	3	2	28		1
SO		Urutan				Makespan		Trial
3		1	3	4	1	28		2
2		4	3	2	1	24		0
1		2	2	3	1	24		1
4		2	2	4	1	28		2
1		2	4	2	1	28		3
3		4	4	2	3	28		4
2		3	4	3	2	28		5
4		1	1	3	2	25		6
Terbaik		3	2	1	4	24		

Tabel 8.14 *Gantt-chart SS* bee ke-3 dengan *fitness* terbaik

Task 1																									
Task 2																									
Task 3																									
Task 4																									
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15										

1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	
6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1

Maka, didapatkan bee baru hasil employeed bee yang dapat dilihat pada Tabel 2.17:

Tabel 8.15 Hasil *improvement* SS pada tahap *employeed bee*

Bee ke-i	Urutan				Makespan	Trial
x1	1	2	3	4	23	5
x2	2	1	4	3	26	5
x3	3	2	1	4	24	6

4. Melakukan seleksi *roulette wheel selection* (RWS) untuk membentuk bee fase *onlooker bee*

Bee baru hasil SS dihitung nilai probabilitasnya dengan rumus:

$$prob_i = \frac{fitness_i}{totalFitness}$$

Hasil nilai probabilitas dapat dilihat pada Tabel 2.18:

Tabel 8.16 Nilai probabilitas keseluruhan *employeed bee* hasil SS

Bee ke-i	Urutan				Makespan	Trial	Prob
x1	1	2	3	4	23	5	0.315068493
x2	2	1	4	3	26	5	0.356164384
x3	3	2	1	4	24	6	0.328767123

Dengan contoh perhitungan probabilitas untuk bee ke-1 adalah sebagai berikut:

$$prob_1 = \frac{23}{73} = 0.315068493$$

Kemudian menghitung nilai probabilitas kumulatif dengan rumus:

$$probCum_i = \sum_{i=1}^i prob_i$$

Lalu mengenerate nilai r dengan batas  $r[0, 1]$  untuk setiap bee, dan memilih serta cek *probCum* dari bee ke 1 hingga akhir, dengan kondisi jika memenuhi  $r \leq probCum_i$  maka bee ke  $i$  tersebut terpilih untuk masuk ke fase *onlooker bee*. Proses randomisasi dan pengecekan ini dilakukan hingga terbentuk bee baru sebanyak inisialisasi *colonySize*. Hasil dari perhitungan *probCum* dan pengecekan kondisi r dapat dilihat pada Tabel 2.19:

Tabel 8.17 Nilai probabilitas kumulatif dan pengecekan kondisi r

Bee ke-i	Urutan				Makespan	Prob	ProbCum	Rand	Terpilih
x1	1	2	3	4	23	0.3151	0.3151	0.2	1
x2	2	1	4	3	26	0.3562	0.6712	0.45	2
x3	3	2	1	4	24	0.3288	1	0.33	2

Contoh perhitungan probabilitas kumulatif untuk bee ke-2:

$$probCum_2 = 0.3151 + 0.3562 = 0.6712$$

Dan bee baru hasil seleksi RWS yang akan digunakan pada fase *onlooker bee* dapat dilihat pada Tabel 2.20:

Tabel 8.18 Bee baru hasil seleksi RWS

Bee ke-i	Urutan				Makespan
x1	1	2	3	4	23
x2	2	1	4	3	26
x3	2	1	4	3	26

5. Melakukan *improvement* terhadap *onlooker bee* dengan *neighborhood operator*:

- a) *Improvement* dengan *insert operator* (IO)

Melakukan IO terhadap setiap bee hasil seleksi RWS, dengan indeks IO yang ditentukan secara random. Perlakuan IO dapat dilihat pada Tabel 2.21 dan *gantt-chart* bee dengan nilai *fitness* terbaik pada Tabel 2.22:

Tabel 8.19 *Improvement* IO pada tahap *onlooker bee*

Bee ke-i	Urutan				Insert Operator		Urutan				Makespan
x1	1	2	3	4	2	1	2	1	3	4	24
x2	2	1	4	3	1	3	4	2	1	3	28
x3	2	1	4	3	2	3	2	4	1	3	28

Tabel 8.20 Gantt-chart bee dengan fitness terbaik

Task 1													
Task 2													
Task 3													
Task 4													
Time	1	2	3	4	5	6	7	8	9	10	11	12	13
													14
													15

16	17	18	19	20	21	22	23	24

Maka didapatkan hasil bee baru yang dapat dilihat pada Tabel 2.23:

Tabel 8.21 Hasil *improvement* IO pada tahap *onlooker bee*

Bee ke-i	Urutan				Makespan	Trial
x1	2	1	3	4	24	6
x2	4	2	1	3	28	6
x3	2	4	1	3	28	7

- b) *Improvement* dengan *insert sequence* (IS)

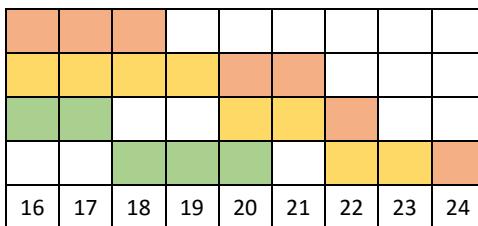
Melakukan IS terhadap setiap bee hasil *improvement* IO, dengan indeks IO yang ditentukan secara random. Perlakuan IS dapat dilihat pada Tabel 2.24, 2.26, 2.28 dan *gantt-chart* bee dengan nilai *fitness* terbaik pada Tabel 2.25, 2.27, 2.29:

Tabel 8.22 *Improvement* IS pada tahap bee ke-1

Bee ke-i	Urutan				Makespan	Trial
x1	2	1	3	4	24	6
IO	Urutan				Makespan	Trial
1	3	3	2	1	4	24
2	4	3	4	2	1	28
3	4	3	4	1	2	28
1	4	2	3	4	1	28
4	3	2	3	1	4	24
1	2	3	2	1	4	24
3	4	3	2	4	1	28
4	2	3	1	2	4	24
Terbaik	2	1	3	4	24	14

Tabel 8.23 Gantt-chart IS bee ke-1 dengan fitness terbaik

Task 1															
Task 2															
Task 3															
Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Tabel 8.24 Improvement IS pada tahap bee ke-2

Bee ke-i		Urutan				Makespan	Trial
x2		4	2	1	3	28	6
IO		Urutan				Makespan	Trial
2	3	4	1	2	3	27	0
3	1	2	4	1	3	28	1
4	1	3	2	4	1	28	2
2	4	3	1	2	4	24	0
4	2	3	4	1	2	28	1
3	2	3	1	4	2	28	2
1	2	1	3	4	2	28	3
1	3	4	1	3	2	29	4
Terbaik		3	1	2	4	24	

Tabel 8.25 Gantt-chart IS bee ke-2 dengan fitness terbaik

Task 1															
Task 2															
Task 3															
Task 4															
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

16	17	18	19	20	21	22	23	24	

Tabel 8.26 *Improvement IS* pada tahap bee ke-3

Bee ke-i		Urutan				Makespan		Trial
x3		2	4	1	3	28		7
IO		Urutan				Makespan		Trial
1	3	1	2	4	3	26		0
2	3	1	4	2	3	27		1
4	2	1	3	4	2	28		2
1	4	2	1	3	4	24		0
1	2	1	2	3	4	23		0
1	3	3	1	2	4	24		1
2	1	1	3	2	4	25		2
3	4	1	3	4	2	28		3
Terbaik		1	2	3	4	23		

Tabel 8.27 Gantt-chart IS bee ke-3 dengan fitness terbaik

Task 1														
Task 2														
Task 3														
Task 4														
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14
														15

16	17	18	19	20	21	22	23		

Maka, didapatkan bee baru hasil *onlooker bee* yang dapat dilihat pada Tabel 2.30:

Tabel 8.28 Hasil *improvement IS* pada tahap *onlooker bee*

Bee ke-i	Urutan				Makespan	Trial
x1	2	1	3	4	24	14
x2	3	1	2	4	24	4
x3	1	2	3	4	23	3

6. Melakukan evaluasi fase *scout bee* untuk mendapatkan bee dengan *fitness* terbaik (global best)  
Evaluasi dilakukan dengan memilih bee dari hasil *onlooker bee* yang memiliki nilai *fitness* terbaik. Dimana didapatkan global best yang dapat dilihat pada Tabel 2.31:

Tabel 8.29 Global best terpilih

Bee ke-i	Urutan				Makespan
x3	1	2	3	4	23

7. Melakukan pengecekan nilai *trial* terhadap *limit* guna membentuk bee baru untuk fase *employeed bee* pada iterasi selanjutnya

Dilakukan dengan membandingkan bee hasil inisialisasi (Tabel 2.32) dengan bee hasil *improvement* terakhir (Tabel 2.33) dan mengecek kondisi jika *trial* maksimum (M) pada populasi bee paling akhir melebihi *limit* yang telah ditentukan ( $M > Limit$ ), maka bee yang tidak mengalami perbaikan dihapus dan digantikan dengan bee baru secara random seperti randomisasi pada fase inisialisasi, serta nilai *trial*-nya di-reset sama dengan 0. Jika  $M > Limit$  dan terdapat bee yang mengalami perbaikan maka bee tersebut tidak perlu digantikan dengan bee baru serta nilai *trial*-nya di-reset sama dengan 0.

Namun, jika  $M < Limit$  maka keseluruhan bee tidak perlu diganti dan nilai *trial*-nya tidak perlu di-reset. Dimana untuk studi kasus ini hasil pengecekannya dapat dilihat pada Tabel 2.34:

Tabel 8.30 Bee hasil inisialisasi awal

Bee ke-i	Urutan				Makespan
x1	2	3	1	4	24
x2	3	4	1	2	28
x3	1	2	3	4	23

Tabel 8.31 Bee hasil *improvement* terakhir

Bee ke-i	Urutan				Makespan	Trial	Perbaikan
x1	2	1	3	4	24	14	Tidak
x2	3	1	2	4	24	4	Ada
x3	1	2	3	4	23	3	Tidak

Dimana nilai  $M = 14$ , dan nilai *limit* yang telah ditentukan diawal adalah 5. Maka,  $M > Limit$  serta terdapat bee yang mengalami perbaikan dan terdapat juga bee yang tidak mengalami perbaikan. Sehingga bee ke-1 dan 3 dirandom ulang dan bee ke-2 dibiarkan tetap, lalu ketiga nilai *trial*-nya di-reset menjadi 0.

Tabel 8.32 Bee baru untuk iterasi selanjutnya

Bee ke-i	Urutan				Trial
x1	1	3	2	4	0
x2	3	1	2	4	0
x3	3	2	1	4	0

Dengan maksimum iterasi sama dengan 1 seperti yang telah ditentukan di awal, maka perulangan diberhentikan.

## 8.1.2 Implementasi

Berikut ini diberikan bagian kode program JAVA untuk implementasi Optimasi Penjadwalan Penggerjaan Software pada Software House dengan Flow-Shop Problem Menggunakan Artificial Bee Colony.

Source Code 8.1 FSPABC

```
Main.java

package fspabc;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import jxl.Sheet;
```

```
import jxl.Workbook;

/**
 *
 * Thanks to @author Daneswara Jauhari
 */
public class Main extends javax.swing.JFrame {

    private JFileChooser chooser = new JFileChooser();
    private int colonySize;
    private int maksIterasi;
    private int sizeProblem;
    private int limit;
    private int nse;
    private int project;
    private int[][] beeColony;
    int[] task = {0, 0, 0, 0, 0};
    int data[][][];

    /**
     * Creates new form Main
     */
    public Main() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of
     * this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
    // Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        _colonySize = new javax.swing.JTextField();
        _maksIterasi = new javax.swing.JTextField();
        _limit = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        table = new javax.swing.JTable();
        jButton1 = new javax.swing.JButton();
        lokasi = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Colony Size");

        jLabel2.setText("Maks Iterasi");
    }
}
```

```
        jLabel3.setText("Limit");

        table.setModel(new javax.swing.table.DefaultTable-
Model(
            new Object [][] {
                },
                new String [] {
                    "Projek", "Mesin 1", "Mesin 2", "Mesin 3",
"Mesin 4"
                }
            ));
        jScrollPane.setViewportView(table);

        jButton1.setText("Load Data");
        jButton1.addActionListener(new java.awt.event.Ac-
tionListener() {
            public void actionPerformed(java.awt.event.Ac-
tionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel5.setFont(new java.awt.Font("Tahoma", 1,
14)); // NOI18N
        jLabel5.setText("Optimasi Penjadwalan Pengerjaan
Software pada Software House");

        jLabel6.setFont(new java.awt.Font("Tahoma", 1,
14)); // NOI18N
        jLabel6.setText("dengan Flow-Shop Scheduling Prob-
lem Menggunakan Algoritma Bee Colony ");

        jLabel7.setFont(new java.awt.Font("Tahoma", 1,
14)); // NOI18N
        jLabel7.setText("_____
_____");

        jButton2.setText("Proses");
        jButton2.addActionListener(new java.awt.event.Ac-
tionListener() {
            public void actionPerformed(java.awt.event.Ac-
tionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new ja-
vax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(ja-
vax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(jav-
ax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel5)
                    .addComponent(jLabel6)
                    .addComponent(jLabel7)
                    .addComponent(jButton1)
                    .addComponent(jButton2)
                )
            )
        );
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable table;
    // End of variables declaration//GEN-END:variables
}
```



```
        .addCompo-
nent(jLabel1)
        .addPre-
ferredGap(javax.swing.LayoutStyle.ComponentPlace-
ment.RELATED, 34, Short.MAX_VALUE)
        .addCompo-
nent(_colonySize, javax.swing.GroupLayout.PREFERRED_SIZE,
83, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE, ja-
vax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))))))
        .addContainerGap(23, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(ja-
vax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(24, 24, 24)
            .addComponent(jLabel5)
            .addPreferredGap(javax.swing.Lay-
outStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel6)
            .addPreferredGap(javax.swing.Lay-
outStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel7)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(ja-
vax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(lokasi, ja-
vax.swing.GroupLayout.PREFERRED_SIZE, 23, ja-
vax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createSequen-
tialGroup()
                .addComponent(jButton1)
                .addPreferredGap(javax.swing.Lay-
outStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createPar-
allelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1, ja-
vax.swing.GroupLayout.PREFERRED_SIZE, 228, ja-
vax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(layout.createSequen-
tialGroup()
                    .addGroup(layout.createPar-
allelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(_colo-
nysize, javax.swing.GroupLayout.PREFERRED_SIZE, ja-
vax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayoutLayout-
PREFERRED_SIZE)
                        .addComponent(jLabel1))
                    .addGap(16, 16, 16)
                    .addGroup(layout.createPar-
allelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(_maksit-
erasi, javax.swing.GroupLayout.PREFERRED_SIZE, ja-
vax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayoutLayout-
PREFERRED_SIZE)
                        .addComponent(jLabel2))
                    .addGap(18, 18, 18)
                )
            )
        )
    )
)
```

```
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(_limit,
        javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel3))
        .addGap(50, 50, 50)
        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 44, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(59, Short.MAX_VALUE))
    );

    pack();
}// </editor-fold>//GEN-END:initComponents

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
    // TODO add your handling code here:
    int jes = chooser.showOpenDialog(this);

    if (jes == chooser.APPROVE_OPTION) {
        File f = chooser.getSelectedFile();
        lokasi.setText(f.getPath());
        muatData(f.getPath());
    }
}//GEN-LAST:event_jButton1ActionPerformed

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed
    // TODO add your handling code here:
    colonySize = Integer.parseInt(_colonySize.getText());
    nse = colonySize * 2;
    beeColony = iBeeColony();
    maksIterasi = Integer.parseInt(_maksiterasi.getText());
    limit = Integer.parseInt(_limit.getText());
    for (int i = 0; i < maksIterasi; i++) {
        System.out.println("");
        System.out.println("*****Iterasi Ke"
+ (i + 1) + "*****");
        beeColony = ABC(beeColony);
        System.out.println("");
    }
    fitness(beeColony);
}//GEN-LAST:event_jButton2ActionPerformed

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    // <editor-fold defaultstate="collapsed" desc=" Look
    and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not
    available, stay with the default look and feel.
}
```

```
* For details see http://download.oracle.com/java-
vase/tutorial/uiswing/lookandfeel/plaf.html
*/
try {
    for (javax.swing.UIManager.LookAndFeelInfo info
: javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager-
ager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLog-
ger(Main.class.getName()).log(java.util.log-
ging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLog-
ger(Main.class.getName()).log(java.util.log-
ging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLog-
ger(Main.class.getName()).log(java.util.log-
ging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelExcep-
tion ex) {
    java.util.logging.Logger.getLog-
ger(Main.class.getName()).log(java.util.log-
ging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new Main().setVisible(true);
    }
});
}

public static void fitness(int[][][] bee) {
    int indexMakespanMin = 0;
    for (int i = 0; i < bee.length; i++) {
        if (bee[indexMakespanMin][10] < bee[i][10]) {
            indexMakespanMin = i;
        }
    }
    double fitness = (double) 1 / bee[indexMakespan-
Min][10];
    System.out.printf("fitness = %.7f", fitness);
}

public int[][][] iBeeColony() {
    int[][][] initbee = new int[colonySize][sizeProblem +
2];
    System.out.println("[Employeed bee] Inisialisasi
bee :");
    for (int i = 0; i < colonySize; i++) {
```

```
ArrayList<Integer> random = new ArrayList<Integer>();
for (int r = 1; r <= sizeProblem; r++) {
    random.add(new Integer(r));
}
Collections.shuffle(random);
for (int j = 0; j < sizeProblem; j++) {
    initbee[i][j] = (int) random.get(j);
}
initbee[i][sizeProblem] = hitungMakeSpan(initbee[i]);
}
tampilanBeeMakespan(initbee);
return initbee;
}

public int[][] ABC(int[][] beeColony) {
    int temp = 0;
    int tmp[] = new int[2];
    int[][] originalBee = new int[beeColony.length][beeColony[0].length];
    for (int i = 0; i < originalBee.length; i++) {
        for (int j = 0; j < originalBee[i].length; j++)
    {
        originalBee[i][j] = beeColony[i][j];
    }
}
// SO
//
// SS
//
// RWS
//
// IO Onlooker Bee
//
// IS Onlooker Bee
//
// IS Onlooker Bee
//
return beeColony;
}

public void tampilanBeeTerbaik(int[][] bee) {
    int terbaik = 0;
    for (int i = 0; i < bee.length; i++) {
        if (bee[terbaik][10] > bee[i][10]) {
            terbaik = i;
        }
    }
    System.out.println("");
}
```

```
        for (int i = 0; i < bee[terbaik].length; i++) {
            System.out.print(bee[terbaik][i] + " ");
        }
    }

    public void tampilkanHasilRWS(int[][][] bee, double[][][]
rws, int[] choosen) {
        for (int i = 0; i < bee.length; i++) {
            for (int j = 0; j < bee[0].length; j++) {
                System.out.print(bee[i][j] + " ");
            }
            System.out.printf("%.2f %.2f %.2f %d\n",
rws[i][0], rws[i][1], rws[i][2], choosen[i]);
        }
    }

    public void tampilkanBeeDenganTrial(int[][][] bee) {
        for (int i = 0; i < bee.length; i++) {
            for (int j = 0; j < bee[0].length; j++) {
                System.out.print(bee[i][j] + " ");
            }
            System.out.println("");
        }
    }

    public void tampilkanBeeMakespan(int[][][] bee) {
        for (int i = 0; i < bee.length; i++) {
            for (int j = 0; j < bee[0].length - 1; j++) {
                System.out.print(bee[i][j] + " ");
            }
            System.out.println("");
        }
    }

    public int hitungMakeSpan(int[] urutan) {
        int hasil;
        for (int i = 0; i < urutan.length - 2; i++) {
            for (int j = 1; j < data[0].length; j++) {
                if (j - 1 == 0) {
                    task[j - 1] += data[urutan[i] - 1][j -
1];
                } else {
                    if (task[j - 1] > task[j - 2]) {
                        task[j - 1] += data[urutan[i] -
1][j - 1];
                    } else {
                        task[j - 1] = data[urutan[i] - 1][j -
1];
                        task[j - 1] += task[j - 2];
                    }
                }
            }
        }
        hasil = task[3];
        task[0] = 0;
        task[1] = 0;
        task[2] = 0;
        task[3] = 0;
    }
}
```

```
        return hasil;
    }

private void muatData(String lokasi) {
    File excelFile = new File(lokasi);

    // buat model untuk file excel
    if (excelFile.exists()) {
        try {
            Workbook workbook = Workbook.getWork-
book(excelFile);
            Sheet sheet = workbook.getSheets()[0];
            String header[] = {"Projek", "Mesin 1",
"Mesin 2", "Mesin 3", "Mesin 4"};
            DefaultTableModel model = new DefaultTable-
Model(null, header);
            model.setColumnCount(sheet.getColumns());
            model.setRowCount(sheet.getRows());
            data = new
int[sheet.getRows()][sheet.getColumns()];
            sizeProblem = sheet.getRows();
            for (int row = 0; row < sheet.getRows();
row++) {
                for (int column = 0; column <
sheet.getColumns(); column++) {
                    String content = sheet.getCell(col-
umn, row).getContents();
                    data[row][column] = Integer.par-
seInt(sheet.getCell(column, row).getContents());
                    model.setValueAt(content, row, col-
umn);
                }
            }
            table.setModel(model);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Error:
" + e);
        }
    }

} else {
    JOptionPane.showMessageDialog(null, "File does
not exist");
}
}

// Variables declaration - do not modify//GEN-
BEGIN:variables
private javax.swing.JTextField _colonysize;
private javax.swing.JTextField _limit;
private javax.swing.JTextField _maksiterasi;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JScrollPane jScrollPane1;
```

```
private javax.swing.JLabel lokasi;
private javax.swing.JTable table;
// End of variables declaration//GEN-END:variables
}
```

### Main.form

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.5" maxVersion="1.9" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
    <Properties>
        <Property name="defaultCloseOperation" type="int" value="3"/>
    </Properties>
    <SyntheticProperties>
        <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
        <SyntheticProperty name="generateCenter" type="boolean" value="false"/>
    </SyntheticProperties>
    <AuxValues>
        <AuxValue name="FormSettings_autoResourcing" type="java.lang.Integer" value="0"/>
        <AuxValue name="FormSettings_autoSetComponentName" type="java.lang.Boolean" value="false"/>
        <AuxValue name="FormSettings_generateFQN" type="java.lang.Boolean" value="true"/>
        <AuxValue name="FormSettings_generateMnemonicsCode" type="java.lang.Boolean" value="false"/>
        <AuxValue name="FormSettings_i18nAutoMode" type="java.lang.Boolean" value="false"/>
        <AuxValue name="FormSettings_layoutCodeTarget" type="java.lang.Integer" value="1"/>
        <AuxValue name="FormSettings_listenerGenerationStyle" type="java.lang.Integer" value="0"/>
        <AuxValue name="FormSettings_variablesLocal" type="java.lang.Boolean" value="false"/>
        <AuxValue name="FormSettings_variablesModifier" type="java.lang.Integer" value="2"/>
    </AuxValues>

    <Layout>
        <DimensionLayout dim="0">
            <Group type="103" groupAlignment="0" attributes="0">
                <Group type="102" attributes="0">
                    <Group type="103" groupAlignment="0" attributes="0">
                        <Group type="102" alignment="0" attributes="0">
                            <EmptySpace min="-2" pref="84" max="-2" attributes="0"/>
                        <Group type="103" groupAlignment="0" attributes="0">

```

```
<Component id="jLabel6" alignment="1" min="-2" max="-2" attributes="0"/>
<Group type="102" alignment="1" attributes="0">
    <Component id="jLabel5" min="-2" max="-2" attributes="0"/>
        <EmptySpace min="-2" pref="40" max="-2" attributes="0"/>
    </Group>
</Group>
<Group type="102" alignment="0" attributes="0">
    <EmptySpace min="-2" pref="25" max="-2" attributes="0"/>
    <Group type="103" groupAlignment="0" attributes="0">
        <Component id="jLabel7" min="-2" max="-2" attributes="0"/>
        <Group type="103" groupAlignment="0" max="-2" attributes="0">
            <Group type="102" alignment="0" attributes="0">
                <Component id="jButton1" min="-2" max="-2" attributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
                <Component id="lokasi" max="32767" attributes="0"/>
            </Group>
            <Group type="102" alignment="0" attributes="0">
                <Component id="jScrollPane1" min="-2" max="-2" attributes="0"/>
                <EmptySpace type="separate" max="-2" attributes="0"/>
            <Group type="103" groupAlignment="1" max="-2" attributes="0">
                <Group type="102" alignment="1" attributes="0">
                    <Group type="103" groupAlignment="0" attributes="0">
                        <Component id="jLabel3" min="-2" max="-2" attributes="0"/>
                        <Component id="jLabel2" min="-2" max="-2" attributes="0"/>
                        <EmptySpace min="-2" pref="31" max="-2" attributes="0"/>
                    <Group type="103" groupAlignment="1" max="-2" attributes="0">
                        <Component id="_maksiterasi" alignment="1" pref="83" max="32767" attributes="0"/>
                        <Component id="_limit" max="32767" attributes="0"/>
                    </Group>
                </Group>
            </Group>
        </Group>
    </Group>
</Group>
```

```
                <Group type="102"
alignment="1" attributes="0">
                <Component
id="jLabel1" min="-2" max="-2" attributes="0"/>
                <EmptySpace
pref="34" max="32767" attributes="0"/>
                <Component
id="_colonySize" min="-2" pref="83" max="-2" attrib-
utes="0"/>
                </Group>
                <Component id="jBut-
ton2" max="32767" attributes="0"/>
                </Group>
                </Group>
                <Group type="102"
id="jLabel5" min="-2" max="-2" attributes="0"/>
                <EmptySpace min="-2" pref="24" max="-2" at-
tributes="0"/>
                <Component id="jLabel6" min="-2" max="-2" at-
tributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
                <Component id="jLabel7" min="-2" max="-2" at-
tributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
                <Component id="lokasi" alignment="0"
min="-2" pref="23" max="-2" attributes="0"/>
                <Group type="102" alignment="0" attrib-
utes="0">
                    <Component id="jButton1" min="-2"
max="-2" attributes="0"/>
                    <EmptySpace max="-2" attributes="0"/>
                    <Group type="103" groupAlignment="0"
attributes="0">
                        <Component id="jScrollPane1"
min="-2" pref="228" max="-2" attributes="0"/>
                        <Group type="102" alignment="0"
attributes="0">
                            <Group type="103" groupAlign-
ment="3" attributes="0">
                                <Component id="_colo-
nysize" alignment="3" min="-2" max="-2" attributes="0"/>
                                <Component id="jLabel1"
alignment="3" min="-2" max="-2" attributes="0"/>
```

```
</Group>
<EmptySpace min="-2"
pref="16" max="-2" attributes="0"/>
<Group type="103" groupAlignment="3" attributes="0">
<Component id="_maksiterasi" alignment="3" min="-2" max="-2" attributes="0"/>
<Component id="jLabel2" alignment="3" min="-2" max="-2" attributes="0"/>
</Group>
<EmptySpace type="separate" max="-2" attributes="0"/>
<Group type="103" groupAlignment="3" attributes="0">
<Component id="_limit" alignment="3" min="-2" max="-2" attributes="0"/>
<Component id="jLabel3" alignment="3" min="-2" max="-2" attributes="0"/>
</Group>
<EmptySpace min="-2"
pref="50" max="-2" attributes="0"/>
<Component id="jButton2" min="-2" pref="44" max="-2" attributes="0"/>
</Group>
</Group>
<EmptySpace pref="59" max="32767" attributes="0"/>
</Group>
</Group>
</DimensionLayout>
</Layout>
<SubComponents>
<Component class="javax.swing.JTextField" name="_colonySize">
</Component>
<Component class="javax.swing.JTextField" name="_maksiterasi">
</Component>
<Component class="javax.swing.JTextField" name="_limit">
</Component>
<Component class="javax.swing.JLabel" name="jLabel1">
<Properties>
<Property name="text" type="java.lang.String" value="Colony Size"/>
</Properties>
</Component>
<Component class="javax.swing.JLabel" name="jLabel2">
<Properties>
<Property name="text" type="java.lang.String" value="Maks Iterasi"/>
</Properties>
</Component>
<Component class="javax.swing.JLabel" name="jLabel3">
<Properties>
<Property name="text" type="java.lang.String" value="Limit"/>
```

```
</Properties>
</Component>
<Container class="javax.swing.JScrollPane"
name="jScrollPane1">
    <AuxValues>
        <AuxValue name="autoScrollPane"
type="java.lang.Boolean" value="true"/>
    </AuxValues>

    <Layout class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayout"/>
    <SubComponents>
        <Component class="javax.swing.JTable" name="table">
            <Properties>
                <Property name="model" type="javax.swing.table.TableModel" editor="org.netbeans.modules.form.editors2.TableModelEditor">
                    <Table columnCount="5" rowCount="0">
                        <Column editable="true" title="Projek"
type="java.lang.Object"/>
                        <Column editable="true" title="Mesin 1"
type="java.lang.Object"/>
                        <Column editable="true" title="Mesin 2"
type="java.lang.Object"/>
                        <Column editable="true" title="Mesin 3"
type="java.lang.Object"/>
                        <Column editable="true" title="Mesin 4"
type="java.lang.Object"/>
                    </Table>
                </Property>
                <Property name="columnModel" type="javax.swing.table.TableColumnModel" editor="org.netbeans.modules.form.editors2.TableColumnModelEditor">
                    <TableColumnModel selectionModel="0">
                        <Column maxWidth="-1" minWidth="-1" pref-
Width="-1" resizable="true">
                            <Title/>
                            <Editor/>
                            <Renderer/>
                        </Column>
                        <Column maxWidth="-1" minWidth="-1" pref-
Width="-1" resizable="true">
                            <Title/>
                            <Editor/>
                            <Renderer/>
                        </Column>
                        <Column maxWidth="-1" minWidth="-1" pref-
Width="-1" resizable="true">
                            <Title/>
                            <Editor/>
                            <Renderer/>
                        </Column>
                        <Column maxWidth="-1" minWidth="-1" pref-
Width="-1" resizable="true">
                            <Title/>
                            <Editor/>
                            <Renderer/>
                        </Column>
                    </TableColumnModel>
                </Property>
            </Properties>
        </Component>
    </SubComponents>
</Container>
```

```
        <Column maxWidth="-1" minWidth="-1" pref-
Width="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
        </Column>
    </TableColumnModel>
</Property>
<Property name="tableHeader" type="ja-
vax.swing.table.JTableHeader" editor="org.netbeans.mod-
ules.form.editors2.JTableHeaderEditor">
    <TableHeader reorderingAllowed="true"
resizingAllowed="true"/>
</Property>
</Properties>
</Component>
</SubComponents>
</Container>
<Component class="javax.swing.JButton" name="jButton1">
    <Properties>
        <Property name="text" type="java.lang.String"
value="Load Data"/>
    </Properties>
    <Events>
        <EventHandler event="actionPerformed" lis-
tener="java.awt.event.ActionListener" parame-
ters="java.awt.event.ActionEvent" handler="jButton1Action-
Performed"/>
    </Events>
</Component>
<Component class="javax.swing.JLabel" name="lokasi">
</Component>
<Component class="javax.swing.JLabel" name="jLabel5">
    <Properties>
        <Property name="font" type="java.awt.Font" edi-
tor="org.netbeans.beaninfo.editors.FontEditor">
            <Font name="Tahoma" size="14" style="1"/>
        </Property>
        <Property name="text" type="java.lang.String"
value="Optimasi Penjadwalan Pengerjaan Software pada Soft-
ware House"/>
    </Properties>
</Component>
<Component class="javax.swing.JLabel" name="jLabel6">
    <Properties>
        <Property name="font" type="java.awt.Font" edi-
tor="org.netbeans.beaninfo.editors.FontEditor">
            <Font name="Tahoma" size="14" style="1"/>
        </Property>
        <Property name="text" type="java.lang.String"
value="dengan Flow-Shop Scheduling Problem Menggunakan Al-
goritma Bee Colony "/>
    </Properties>
</Component>
<Component class="javax.swing.JLabel" name="jLabel7">
    <Properties>
        <Property name="font" type="java.awt.Font" edi-
tor="org.netbeans.beaninfo.editors.FontEditor">
            <Font name="Tahoma" size="14" style="1"/>
        </Property>
```

```
</Property>
<Property name="text" type="java.lang.String"
value="-----" />
</Properties>
</Component>
<Component class="javax.swing.JButton" name="jButton2">
<Properties>
<Property name="text" type="java.lang.String"
value="Proses"/>
</Properties>
<Events>
<EventHandler event="actionPerformed" lis-
tener="java.awt.event.ActionListener" parame-
ters="java.awt.event.ActionEvent" handler="jButton2Action-
Performed"/>
</Events>
</Component>
</SubComponents>
</Form>
```

## 8.2 Optimasi Komposisi Menu Makanan

Perkembangan zaman yang semakin maju membuat segala hal menjadi lebih cepat, bahkan hal ini mempengaruhi gaya hidup masyarakat masa kini dengan lebih seringnya mengkonsumsi makanan cepat saji yang menyebabkan berbagai macam penyakit salah satunya ialah diabetes melitus. Indonesia sebagai negara yang berkembang, pola hidup masyarakatnya masih jauh dari pola hidup sehat. Hal ini senada dengan hasil survei yang dikeluarkan oleh AIA Grup bahwa Indonesia hanya meraih skor 55 dari batas skor 100 dalam AIA Healthy Living Index 2013 (AIA Grup, 2013). Untuk penderita diabetes mellitus sendiri, Indonesia menempati urutan ke-4 terbesar di dunia dengan jumlah penderita sebesar 5,6% dan menurut WHO penderita diabetes mellitus diperkirakan dapat mencapai 21,3 juta pada tahun 2030.

Menurut Suryono dalam Mahmudy dan Rianawati (2015) "Diabetes mellitus merupakan kumpulan gejala yang timbul pada diri seseorang yang disebabkan oleh adanya peningkatan glukosa darah akibat kekurangan insulin baik absolut maupun relatif". Umumnya setiap asupan yang dicerna manusia akan diubah menjadi glukosa lalu di salurkan kesetiap bagian sel tubuh dengan bantuan insulin. Bagi penderita diabetes melitus memiliki kesulitan dalam mencerna glukosa dikarenakan kurangnya kadar insulin sehingga menimbulkan penumpukan glukosa dalam sel.

Karena asupan bagi penderita diabetes harus sangat diperhatikan dan dipantau untuk mendapatkan komposisi menu

makanan yang tepat. Namun hal ini terkadang mengalami kendala, maka dibutuhkan suatu metode untuk mengoptimalkan komposisi yang tepat bagi setiap penderita. Salah satu metode optimasi ialah Particle Swarm Optimization (PSO). PSO sendiri cukup mudah untuk diimplementasikan dan menghasilkan solusi yang cukup stabil. Oleh karena itu PSO sangat cocok digunakan untuk penentuan komposisi menu makanan bagi penderita diabetes melitus.

### 8.2.1 Formulasi Permasalahan

Terdapat kasus yaitu optimasi penentuan komposisi menu makanan penderita Diabetes Mellitus. Optimasi dilakukan dengan menggunakan PSO. Komposisi menu makanan yang dihasilkan akan memiliki nilai fitness yang menunjukkan seberapa bagus solusi yang dihasilkan. Tujuan utamanya adalah memaksimalkan nilai fitness komposisi menu makanan. Proses penentuan komposisi menu makanan menggunakan PSO adalah sebagai berikut:

1. Menghitung kebutuhan kalori berdasarkan data penderita.
2. Inisialisasi parameter PSO.
3. Inisialisasi populasi awal.
4. Proses PSO (diulang hingga kondisi berhenti terpenuhi)
  - a. Update kecepatan masing-masing individu.
  - b. Update posisi masing-masing individu.
  - c. Hitung nilai fitness masing-masing individu.
  - d. Pilih global best dan local best untuk masing-masing individu.
5. Dekode global best menjadi komposisi menu makanan.

Pada Studi kasus ini, digunakan data makanan sebanyak 55 makanan yang dibagi menjadi 4 bagian (makanan pokok, makanan sumber protein nabati, hewani, sayuran, dan pelengkap). Sedangkan pada penelitian kami, data yang digunakan sebanyak 18 data makanan untuk jenis Makanan Pokok (MP), 20 data makanan untuk jenis makanan Sumber Protein Nabati (SN), 33 data makanan untuk jenis makanan Sumber Protein Hewani (SH), 35 data makanan untuk jenis makanan Sayuran (SY), dan 35 data makanan untuk jenis makanan Pelengkap (PLK).

Tabel 8.33. Daftar Komposisi Zat Gizi Makanan per 100 gram

No	Nama Bahan Makanan	Energi (Kal)	Karbohidrat (gr)	Protein (gr)	Lemak (gr)
Makanan Pokok					
1	Nasi Putih	180	39.8	3	0.3

2	Nasi Beras Merah	149	32.5	2.8	0.4
3	Nasi Tim	120	26	2.4	0.4
4	Nasi Ketan Putih	163	26	3	0.4
5	Nasi Ketan Hitam	181	37.3	4	1.2
6	Bihun	348	82.1	4.7	0.1
7	Biskuit	458	75.1	6.9	14.4
8	Kentang rebus	62	13.5	2.1	0.2
9	Makroni rebus	353	78.7	8.7	0.4
10	Maizena	341	85	0.3	0
11	Nasi Jagung Kuning	358	82.7	5.5	0.1
<b>Makanan Sumber Protein Nabati</b>					
1	Kembang Tahu Rebus	90	4.7	10.7	4
2	Kacang Kedelai Rebus	189	12.7	20.2	8.2
3	Tahu Goreng	115	2.5	9.7	8.5
4	Tempe Goreng	336	7.8	20	28
5	Tauco	184	22.2	11.4	5.5
6	Kacang Tolo Rebus	138	2.6	10.7	1.1
7	Oncom	187	22.6	13	6
8	Kacang Tanah Rebus	220	8	10.6	18
9	Kacang Hijau Rebus	109	18.3	8.7	0.5
10	Keripik Tempe	581	41.7	12.1	40.6
11	Keripik Oncom	598	8.3	43.8	42.6
<b>Makanan Sumber Protein Hewani</b>					
1	Ayam Goreng	27	9.9	32.3	11.2
2	Empal Goreng	248	10.1	36.2	6.9
3	Bandeng Goreng	123	0	20	4.8
4	Pempek Tengiri	173	33.4	7.2	1.2
5	Tumis Bandeng	189	0.6	11.3	15.7
6	Gurame Asam Manis	192	12.7	12.7	10.1
7	Jambal Goreng	286	2.3	48.5	9.2
8	Mujair Acar Kuning	330	12.1	17.8	23.4

9	Ikan Mas Pepes	209	11.8	15.2	11.3
10	Belut Goreng	417	32	25.9	19.4
11	Sop Daging Sapi	49	5.3	0.8	0.5
Sayuran					
1	Brongkos	141	12.6	15.3	3.3
2	Toge Goreng	88	14	3.2	2.1
3	Botok Lamtoro	186	13	11.7	9.7
4	Sayur Bunga Pepaya	49	9.8	1.7	0.3
5	Sayur asem	29	5	0.7	0.6
6	Cap cai	97	4.2	5.8	6.3
7	Sayur Sop	27	1	2	1.3
8	Tumis Bayam Bersantan	48	2.6	1.4	4.2
9	Semur Jengkol	212	29.1	6	10
10	Ketimun	8	1.4	0.2	0.2
11	Pelecing Kangkung	75	10	2.3	2.8
Makanan Pelengkap					
1	Yoghurt	52	4	3.3	2.5
2	Alpukat	85	7.7	0.9	6.5
3	Anggur hutan	30	6.8	0.5	0.2
4	Apel	58	14.9	0.3	0.4
5	Arbai	37	8.3	0.8	0.5
6	Duku	63	16.1	1	0.2
7	Jambu biji	49	11.8	0.6	0.2
8	Jeruk Bali	48	12.4	0.6	0.2
9	Jeruk Manis	45	11.2	0.9	0.2
10	Mangga	52	12.3	0.7	0
11	Kesemek	78	20	0.8	0.4

Data penderita Diabetes Millitus adalah adalah sebagai berikut :

Umur	Berat Badan	Tinggi Badan	Jenis Kelamin	Pekerjaan
50	55	175	L	Bengkel

1. Menghitung kebutuhan kalori dari data penderita Diabetes Mellitus

Perhitungan kalori dari pasien tersebut adalah sebagai berikut:

- BBI =  $0,9 \times (175 - 100) = 67,5$
- AMB Pria =  $67,5 \times 30 = 2025$  Kkal
- AMB Umur =  $-5\% \times 2025 = -101,25$  Kkal
- TEE Ringan =  $+20\% \times 2025 = 405$  Kkal
- AMB Berat Badan =  $+20\% \times 2025 = 405$  Kkal

Sehingga diperoleh kebutuhan gizi pada penderita tersebut adalah sebagai berikut:

- Total Kalori =  $2025 - 101,25 + 405 + 405 = 2733,75$  Kkal
- Karbohidrat (Kkal) =  $60\% \times 2733,75 = 1858,95$  Kkal = 410,063 gr
- Protein (Kkal) =  $12\% \times 2733,75 = 328,05$  Kkal = 82,0125 gr
- Lemak (Kkal) =  $20\% \times 2733,75 = 546,75$  Kkal = 60,75 gr

Dari perhitungan yang telah dilakukan maka diperoleh kebutuhan karbohidrat sebesar 410,063 gr, protein sebesar 82,0125 gr dan lemak sebesar 60,75gr.

## 2. Inisialisasi parameter PSO

Sebelumnya, data makanan dikonversi ke agar memenuhi kebutuhan asupan dengan meningkatkan jumlah berat tiap makanan, untuk bahan makanan pokok ditingkatkan menjadi 2 kali, sumber protein nabati menjadi  $\frac{1}{2}$  kali, sumber protein hewani menjadi 0.8 kali, sayuran menjadi 2 kali dan pelengkap menjadi  $1\frac{1}{2}$  kali.

Tabel 8.34. Hasil Konversi Komposisi Gizi Data Makanan

No	Nama Bahan Makanan	Energi (Kal)	Karbohidrat (gr)	Protein (gr)	Lemak (gr)
Makanan Pokok					
1	Nasi Putih	360	79.6	6	0.6
2	Nasi Beras Merah	298	65	5.6	0.8
3	Nasi Tim	240	52	4.8	0.8
4	Nasi Ketan Putih	326	52	6	0.8
5	Nasi Ketan Hitam	362	74.6	8	2.4
6	Bihun	696	164.2	9.4	0.2
7	Biskuit	916	150.2	13.8	28.8
8	Kentang rebus	124	27	4.2	0.4
9	Makroni rebus	706	157.4	17.4	0.8
10	Maizena	682	170	0.6	0
11	Nasi Jagung Kuning	716	165.4	11	0.2
Makanan Sumber Protein Nabati					

1	Kembang Tahu Rebus	45	2.35	5.35	2
2	Kacang Kedelai Rebus	94.5	6.35	10.1	4.1
3	Tahu Goreng	57.5	1.25	4.85	4.25
4	Tempe Goreng	168	3.9	10	14
5	Tauco	92	11.1	5.7	2.75
6	Kacang Tolo Rebus	69	1.3	5.35	0.55
7	Oncom	93.5	11.3	6.5	3
8	Kacang Tanah Rebus	110	4	5.3	9
9	Kacang Hijau Rebus	54.5	9.15	4.35	0.25
10	Keripik Tempe	290.5	20.85	6.05	20.3
11	Keripik Oncom	299	4.15	21.9	21.3
Makanan Sumber Protein Hewani					
1	Ayam Goreng	21.6	7.92	25.84	8.96
2	Empal Goreng	198.4	8.08	28.96	5.52
3	Bandeng Goreng	98.4	0	16	3.84
4	Pempek Tengiri	138.4	26.72	5.76	0.96
5	Tumis Bandeng	151.2	0.48	9.04	12.56
6	Gurame Asam Manis	153.6	10.16	10.16	8.08
7	Jambal Goreng	228.8	1.84	38.8	7.36
8	Mujair Acar Kuning	264	9.68	14.24	18.72
9	Ikan Mas Pepes	167.2	9.44	12.16	9.04
10	Belut Goreng	333.6	25.6	20.72	15.52
11	Sop Daging Sapi	39.2	4.24	0.64	0.4
Sayuran					
1	Brongkos	282	25.2	30.6	6.6
2	Toge Goreng	176	28	6.4	4.2
3	Botok Lamtoro	372	26	23.4	19.4
4	Sayur Bunga Pepaya	98	19.6	3.4	0.6
5	Sayur asem	58	10	1.4	1.2
6	Cap cai	194	8.4	11.6	12.6
7	Sayur Sop	54	2	4	2.6

8	Tumis Bayam Bersantan	96	5.2	2.8	8.4
9	Semur Jengkol	424	58.2	12	20
10	Ketimun	16	2.8	0.4	0.4
11	Pelecing Kangkung	150	20	4.6	5.6
Makanan Pelengkap					
1	Yoghurt	78	6	4.95	3.75
2	Alpukat	127.5	11.55	1.35	9.75
3	Anggur hutan	45	10.2	0.75	0.3
4	Apel	87	22.35	0.45	0.6
5	Arbai	55.5	12.45	1.2	0.75
6	Duku	94.5	24.15	1.5	0.3
7	Jambu biji	73.5	17.7	0.9	0.3
8	Jeruk Bali	72	18.6	0.9	0.3
9	Jeruk Manis	67.5	16.8	1.35	0.3
10	Mangga	78	18.45	1.05	0
11	Kesemek	117	30	1.2	0.6

### 3. Inisialisasi populasi awal.

Representasi partikel dalam metode PSO untuk permasalahan ini menggunakan representasi *real code*. Dalam satu partikel terdapat 15 gen memuat bilangan *real code* yang merupakan representasi dari no index makanan. 15 gen menunjukkan jumlah makanan yang dikonsumsi oleh pasien dalam satu hari terdiri atas 3 waktu yaitu pagi, siang dan malam. Susunan makanan yang dikonsumsi setiap waktu harus memenuhi komposisi makanan yang sudah ditentukan sehingga diperoleh keseimbangan gizi. Susunan makanan tersebut adalah Makanan Pokok (MP), Makanan yang mengandung Sumber Protein Nabati (SN), Makanan yang mengandung Sumber Protein Hewani (SH), Sayuran (SY), dan Makanan Pelengkap (PLK). Range angka untuk representasi gen partikel dapat menggunakan range angka berapapun. Pada penelitian ini, range nilai yang digunakan berada diantara 1 - 100 dimaksudkan agar ruang pemilihan gen partikel tidak terlalu besar dan kompleks.

N O	PAGI					SIANG		MALAM				
	MP	SN	SH	SY	PL K	MP	..	MP	SN	S H	SY	PL K
P1	36	32	98	81	83	91	..	..	73	55	23	25

P2	19	13	32	71	92	64	..	86	31	23	63	97
P3	24	98	80	16	68	22	..	10	74	43	26	38

Kemudian nilai tersebut kemudian dikonversi (normalisasi) menjadi sebagai berikut. Nilai dari tiap populasi menyatakan no index dari tiap bahan makanan.

NO	PAGI					SIANG		MALAM				
	MP	SN	SH	SY	PLK	MP	..	MP	SN	SH	SY	PLK
P1	4	4	11	9	10	11	..	8	9	7	3	3
P2	3	2	4	8	11	8	..	10	4	3	7	11
P3	3	11	9	2	8	3	..	2	9	5	3	5

- Keterangan :

- MP : Makanan Pokok
- SN : Makanan Sumber Protein Nabati
- SH : Makanan Sumber Protein Hewani
- SY : Sayuran
- PLK : Makanan Pelengkap

Dihitung nilai fitness tiap partikel menggunakan rumus berikut.

$$\text{Nilai Fitness} = \frac{1000}{\text{penalty}}$$

Nilai penalti merupakan nilai yang diberikan jika nilai total dari tiap komponen gizi tidak sesuai dengan kebutuhan dari penderita. Nilai Penalti =  $|x_i - x'_i| \times \text{prioritas}$

	Kebutuhan penderita	Prioritas
Karbohidrat	410.063	5
Protein	82.0125	5
Lemak	60.75	5

Tabel 8.35. Fitnes tiap partikel

No	Kebutuhan	Pagi	Siang	Malam	Total	Penalti

		MP	SN	SH	SY	PLK	...	...											
P1	Karbohidrat	52	3.9	4.24	58.2	18.45	...	...	436.57	132.535									
	Protein	6	10	0.64	12	1.05	...	...	130.15	240.688									
	Lemak	0.8	14	0.4	20	0	...	...	85.37	123.1									
	TOTAL PENALTI									496.323									
FITNESS																			
2.01482																			
P2	Karbohidrat	52	6.35	26.72	5.2	30	...	...	436.57	205.135									
	Protein	4.8	10.1	5.76	2.8	1.2	...	...	130.15	13.4625									
	Lemak	0.8	4.1	0.96	8.4	0.6	...	...	85.37	8.05									
	TOTAL PENALTI									496.323									
FITNESS																			
2.01482																			
P3	Karbohidrat	52	4.15	9.44	28	18.6	...	...	315.81	471.265									
	Protein	4.8	21.9	12.16	6.4	0.9	...	...	168.35	431.688									
	Lemak	0.8	21.3	9.04	4.2	0.3	...	...	111.76	255.05									
	TOTAL PENALTI									1158									
FITNESS																			
0.86356																			

- **GBest**

P2	19	13	32	71	92	64	86	35	11	33	86	31	23	63	97
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

- **PBest**

NO	PAGI					SIANG	MALAM
	MP	SN	SH	SY	PLK		
P1	36	32	98	81	83	..	..
P2	19	13	32	71	92	..	..
P3	24	98	80	16	68	..	..

2. Proses PSO (diulang hingga kondisi berhenti terpenuhi)

Untuk iterasi pada kasus ini dibatasi hanya 1 iterasi.

a. Update kecepatan masing-masing individu.

Dengan  $r_1=0.3$ ,  $r_2=0.3$ , dan threshold = 50.

NO	PAGI					SIANG	MALAM

	MP	SN	SH	SY	PLK	...	...
P1	-5.1	-5.7	- 19.8	-3	2.698	...	...
P2	0	0	0	0	0	...	...
P3	-1.5	-25.5	- 14.4	16.49	7.195	...	...

b. Update posisi masing-masing individu.

NO	PAGI					SIANG	MALAM
	MP	SN	SH	SY	PLK		
P1	30.9	26.3	78.21	78	85.7	...	...
P2	19	13	32	71	92	...	...
P3	22.5	72.52	65.61	32.49	75.2	...	...

c. Hitung nilai fitness masing-masing individu.

- Nilai partikel yang diupdate dikonversi terlebih dahulu.

NO	PAGI					SIANG	MALAM
	MP	SN	SH	SY	PLK		
P1	4	3	9	9	10	...	...
P2	3	2	4	8	11	...	...
P3	3	8	8	4	9	...	...

- Nilai fitness tiap partikel kemudian dihitung.

No	Kebutuhan	Pagi					Siang	Malam	Total	Penalty
		MP	SN	SH	SY	PLK				
P1	Karbohidrat	65	2.35	0	26	10.2	...	...	306.91	515.765
	Protein	5.6	5.35	16	23.4	0.75	...	...	196.52	572.538
	Lemak	0.8	2	3.84	19.4	0.3	...	...	62.48	8.65
	TOTAL PENALTY								1096.95	
	FITNESS								0.91162	
P2	Karbohidrat	79.6	2.35	7.92	26	10.2	...	...	331.91	390.765
	Protein	6	5.35	25.8 4	23.4	0.75	...	...	176.32	471.538
	Lemak	0.6	2	8.96	19.4	0.3	...	...	71.88	55.65
	TOTAL PENALTY								917.953	

	FITNESS									1.08938
P3	Karbohidrat	79.6	1.25	0	25.2	11.55	...	...	371.76	191.515
	Protein	6	4.85	16	30.6	1.35	...	...	178.12	480.538
	Lemak	0.6	4.25	3.84	6.6	9.75	...	...	76.08	76.65
	TOTAL PENALTI									748.703
	FITNESS									1.33564

d. Pilih global best dan local best untuk masing-masing individu.

- PBEST

NO	PAGI					SIANG		MALAM	
	MP	SN	SH	SY	PLK				
P1	30.9	26.3	78.21	78	85.7	...	...		
P2	19	13	32	71	92	...	...		
P3	22.5	72.52	65.61	32.49	75.2	...	...		

- GBest

P2	19	13	32	71	92	64	86	35	11	33	86	31	23	63	97
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

3. Dekode global best menjadi komposisi menu makanan.

Komposisi Makanan yang dihasilkan adalah sebagai berikut:

P2	3	2	4	8	11	8	10	4	2	4	10	4	3	7	11
----	---	---	---	---	----	---	----	---	---	---	----	---	---	---	----

- Pagi:
  - Makanan Pokok: Nasi Tim
  - Protein Nabati: Kacang Kedelai Rebus
  - Protein Hewani: Pempek Tengiri
  - Sayuran: Tumis Bayam Bersantan
  - Pelengkap: Ketimun
- Siang
  - Makanan Pokok: Kentang Rebus
  - Protein Nabati: Keripik Tempe
  - Protein Hewani: Pempek Tengiri
  - Sayuran: Toge Goreng
  - Pelengkap: Apel
- Malam

- Makanan Pokok: Nasi Jagung Kuning
- Protein Nabati: Tempe Goreng
- Protein Hewani: Bandeng Goreng
- Sayuran: sayur Sop
- Pelengkap: Kesemek

## 8.2.2 Implementasi

Berikut ini diberikan bagian kode program PHP untuk implementasi Optimasi Komposisi Menu Makanan bagi Penderita Diabetes Mellitus Menggunakan Particle Swarm Optimization.

Source Code 8.2 Perhitungan Kalori

Kalori.php

```
private function perhitungan_kalori($bb, $tb, $usia, $gender, $aktivitas) {
    if (($tb < 160 AND $gender == TRUE) OR ($tb < 150 AND $gender == FALSE)) {
        $bbi = $tb - 100;
    } else {
        $bbi = 0.9 * ($tb - 100);
    }

    //AMB Gender
    if ($gender == true) {
        $ambgender = $bbi * 30;
    } else {
        $ambgender = $bbi * 25;
    }

    //AMB Umur
    if ($usia >= 40 and $usia <= 59) {
        $ambumur = -5 * $ambgender / 100;
    } else if ($usia >= 60 and $usia <= 69) {
        $ambumur = -10 * $ambgender / 100;
    } else if ($usia >= 70) {
        $ambumur = -20 * $ambgender / 100;
    }

    //TEE Aktivitas
    if ($aktivitas == 'BR') {
        $tee = 10 * $ambgender / 100;
    } else if ($aktivitas == 'R') {
        $tee = 20 * $ambgender / 100;
    } else if ($aktivitas == 'S') {
        $tee = 30 * $ambgender / 100;
    } else if ($aktivitas == 'B') {
        $tee = 40 * $ambgender / 100;
    } else if ($aktivitas == 'SB') {
```

```
        $tee = 50 * $ambgender / 100;
    }

    //AMB Berat Badan
    $ambbbb = 0;
    if ($bb <= (90 * $bbi / 100)) {
        $ambbbb = 20 * $ambgender / 100;
    } else if ($bb >= (110 * $bbi / 100) and $bb < (120
* $bbi / 100)) {
        $ambbbb = -20 * $ambgender / 100;
    } else if ($bb >= (120 * $bbi / 100)) {
        $ambbbb = -20 * $ambgender / 100;
    }

    return $ambumur + $ambgender + $tee + $ambbbb;
}
```

### Source Code 8.3 Inisialisasi Partikel Bagian 1

Particle.php

```
public static function setup($min, $max, Kalori &$kalori) {
    Particle::$min = $min;
    Particle::$max = $max;
    Particle::$bound = (Particle::$max - Particle::$min) * 0.6 / 2;
    Particle::$kalori = $kalori;
    Particle::$jumlah = [
        count(Particle::$kalori->data["MP"]),
        count(Particle::$kalori->data["SN"]),
        count(Particle::$kalori->data["SH"]),
        count(Particle::$kalori->data["SY"]),
        count(Particle::$kalori->data["PLK"])
    ];
}
```

### Source Code 8.4 Inisialisasi Partikel Bagian 2

Particle.php

```
public function __construct($length) {
    for ($i = 0; $i < $length; $i++) {
        $this->x[] = rand(Particle::$min, Particle::$max);
        $this->v[] = 0.0;
    }

    $this->pbest = $this->x;
```

```
    $this->fitness = $this->pbest_fitness = $this->fit-
ness();
}
```

### Source Code 8.5 Update Kecepatan dan Posisi Partikel

#### Particle.php

```
public function update(array $gBest, $w, $n1, $r1, $n2,
$r2) {
    for ($i = 0; $i < count($this->x); $i++) {
        $x = $this->x[$i];

        $inertia = $this->v[$i] * $w;
        $local = $n1 * $r1 * ($this->pbest[$i] - $x);
        $global = $n2 * $r2 * ($gBest[$i] - $x);
        $v = $inertia + $local + $global;

        $this->v[$i] = $v;
        $this->v[$i] = min($this->v[$i], Parti-
cle::$bound);
        $this->v[$i] = max($this->v[$i], -Parti-
cle::$bound);

        $this->x[$i] += $this->v[$i];
        $this->x[$i] = min($this->x[$i], Parti-
cle::$max);
        $this->x[$i] = max($this->x[$i], Parti-
cle::$min);
    }

    $this->fitness = $this->fitness();
    if ($this->fitness > $this->pbest_fitness) {
        $this->pbest = $this->x;
        $this->pbest_fitness = $this->fitness;
    }
}
```

### Source Code 8.6 Perhitungan Penalti

#### Kalori.php

```
public function penalti(array $indeks_makanan) {
    $jenis = ["MP", "SN", "SH", "SY", "PLK"];

    $penalti = 0.0;
    $karbohidrat = 0.0;
    $protein = 0.0;
    $lemak = 0.0;

    for ($i = 0; $i < count($indeks_makanan); $i++) {
        $j = $jenis[$i % 5];
```

```
$m = $indeks_makanan[$i];  
  
$makanan = $this->data[$j][$m];  
$karbohidrat += $makanan["karbohidrat"];  
$protein += $makanan["protein"];  
$lemak += $makanan["lemak"];  
}  
  
$penalti += abs($karbohidrat - $this->karbohidrat)  
* $this->faktor_karbohidrat;  
$penalti += abs($protein - $this->protein) * $this-  
>faktor_protein;  
$penalti += abs($lemak - $this->lemak) * $this-  
>faktor_lemak;  
  
return $penalti;  
}
```

### Source Code 8.7 Perhitungan Fitness

Kalori.php

```
public function fitness() {  
$kode = $this->x;  
for ($i = 0; $i < count($kode); $i++) {  
/*  
* Rumus konversi ditambahkan dengan -1 karena indeks ma-  
kanan  
* dimulai dari 1, sedangkan array dimulai dari 0.  
*/  
$jumlah = Particle::$jumlah[$i % 5];  
$kode[$i] = ceil($kode[$i] / Particle::$max * $jumlah) - 1;  
}  
  
return 1000 / (Particle::$kalori->penalti($kode) + 1);  
}
```

## 8.3 Optimasi Penjadwalan Praktikum

Masalah penjadwalan dalam universitas merupakan salah satu masalah kompleks yang bisa dikaitkan dengan masalah optimasi. Dan kenyataannya proses pencarian solusi pada permasalahan ini memakan waktu komputasi yang cukup lama. Terlebih lagi harus memenuhi tiap komponen yang ada dan berbeda, sebagai contoh kebutuhan mahasiswa dalam perkuliahan sebisa mungkin jangan sampai terkendala hanya karena tidak bisa mengambil matakuliah wajib dikarenakan jadwal yang bentrok dengan matakuliah lain. Selain itu dari segi dosen, jangan sampai terjadi bentrok dengan matakuliah lain yang diajarkan (Puspaningrum et. al, 2013). Salah satu

permasalahan penjadwalan yang kompleks adalah penjadwalan praktikum, dimana dalam permasalahan ini memiliki komponen yang harus dipenuhi yaitu waktu dosen pengampu, waktu asisten, waktu mahasiswa, jumlah ruang praktikum atau laboratorium (lab), dan masih banyak lagi.

Pada penelitian ini, akan dibahas mengenai penjadwalan praktikum matakuliah Kecerdasan Buatan dengan menggunakan *Modified Real Code Particle Swarm Optimization*. Permasalahan yang akan diselesaikan adalah pengelolaan komponen jadwal yang terdiri dari data dosen, data asisten, data mahasiswa, data ruangan, dan waktu praktikum. Penjadawalan ini tidak boleh bentrok dengan jadwal asisten dan juga mahasiswa, dimana ruangan dan dosen juga dijadikan acuan dalam penyusunannya.

### 8.3.1 Formulasi Permasalahan

Terdapat sebuah studi kasus yaitu melakukan Optimasi Penjadwalan Praktikum Kecerdasan Buatan menggunakan *Modified Real Code Particle Swarm Optimization*. Representasi partikel menggunakan representasi bilangan *real* yang telah dilakukan pembulatan menjadi bilangan integer, dimana menyatakan asisten praktikum tiap kelas. Jumlah data dosen yang digunakan yaitu 7 dosen dengan jumlah asisten 13 orang. Rincian data sebagai berikut:

1. Jumlah kelas yang mengikuti praktikum sebanyak 10 kelas (Kelas A-J).

Kelas	A	B	C	D	E	F	G	H	I	J
-------	---	---	---	---	---	---	---	---	---	---

2. Ruangan yang dapat digunakan untuk praktikum sebanyak 2 ruangan. Setiap ruangan hanya dapat digunakan oleh 5 kelas seperti yang dijelaskan pada Tabel 2.2 karena jumlah kelas yang ada adalah 10 kelas.

Tabel 8.36 Data Kelas Praktikum

	Kelas				
R1	A	B	C	D	E
R2	F	G	H	I	J

3. Waktu praktikum dilaksanakan mulai pukul 07.00 – 16.00 dalam 5 sesi. Detail jam untuk tiap sesi dijelaskan pada Tabel 2.3.

Tabel 8.37 Data Sesi Praktikum

Sesi	Jam
------	-----

1	07.00 – 08.40
2	08.40 – 10.20
3	10.20 – 12.00
4	12.50 – 14.30
5	14.30 – 16.10

4. Jumlah dosen pengampu sebanyak 7 dosen. Dosen pertama mengampu 3 kelas, dosen kedua mengampu 2 kelas dan lainnya masing-masing mengampu 1 kelas. Detail dosen pengampu kelas dijelaskan pada Tabel 2.4 berikut.

Tabel 8.38 Data Dosen Pengampu

Dosen Pengampu	Kelas
Satrio Hadi Wijoyo	C
	I
Lailil Muflikhah	D
	E
	F
Ika Kusumaning Putri	J
Mochammad Hannats Hanafi I.	G
M. Ali Fauzi	H
Imam Cholissodin	B
Nurizal Dwi Priandani	A

5. Jumlah asisten praktikum untuk matakuliah Kecerdasan Buatan adalah 13 asisten seperti pada Tabel 2.5, dimana nantinya masing – masing kelas akan diampu oleh 2 asisten.

Tabel 8.39 Data Asisten Praktikum

1	M. Syafiq
2	M. Kadafi
3	Sabrina Nurfadilla
4	Diva Kurnianingtyas
5	Nanda Putri

6	Radita Noer Pratiwi
7	Daneswara
8	Andriansyah Yusuf R.
9	Karina
10	Anandhi Tristiaratri
11	Agung Nurjaya Megantara
12	Ardiansyah S
13	Fathor Rosi

Dari data – data tersebut akan dimodelkan secara matematis yang kemudian akan dihitung menggunakan *Modified RCPSO* untuk mencari jadwal praktikum yang paling optimal. Pencarian jadwal yang optimal juga dilakukan dengan memperhatikan batasan – batasan atau *constraints* (*hard & soft*) yang telah ditentukan. *Hard Constraint* merupakan batasan yang harus dipenuhi sedangkan *Soft Constraint* merupakan batasan yang tidak harus dipenuhi namun tetap dijadikan acuan dalam penyusunannya. Berikut dijelaskan secara detail batasan tersebut:

1. *Hard Constraints*

- a) Asisten hanya boleh mengajar maksimal 2 kelas (Berlebih).
- b) Asisten tidak boleh mengampu kelas pada dosen yang sama (Bentrok Dosen).
- c) Dalam 1 kelas tidak boleh diajar oleh 1 asisten (Berbeda).
- d) Semua asisten harus mendapat jadwal praktikum (AssKosong).

2. *Soft Constraints*

- a) Pasangan asisten harus berbeda tiap kelasnya (Bentrok).

Proses pencarian solusi untuk penjadwalan praktikum Kecerdasan Buatan menggunakan *Modified RCPSO* adalah sebagai berikut:

1. Melakukan inisialisasi yang terdiri dari jumlah populasi, posisi awal partikel ( $X(t)$ ) dan kecepatan awal partikel ( $V(t)$ ), yang ditunjukkan pada Tabel 2.6 dan Tabel 2.7. Pada studi kasus ini, digunakan 5 data jadwal praktikum Kecerdasan Buatan dengan inisialisasi nilai random. Dimana representasi partikelnya diperlukan 2 asisten dalam 1 kelas, yang terdiri dari 10 kelas. Ukuran partikel adalah 10 kelas \* 2 asisten.

Tabel 8.40 Inisialisasi Posisi Awal

Partikel	Dosen 1						Dosen 2			
	R1									
	1	2	3	4	5	6	7	8	9	10
x1 (0)	10,22395	9,16237	3,030135	4,786689	6,321767	1,955164	1,032145	8,338708	5,479899	1,138948
x2 (0)	9,781649	12,98802	12,9623	12,17555	11,31893	3,974978	5,298702	12,57014	3,053669	12,95498
x3 (0)	9,949213	3,751939	7,778681	2,136219	8,928919	8,159139	5,039251	3,107157	4,905078	7,675562
x4 (0)	11,32501	11,06555	12,23801	11,42255	10,8512	5,957511	9,677188	5,721055	8,355164	8,998986
x5 (0)	6,713819	4,821933	9,143745	6,466953	5,65612	6,073682	5,928162	8,384774	8,51198	7,525965

Dosen 3		Dosen 4		Dosen 5		Dosen 6		Dosen 7							
R2															
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
4,680729	6,538516	11,41938	9,379669	11,20815	5,022638	3,075084	12,8808	6,894117	5,803907						
5,819601	2,828956	2,335693	5,034577	10,4637	12,68408	7,964992	9,096512	12,90121	3,039609						
2,971217	3,415325	4,037991	12,77086	12,81367	6,672778	2,99986	1,405522	10,73455	6,24005						
10,88537	8,395882	2,161735	11,76061	4,114144	3,748596	4,019999	11,77324	5,992979	4,864754						
5,808836	2,742267	10,60418	8,858045	2,213391	5,795307	4,724929	7,514536	10,07346	5,0725						

Keterangan:

X = Partikel, terdapat 5 partikel

R = Ruang, terdapat 2 ruang

K = Kelas, terdapat 10 kelas

A = Fitur, terdapat 20 fitur

Tabel 8.41 Inisialisasi Kecepatan Awal Partikel

V(t)	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
v1 (0)	0	0	0	0	0	0	0	0	0	0
v2 (0)	0	0	0	0	0	0	0	0	0	0
v3 (0)	0	0	0	0	0	0	0	0	0	0
v4 (0)	0	0	0	0	0	0	0	0	0	0
v5 (0)	0	0	0	0	0	0	0	0	0	0

x11	x12	x13	x14	x15	x16	x17	x18	x19	x20
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

2. Melakukan pembulatan dari bilangan *real* menjadi integer terhadap representasi partikel seperti pada Tabel 2.8.

Tabel 8.42 Pembulatan Menjadi Bilangan Integer

Partikel	Dosen 1						Dosen 2			
				R1						
	1	2	3	4	5	6	7	8	9	10
x1 (0)	10	9	3	5	6	2	1	8	5	1
x2 (0)	10	13	13	12	11	4	5	13	3	13
x3 (0)	10	4	8	2	9	8	5	3	5	8
x4 (0)	11	11	12	11	11	6	10	6	8	9
x5 (0)	7	5	9	6	6	6	6	8	9	8

Dosen 3	Dosen 4	Dosen 5	Dosen 6	Dosen 7		
R2						
6	7	8	9	10		
11	12	13	14	15	16	17
5	7	11	9	11	5	3
6	3	2	5	10	13	8
3	3	4	13	13	7	3
11	8	2	12	4	4	12
6	3	11	9	2	6	5

3. Melakukan proses repair yaitu dilakukan pengecekan terhadap asprak yang belum dapat jadwal/kelas (repair dilakukan setiap initialisasi awal & update posisi) dan akan menggantikan asprak yang mengajar lebih dari 2 kelas. Selanjutnya dilakukan pengecekan *constraint* untuk mengevaluasi nilai *fitness* ( $F(X_i)$ ). Berdasarkan penjelasan *constraint* sebelumnya pada kasus ini nilai *fitness* diamambil dari banyaknya batasan yang dilanggar sehingga semakin kecil jumlah pelanggaran yang dihasilkan maka solusi yang dihasilkan akan semakin baik. Untuk tiap pelanggaran yang terjadi akan diberikan nilai 1. Agar tidak terjadi nilai *fitness* yang tak terhingga maka jumlah total semua pelanggaran ditambahkan 1. Hasilnya ditunjukkan pada Tabel 2.9 dan 2.10.

Rumus *fitness*:

$$F = \frac{1}{1 + (\sum \text{Berlebih} + \sum \text{Bentrok Dosen} + \sum \text{Berbeda} + \sum \text{Ass Kosong})}$$

Tabel 8.43 Repair dan Cek Constraint

Partikel	Dosen 1						Dosen 2			
	R1						4		5	
	1	2	3	4	5	6	7	8	9	10
x1(0)	10	9	3	5	6	2	1	8	5	1
x2(0)	10	13	13	12	11	4	5	13	3	13
x3(0)	10	4	8	2	9	8	5	3	5	8
x4(0)	11	11	12	11	11	6	10	6	8	9
x5(0)	7	5	9	6	6	6	6	8	9	8

Dosen 3		Dosen 4		Dosen 5		Dosen 6		Dosen 7						
R2														
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
5	7	11	9	11	5	3	3	13	7	6	13	3	13	6
6	3	2	5	10	13	13	8	9	9	10	11	1	11	6
3	3	4	13	13	7	3	1	1	11	12	6	5	5	5
11	8	2	12	4	4	4	4	4	12	6	6	5	5	5
6	3	11	9	2	6	6	5	8	10	5	5	13	8	8

Tabel 8.44 Hasil Repair

Partikel	Dosen 1						Dosen 2			
	R1						4		5	
	1	2	3	4	5	6	7	8	9	10
x1(0)	10	9	3	5	6	2	1	8	5	1
x2(0)	10	13	1	12	11	4	5	7	3	13
x3(0)	10	4	12	2	9	8	5	3	5	8
x4(0)	1	11	12	3	7	6	10	6	8	9
x5(0)	7	5	9	1	4	6	12	8	13	8

Dosen 3		Dosen 4		Dosen 5		Dosen 6		Dosen 7						
R2														
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4	7	11	9	11	12	3	13	7	7	6	13	3	13	6
6	3	2	5	10	13	13	8	9	9	10	11	1	11	6
3	3	4	13	13	7	3	1	1	11	12	6	5	5	5
11	8	2	12	13	4	4	4	4	12	6	6	5	5	5
6	3	11	9	2	6	6	5	8	10	5	5	13	8	8

Hard									
Berlebih	Bentrok Dosen	Berbeda	AssKosong	Jumlah	Fitness				
0	1	0	0	1	0,5				
3	0	0	0	3	0,25				
2	2	1	0	5	0,166667				
2	0	0	0	2	0,333333				
3	1	0	0	4	0,2				

Fitness:  

$$\frac{1}{1 + (\text{jumlah Constraint})}$$

Keterangan:

Berlebih	Bentrok Dosen	Berbeda	AssKosong
A		A	

- Melakukan inisialisasi Pbest awal tiap partikel. Karena masih pada iterasi 0 sehingga nilai Pbest akan sama dengan posisi awal partikel yang ditunjukkan pada Tabel 2.11.

Tabel 8.45 Pbest Partikel

Pbest(t)	Dosen 1						Dosen 2			
	R1									
	1	2	3	4	5	6	7	8	9	10
Pbest1 (0)	10	9	3	5	6	2	1	8	5	1
Pbest2 (0)	10	13	1	12	11	4	5	7	3	13
Pbest3 (0)	10	4	12	2	9	8	5	3	5	8
Pbest4 (0)	1	11	12	3	7	6	10	6	8	9
Pbest5 (0)	7	5	9	1	4	6	12	8	13	8

Dosen 3		Dosen 4		Dosen 5		Dosen 6		Dosen 7						
R2														
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4	7	11	9	11	12	3	13	7	7	6	3	1	11	6
6	3	2	5	10	13	8	9	13	5	0,166667	1	1	13	3
3	3	4	13	13	7	3	1	11	2	0,333333	4	12	6	5
11	8	2	12	13	4	4	4	13	6	5	5	8	10	5
6	3	11	9	2	6	5	8	10	4	0,2				

Hard							
Berlebih	Bentrok Dosen	Berbeda	AssKosong	Jumlah	Fitness		
0	1	0	0	1	0,5		
3	0	0	0	3	0,25		
2	2	1	0	5	0,166667		
2	0	0	0	2	0,333333		
3	1	0	0	4	0,2		

5. Mencari Gbest dengan mengevaluasi keseluruhan fitness Pbest dan mengambil yang paling maksimum, hasilnya ditunjukkan pada Tabel 2.12.

Tabel 8.46 Gbest Partikel

Gbest(0) =	P1										
Pertikel ke-											
1	10	9	3	5	6	2	1	8	5	1	
4	7	11	9	11	12	3	13	7	6	0,5	Fitness

6. Masuk pada iterasi pertama ( $t=1$ ), maka dilakukan update kecepatan partikel ditunjukkan pada Tabel 2.13 dengan rumus seperti pada 2.1. Dimana nilai  $\lambda$  dicari dengan rumus seperti pada 2.2 dan 2.3. Begitupula dengan nilai  $\omega$  dicari dengan rumus seperti pada 2.4.

vmax	1,507734	xmax	13	c1	1	r1	0,43341	w	0,8995
vmin	-1,50773	xmin	1	c2	1	r2	0,650849		
k	0,251289			c	2	tmax	1000	t	1

Tabel 8.47 Update Kecepatan

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
v1 (1)	0	0	0	0	0	0	0	0	0	0
v2 (1)	0	-2,6034	1,301698	-4,55594	-3,25424	-1,3017	-2,6034	0,650849	1,301698	-7,81019
v3 (1)	0	3,254245	-5,85764	1,952547	-1,95255	-3,90509	-2,6034	3,254245	0	-4,55594
v4 (1)	5,857641	-1,3017	-5,85764	1,301698	-0,65085	-2,6034	-5,85764	1,301698	-1,95255	-5,20679
v5 (1)	1,952547	2,603396	-3,90509	2,603396	1,301698	-2,6034	-7,15934	0	-5,20679	-4,55594

x11	x12	x13	x14	x15	x16	x17	x18	x19	x20
0	0	0	0	0	0	0	0	0	0
-1,3017	2,603396	5,857641	2,603396	0,650849	-0,65085	-3,25424	2,603396	-3,90509	1,952547
0,650849	2,603396	4,555943	-2,6034	-1,3017	3,254245	0	7,810188	-2,6034	0
-4,55594	-0,65085	5,857641	-1,95255	-1,3017	5,206792	-0,65085	0,650849	0,650849	0,650849
-1,3017	2,603396	0	0	5,857641	3,905094	-1,3017	3,254245	-1,95255	0,650849

7. Lakukan perbaikan terhadap hasil update kecepatan dengan memenuhi kondisi di bawah ini, hasilnya ditunjukkan pada Tabel 2.14.

$$v_i = \begin{cases} V_{max} & \text{if } v_i > V_{max} \\ -V_{max} & \text{if } v_i < -V_{max} \end{cases}$$

Tabel 8.48 Perbaikan Kecepatan

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
v1 (1)	0	0	0	0	0	0	0	0	0	0
v2 (1)	0	-1,50773	1,301698	-1,50773	-1,50773	-1,3017	-1,50773	0,650849	1,301698	-1,50773
v3 (1)	0	1,507734	-1,50773	1,507734	-1,50773	-1,50773	-1,50773	1,507734	0	-1,50773
v4 (1)	1,507734	-1,3017	-1,50773	1,301698	-0,65085	-1,50773	-1,50773	1,301698	-1,50773	-1,50773
v5 (1)	1,507734	1,507734	-1,50773	1,507734	1,301698	-1,50773	-1,50773	0	-1,50773	-1,50773

	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20
0	0	0	0	0	0	0	0	0	0	0
-1,3017	1,507734	1,507734	1,507734	0,650849	-0,65085	-1,50773	1,507734	-1,50773	1,507734	1,507734
0,650849	1,507734	1,507734	-1,50773	-1,3017	1,507734	0	1,507734	-1,50773	0	1,50773
-1,50773	-0,65085	1,507734	-1,50773	-1,3017	1,507734	-0,65085	0,650849	0,650849	0,650849	0,650849
-1,3017	1,507734	0	0	1,507734	1,507734	-1,3017	1,507734	-1,50773	0	-1,50773

Keterangan:

$$\begin{aligned} Vi,j = f(Xi) < f(Pbestk), \text{ maka } f(Pbestk) = Xi. \\ = 0 < 0 \\ = 0 \end{aligned}$$

8. Melakukan update posisi partikel dan hasilnya ditunjukkan pada Tabel 2.15.

Tabel 8.49 Update Posisi

	1	2	3	4	5	6	7	8	9	10
x1 (1)	10	9	3	5	6	2	1	8	5	1
x2 (1)	10	11,49227	2,301698	10,49227	9,492266	2,698302	3,492266	7,650849	4,301698	11,49227
x3 (1)	10	5,507734	10,49227	3,507734	7,492266	6,492266	3,492266	4,507734	5	6,492266
x4 (1)	2,507734	9,698302	10,49227	4,301698	6,349151	4,492266	8,492266	7,301698	6,492266	7,492266
x5 (1)	8,507734	6,507734	7,492266	2,507734	5,301698	4,492266	10,49227	8	11,49227	6,492266

11	12	13	14	15	16	17	18	19	20
4	7	11	9	11	12	3	13	7	6
4,698302	4,507734	3,507734	6,507734	10,65085	12,34915	6,492266	10,50773	11,49227	4,507734
3,650849	4,507734	5,507734	11,49227	11,6983	8,507734	3	2,507734	9,492266	6
9,492266	7,349151	3,507734	10,49227	11,6983	5,507734	3,349151	12,65085	6,650849	5,650849
4,698302	4,507734		11	9	3,507734	7,507734	3,698302	9,507734	8,492266
									5,650849

Keterangan:

$$\begin{aligned}
 X1^{1,1} &= V1,1 + (\omega * X1^0,1) \\
 &= 0 + (0.8995 * 10) \\
 &= 8.995 \\
 &= 9
 \end{aligned}$$

9. Melakukan kembali proses di atas dengan partikel baru seperti pada Tabel 2.16 hingga kriteria berhenti terpenuhi (Iterasi Maksimum). Proses repair kembali dilakukan seperti pada Tabel 2.17.

Tabel 8.50 Representasi Partikel Baru

Partikel	Dosen 1						Dosen 2			
	R1						R2			
	1	2	3	4	5	6	7	8	9	10
x1 (1)	10	9	3	5	6	2	1	8	5	1
x2 (1)	10	11	2	10	9	3	3	8	4	11
x3 (1)	10	6	10	4	7	6	3	5	5	6
x4 (1)	3	10	10	4	6	4	8	7	6	7
x5 (1)	9	7	7	3	5	4	10	8	11	6

Dosen 3	Dosen 4	Dosen 5	Dosen 6	Dosen 7
R2				
6	7	8	9	10
11	12	13	14	15
4	7	11	9	11
5	5	4	7	11
4	5	6	11	12
9	7	4	10	12
5	5	11	9	4

Tabel 8.51 Hasil Repair

Partikel	Dosen 1						Dosen 2			
	R1									
	1	2	3	4	5	6	7	8	9	10
x1 (1)	10	9	3	5	6	2	1	8	5	1
x2 (1)	10	1	2	10	9	3	3	8	4	13
x3 (1)	10	1	10	4	7	2	8	13	5	6
x4 (1)	3	1	2	5	11	4	8	7	6	7
x5 (1)	9	7	7	3	1	2	10	12	11	6

Dosen 3		Dosen 4		Dosen 5		Dosen 6		Dosen 7						
R2														
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4	7	11	9	11	12	3	13						7	6
5	5	4	7	11	12	6	11						11	5
4	5	6	11	12	9	3	3						9	6
9	7	4	10	12	6	3	13						7	6
13	5	11	9	4	8	4	10						8	6

Hard						
Berlebih	Bentrok Dosen	Berbeda	AssKosong	Jumlah	Fitness	
0	1	0	0	1	0,5	
2	2	1	0	5	0,166667	
1	2	1	0	4	0,2	
3	1	0	0	4	0,2	
0	1	0	0	1	0,5	

10. Melakukan update Pbest dengan mencari nilai *fitness* maksimum tiap partikel seperti pada Tabel 2.18 dan menentukan Gbest pada iterasi pertama seperti pada Tabel 2.19.

Tabel 8.52 Update Pbest

	1	2	3	4	5	6	7	8	9	10
Pbest1 (1)	10	9	3	5	6	2	1	8	5	1
Pbest2 (1)	10	13	1	12	11	4	5	7	3	13
Pbest3 (1)	10	1	10	4	7	2	8	13	5	6
Pbest4 (1)	1	11	12	3	7	6	10	6	8	9
Pbest5 (1)	9	7	7	3	1	2	10	12	11	6

11	12	13	14	15	16	17	18	19	20	Fitness
4	7	11	9	11	12	3	13	7	6	0,5
6	3	2	5	10	13	8	9	13	3	0,25
4	5	6	11	12	9	3	3	9	6	0,2
11	8	2	12	13	4	4	12	6	5	0,33333333
13	5	11	9	4	8	4	10	8	6	0,5

Tabel 8.53 Update Gbest

Gbest(1) =	P1										
Partikel ke-											
1	10	9	3	5	6	2	1	8	5	1	

											Fitness
4	7	11	9	11	12	3	13	7	6	0,5	

### 8.3.2 Implementasi

Berikut ini diberikan bagian kode program PHP untuk implementasi Optimasi Penjadwalan Praktikum menggunakan Modified Real Code Particle Swarm Optimization.

Source Code 8.8 MRCPSO

MRCPSO.php

```
<?php
..
//1) inisialisasi parameter
..
//2) inisialisasi posisi awal
do{
    echo
=====
    echo "ITERASI [\".$iterasi.\"]";
    if($iterasi == 0) {
        $tr4 = mysql_query("truncate table
partikel");
        $tr2 = mysql_query("truncate table gbest");
        $tr3 = mysql_query("truncate table pbest");
        kecepatanawal($jml_partikel, $iterasi);
        $pop1 = mysql_query("select * from
partikel");
        $pop2 = mysql_fetch_array($pop1);
        .
        //echo "<br/>";
        cetakposisi($iterasi);
        pBest($jml_partikel);
        cetakpbest($iterasi);
        gbest($iterasi);
    } else {
        updatekecepatan($iterasi, $iter_max, $x_max,
$x_min, $c1, $c2, $jml_partikel);
        updateposisi($iterasi, $jml_partikel,
$x_max, $x_min, $dimensi, $jml_dosen);
        updatepbest($iterasi, $jml_partikel);
        gbest($iterasi);
    }
    $iterasi++;
} while ($iterasi <= $iter_max);
cetakgbest();
```

```
//inisialisasi kecepatan awal
function kecepatanawal($jml_partikel, $iterasi){
    $kecepatanawall = mysql_query("select * from ke-
cepatan");
    $kecepatanawal2 = mysql_fetch_array($kecepatan-
awall);
    if(empty($kecepatanawal2)){
        for($k = 1; $k <= $jml_partikel; $k++){
            $kecepatanawal3 = mysql_query("INSERT
into kecepatan (id) values('$k')");
        }
    } else {
        $kcp = mysql_query("truncate table ke-
cepatan");
        for($k = 1; $k <= $jml_partikel; $k++){
            $kecepatanawal6 = mysql_query("INSERT
into kecepatan (id) values('$k')");
        }
    }
    $kecepatanawal4 = mysql_query("select * from ke-
cepatan");
    echo '<table>
        <thead>
            <tr>
                <th>Kecepatan</th>';
        for($x=1; $x<=20; $x++){
            echo
"<th>X".$x."</th>";
        }
    echo'      </tr>
        </thead>
        <tbody>';
    while ($newArray = mysql_fetch_array($kecepatan-
awal4)){
        echo "<tr class='gradeA'>";
        echo "<td><center>V".$newAr-
ray['id']."'".$iterasi."</center></td>";
        for($x2=1; $x2<21; $x2++){
            echo "<td><center>".$newAr-
ray['x'].$x2."</center></td>";
        }
        echo "</tr>";
    }
    echo "</tbody>";
    echo "</table>";
    //echo "Kecepatan[$k]: ";
    /*for($l = 1; $l <= $dimensi; $l++){
        $V[$k][$l] = 0;
        if(empty($kecepatanawal2)){
            $kecepatanawal4 =
mysql_query("INSERT into kecepatan (x$l) val-
ues('".$V[$k][$l]."')");
        }
        echo $V[$k][$l]." | ";
    }*/
    //echo "<br/>";
}
echo "<br/>";
}
```

```
function carikosong($i, $as, $jml_partikel, $dimensi){  
    $query3 = mysql_query("select  
x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,  
x18,x19,x20 from partikel where id = '$i'");  
    $query4 = mysql_fetch_array($query3);  
    for($l = 1; $l <= 13; $l++){  
        for($m = 0; $m < $dimensi; $m++){  
//perulangan dimensi x1 - x20  
            if($query4[$m] == $l){  
                $count[$l][$m] = 1;  
            } else {  
                $count[$l][$m] = 0;  
            } $berlebih[$l][$m] =  
$count[$l][$m];  
        }  
        $au = $l+1;  
        $sum_lebih[$i][$l] = array_sum($ber-  
lebih[$l]);  
        //echo "Jumlah jadwal asisten ke-  
".$l." [".$i."] = ".$sum_lebih[$i][$l]."<br>";  
  
        //repair  
        if($sum_lebih[$i][$l] > 2){  
            $al = $l;  
            for($b = 0; $b < $dimensi;  
$b++){  
                if($query4[$b] ==  
$as);  
                $query4[$b] =  
$bb = $b + 1;  
                $upd  
=mysql_query("update partikel set x$bb = '$query4[$b]'  
where id = '$i'" or die("Gagal Repair".mysql_error()));  
                break;  
            }  
        }  
    } //echo "<br>";  
}  
  
function constraint($i, $iterasi, $jml_partikel, $dimensi,  
$jml_dosen){  
    $query3 = mysql_query("select  
x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,  
x18,x19,x20 from partikel where id = '$i'");  
    $query4 = mysql_fetch_array($query3);  
    for($l = 1; $l <= 13; $l++){  
        for($m = 0; $m < $dimensi; $m++){ //perulan-  
gan dimensi x1 - x20  
            if($query4[$m] == $l){  
                $count[$l][$m] = 1;  
            } else {  
                $count[$l][$m] = 0;  
            } $berlebih[$l][$m] = $count[$l][$m];  
        }  
        $au = $l+1;
```

```
$jumlah_lebih[$i][$l] = array_sum($ber-lebih[$l]);
//echo "Jumlah jadwal asisten ke-".$l."
[$i][$l] = ".$jumlah_lebih[$i][$l]."<br>";

if($jumlah_lebih[$i][$l] == 0){
    $count2[$i][$l] = 1;
} else {
    $count2[$i][$l] = 0;
} $askosong[$i][$l] = $count2[$i][$l];

if($jumlah_lebih[$i][$l] > 2){
    $count3[$i][$l] = $jumlah_lebih[$i][$l]-2;
} else {
    $count3[$i][$l] = 0;
} $aslebih[$i][$l] = $count3[$i][$l];

$jumlah_askosong[$i] = array_sum($askosong[$i]);
$jumlah_aslebih[$i] = array_sum($aslebih[$i]);
echo "C1) Asisten yg belum dapat jadwal =
".$jumlah_askosong[$i];
echo "<br>";
echo "C2) Asisten yg kelebihan jadwal =
".$jumlah_aslebih[$i];
echo "<br>";

$dosen11 = mysql_query("select x1,x2,x3,x4,x5,x6
from partikel where id = '$i'");
$dosen22 = mysql_query("select x7,x8,x9,x10 from
partikel where id = '$i'");
$dosen33 = mysql_query("select x11,x12 from partikel
where id = '$i'");
$dosen44 = mysql_query("select x13,x14 from partikel
where id = '$i'");
$dosen55 = mysql_query("select x15,x16 from partikel
where id = '$i'");
$dosen66 = mysql_query("select x17,x18 from partikel
where id = '$i'");
$dosen77 = mysql_query("select x19,x20 from partikel
where id = '$i'");
$dosen1 = mysql_fetch_array($dosen11);
$dosen2 = mysql_fetch_array($dosen22);
$dosen3 = mysql_fetch_array($dosen33);
$dosen4 = mysql_fetch_array($dosen44);
$dosen5 = mysql_fetch_array($dosen55);
$dosen6 = mysql_fetch_array($dosen66);
$dosen7 = mysql_fetch_array($dosen77);
//echo "<br>";
for($g = 1; $g <= $jml_dosen; $g++){ //perulangan jumlah dosen
    if($g == 1){
        $jumlah_arr = count($dosen1)/2;
        for($h=0; $h<$jumlah_arr; $h++){
            $ds11[$h] = $dosen1[$h];
            //echo $ds11[$h]." || ";
        }
        //echo "<br>";
        $ds2 = array_count_values($ds11);
```

```
$count4[$i][$g] = 0;
foreach($ds2 as $key => $val){
    //echo $key."=>".$val." ";
    if($val > 1){
        $hit = $val - 1;
        $count4[$i][$g] += $hit;
    } else {
        $hit = 0;
        $count4[$i][$g] += $hit;
    }
    $bentrokdosen[$i][$g] = $count4[$i][$g];
} //echo "<br>";
} else if($g == 2){
    $jum_arr = count($dosen2)/2;
    for($h=0; $h<$jum_arr; $h++){
        $ds22[$h] = $dosen2[$h];
        //echo $ds22[$h]." || ";
    }
    //echo "<br>";
    $ds2 = array_count_values($ds22);
    $count4[$i][$g] = 0;
    foreach($ds2 as $key => $val){
        //echo $key."=>".$val." ";
        if($val > 1){
            $hit = $val - 1;
            $count4[$i][$g] += $hit;
        } else {
            $hit = 0;
            $count4[$i][$g] += $hit;
        }
        $bentrokdosen[$i][$g] = $count4[$i][$g];
    } //echo "<br>";
} else if($g == 3){
    $jum_arr = count($dosen3)/2;
    for($h=0; $h<$jum_arr; $h++){
        $ds33[$h] = $dosen3[$h];
        //echo $ds33[$h]." || ";
    }
    //echo "<br>";
    $ds2 = array_count_values($ds33);
    $count4[$i][$g] = 0;
    foreach($ds2 as $key => $val){
        //echo $key."=>".$val." ";
        if($val > 1){
            $hit = $val - 1;
            $count4[$i][$g] += $hit;
        } else {
            $hit = 0;
            $count4[$i][$g] += $hit;
        }
        $bentrokdosen[$i][$g] = $count4[$i][$g];
    } //echo "<br>";
} else if($g == 4) {
```

```
$jum_arr = count($dosen4)/2;
for($h=0; $h<$jum_arr; $h++){
    $ds44[$h] = $dosen4[$h];
    //echo $ds44[$h]." || ";
}
//echo "<br>";
$ds2 = array_count_values($ds44);
$count4[$i][$g] = 0;
foreach($ds2 as $key => $val){
    //echo $key."=>".$val." ";
    if($val > 1){
        $hit = $val - 1;
        $count4[$i][$g] +=
$hit;
    } else {
        $hit = 0;
        $count4[$i][$g] +=
$hit;
    }
    $bentrokdosen[$i][$g] =
$count4[$i][$g];
} //echo "<br>";
} else if($g == 5){
$jum_arr = count($dosen5)/2;
for($h=0; $h<$jum_arr; $h++){
    $ds55[$h] = $dosen5[$h];
    //echo $ds55[$h]." || ";
}
//echo "<br>";
$ds2 = array_count_values($ds55);
$count4[$i][$g] = 0;
foreach($ds2 as $key => $val){
    //echo $key."=>".$val." ";
    if($val > 1){
        $hit = $val - 1;
        $count4[$i][$g] +=
$hit;
    } else {
        $hit = 0;
        $count4[$i][$g] +=
$hit;
    }
    $bentrokdosen[$i][$g] =
$count4[$i][$g];
} //echo "<br>";
} else if($g == 6){
$jum_arr = count($dosen6)/2;
for($h=0; $h<$jum_arr; $h++){
    $ds66[$h] = $dosen6[$h];
    //echo $ds66[$h]." || ";
}
//echo "<br>";
$ds2 = array_count_values($ds66);
$count4[$i][$g] = 0;
foreach($ds2 as $key => $val){
    //echo $key."=>".$val." ";
    if($val > 1){
        $hit = $val - 1;
        $count4[$i][$g] +=
$hit;
    } else {

```

```
$hit = 0;
$count4[$i][$g] +=
$hit;
} $bentrokdosen[$i][$g] =
$count4[$i][$g];
} //echo "<br>";
} else if($g == 7){
$jum_arr = count($dosen7)/2;
for($h=0; $h<$jum_arr; $h++){
$ds77[$h] = $dosen7[$h];
//echo $ds77[$h]." ";
}
//echo "<br>";
$ds2 = array_count_values($ds77);
$count4[$i][$g] = 0;
foreach($ds2 as $key => $val){
//echo $key."=>".$val." ";
if($val > 1){
$hit = $val - 1;
$count4[$i][$g] +=
$hit;
} else {
$hit = 0;
$count4[$i][$g] +=
$hit;
} $bentrokdosen[$i][$g] =
$count4[$i][$g];
} //echo "<br>";
} else {
echo "SALAH";
}
$jum_bendos[$i] = array_sum($bentrok-
dosen[$i]);
}
//echo "<br>";
echo "C3) Bentrok Dosen = ".$jum_bendos[$i];
echo "<br>";

$count5 = 0;
for($z = 0; $z < $dimensi; $z++) {
if($z%2==0) {
if($query4[$z] == $query4[$z+1] ||
$query4[$z+1] == $query4[$z]){
$count5 += 1;
} else {
$count5 += 0;
$berbeda = $count5;
} else {
continue;
}
}
$jum_berbeda[$i] = $berbeda;
echo "C4) Berbeda = ".$jum_berbeda[$i];
echo "<br>";

$fitness[$i] = 1/(1+ ($jum_askosong[$i] +
$jum_aslebih[$i] + $jum_bendos[$i] + $jum_berbeda[$i]));
echo "Fitness P".$i." = ".$fitness[$i];
echo "<br>";
```

```
$fx =mysql_query("update partikel set fitness =
'$fitness[$i]' where id = '$i'" or die("Gagal Tambah
Data".mysql_error()); //simpan nilai fitness ke database
}

function pBest($jml_partikel){
    for($a1 = 1; $a1 <= $jml_partikel; $a1++){
        $pbest1 = mysql_query("select * from pbest
where id = '$a1'");
        $pbest2 = mysql_fetch_array($pbest1);
        if(empty($pbest2)){
            $pbest8 = mysql_query("select * from
partikel where id = '$a1'");
            $pbest9 = mysql_fetch_array($pbest8);
            $hit = count($pbest9)/2;
            for($a2 = 0; $a2 < $hit; $a2++){
                if($a2 == 0){
                    $pbest5 =
mysql_query("insert into pbest (id) values ('$a1')");
                }
                if ($a2 >= 1 && $a2 < 21) {
                    $pbest6 =
mysql_query("update pbest set x$a2 = '$pbest9[$a2]' where
id = '$a1'");
                }
                if ($a2 == 21) {
                    $pbest7 =
mysql_query("update pbest set fitness = '$pbest9[$a2]'"
where id = '$a1');
                }
            }
        } else {
            //$pbest8 = mysql_query("truncate ta-
ble pbest");
            $pbest12 = mysql_query("select * from
partikel where id = '$a1'");
            $pbest13 = mysql_fetch_ar-
ray($pbest12);
            $hit = count($pbest13)/2;
            for($a3 = 0; $a3 < $hit; $a3++){
                if($a3 == 0){
                    continue;
                }
                if ($a3 >= 1 && $a3 < 21) {
                    $pbest10 =
mysql_query("update pbest set x$a3 = '$pbest13[$a3]' where
id = '$a1'");
                }
                if ($a3 == 21) {
                    $pbest11 =
mysql_query("update pbest set fitness = '$pbest13[$a3]'"
where id = '$a1');
                }
            }
        }
    }
}

function cetakpbest($iter){
```

```
$cpbest1 = mysql_query("select * from pbest");
echo '<table>
    <thead>
        <tr>
            <th>Pbest</th>';
        for($x=1; $x<=20; $x++) {
            echo
"<th>X".$x."</th>";
        }
    echo'          <th>Fitness</th>
        </tr>
    </thead>
    <tbody>';
while ($newArray = mysql_fetch_array($cpbest1)) {
    echo "<tr class='gradeA'>";
    echo "<td><center>Pbest".$newAr-
ray['id']."'.".$iter."</center></td>";
    for($x2=1; $x2<21; $x2++) {
        echo "<td><center>".$newAr-
ray['x'].$x2."</center></td>";
    }
    echo "<td><center>".$newArray['fit-
ness']."'."</center></td>";
    echo "</tr>";
    //break;
}
echo "</tbody>";
echo "</table>";
echo "<br>";
}

function cetakposisi($iter) {
    $pos3 = mysql_query("select * from partikel");
    echo '<table>
        <thead>
            <tr>
                <th>Posisi</th>';
            for($x=1; $x<=20; $x++) {
                echo
"<th>X".$x."</th>";
            }
        echo'          <th>Fitness</th>
        </tr>
    </thead>
    <tbody>';
    while ($newArray = mysql_fetch_array($pos3)) {
        echo "<tr class='gradeA'>";
        echo "<td><center>X".$newAr-
ray['id']."'.".$iter."</center></td>";
        for($x2=1; $x2<21; $x2++) {
            echo "<td><center>".$newAr-
ray['x'].$x2."</center></td>";
        }
        echo "<td><center>".$newArray['fit-
ness']."'."</center></td>";
        echo "</tr>";
    }
    echo "</tbody>";
    echo "</table>";
```

```
        echo "<br>";
    }

function cetakgbest(){ //cetak gbest dari seluruh iterasi
    echo "-----Gbest dari seluruh iterasi-----";
    echo "<table class=' id ='>";
    echo "<thead>";
    echo "<tr>";
    echo "<th>Gbest</th>";
    for($x=1; $x<=20; $x++) {
        echo "<th>x".$x."</th>";
    }
    echo "<th>Fitness</th>";
    echo "</tr>";
    echo "</thead>";
    echo "<tbody>";
    $gbest9 = mysql_query("select * from gbest");
    while ($newArray = mysql_fetch_array($gbest9)){
        echo "<tr class='gradeA'>";
        echo "<td><center>Gbest(\".$newArray['iterasi'].\")</center></td>";
        for($x2=1; $x2<21; $x2++){
            echo "<td><center>".$newArray['x'.$x2]."</center></td>";
        }
        echo "<td><center>".$newArray['fitness']. "</center></td>";
        echo "</td>";
    }
    echo "</tbody>";
    echo "</table>";
    echo "<br>";
}

function gBest($iterasi){
    $gbest1 = mysql_query("select MAX(fitness) as fitness from pbest");
    $gbest2 = mysql_fetch_array($gbest1);
    $gbest3 = mysql_query("select * from pbest where fitness = '$gbest2[0]'");
    $gbest4 = mysql_fetch_array($gbest3);
    $count8 = (count($gbest4)/2) + 1;
    for($a1=0; $a1<$count8; $a1++){
        if($a1 == 0){
            $gbest5 = mysql_query("insert into gbest (iterasi) values ('$iterasi')");
        } else if($a1 == 1){
            $z1 = $a1 - 1;
            $gbest6 = mysql_query("update gbest set id = '$gbest4[$z1]' where iterasi = '$iterasi'");
        } else if ($a1 >= 2 && $a1 < 22) {
            $z = $a1 - 1;
            $gbest7 = mysql_query("update gbest set x$z = '$gbest4[$z]' where iterasi = '$iterasi'");
        } else {
            $zz = $a1 - 1;
            $gbest8 = mysql_query("update gbest set fitness = '$gbest4[$zz]' where iterasi = '$iterasi'");
        }
    }
}
```

```
    }
    echo "<table class=' ' id=' '>";
    echo "<thead>";
    echo "<tr>";
    echo "<th>Gbest(\".$iterasi.\")</th>";
    for($x=1; $x<=20; $x++) {
        echo "<th>X\".$x.\"</th>";
    }
    echo "<th>Fitness</th>";
    echo "</tr>";
    echo "</thead>";
    echo "<tbody>";
    $gbest9 = mysql_query("select * from gbest where it-
erasi = '$iterasi'");
    while ($newArray = mysql_fetch_array($gbest9)){
        echo "<tr class='gradeA'>";
        echo "<td><center>P(\".$newAr-
ray['id'].\")</center></td>";
        for($x2=1; $x2<21; $x2++) {
            echo "<td><center>\".$newAr-
ray['x'.'.$x2].\"</center></td>";
        }
        echo "<td><center>\".$newArray['fit-
ness'].\"</center></td>";
        echo "</tr>";
        break;
    }
    echo "</tbody>";
    echo "</table>";
    echo "<br>";
}

function updatekecepatan($iterasi, $iter_max, $x_max,
$x_min, $c1, $c2, $jml_partikel){
    ..
}

function updateposisi($iterasi, $jml_partikel, $x_max,
$x_min, $dimensi, $jml_dosen){
    ..
}

function updatepbest($iterasi, $jml_partikel){
    for($a = 1; $a <= $jml_partikel; $a++){
        $pos = mysql_query("select * from partikel
where id = '$a'");
        $pos2 = mysql_fetch_array($pos);
        $pbest = mysql_query("select * from pbest
where id = '$a'");
        $pbest2 = mysql_fetch_array($pbest);
        $count1 = count($pbest2)/2;
        for($b = 1; $b < $count1; $b++){
            if($pos2[21] > $pbest2[21]){
                $updatepbest[$b] = $pos2[$b];
                $upfitness[$a] = $pos2[21];
            } else {
                $updatepbest[$b] =
$pbest2[$b];
                $upfitness[$a] = $pbest2[21];
            }
        }
    }
}
```

```
        }
        $update = mysql_query("update pbest
set x$b = '$updatepbest[$b]' where id = '$a'");
        } $update2 = mysql_query("update pbest set
fitness = '$upfitness[$a]' where id = '$a'");
    }
cetakpbest($iterasi);
}
?>
```

## 8.4 Optimasi Naïve Bayes Classifier dengan PSO

Klasifikasi merupakan proses pengidentifikasiannya obyek ke dalam sebuah kelas, kelompok, atau kategori berdasarkan prosedur, karakteristik dan definisi yang telah ditentukan sebelumnya (U.S Fish and Wildlife Service, 2013). Klasifikasi bertujuan untuk membagi objek yang ditugaskan hanya ke salah satu nomor kategori yang disebut kelas (Bramer, 2007). Salah satu metode klasifikasi yang biasa digunakan adalah *Naïve Bayes*. Klasifikasi *Naïve Bayes* pertama kali dikemukakan oleh *Thomas Bayes*. Penggunaan metode *Naïve Bayes* sudah dikenalkan sejak tahun 1702-1761. *Naïve Bayes* menurut *Lewis, Hand* dan *Yu* (1998) merupakan pendekatan yang sangat sederhana dan sangat efektif untuk *classification learning* (Hand and Yu, 2001). Menurut Hamzah, *Naïve Bayes* memiliki beberapa kelebihan, yaitu algoritma yang sederhana, lebih cepat dalam perhitungan dan berakurasi tinggi (Hamzah, 2012). Akan tetapi, pada metode *Naïve Bayes* juga memiliki kelemahan dimana sebuah probabilitas tidak bisa mengukur seberapa besar tingkat keakuratan sebuah prediksi. Maka dari itu, metode *Naïve Bayes* perlu dioptimasi dengan cara pemberian bobot menggunakan *Particle Swarm Optimization* (PSO). Metode PSO yang akan digunakan disini diharapkan dapat meningkatkan akurasi dari *Naïve Bayes Classifier*.

### 8.4.1 Formulasi Permasalahan

Data klasifikasi yang digunakan adalah *dataset Iris* yang bisa didapatkan dari *UCI Machine Learning*. Data ini terdiri dari 150 set data yang terbagi menjadi 3 kelas. Setiap kelas memiliki 4 atribut, yaitu *sepal length*, *sepal width*, *petal length* dan *petal width*. Studi kasus ini menggunakan 3 data training class Iris-sentosa, 3 data training class Iris-versicolor dan 3 data training class Iris-virginica. Data training yang digunakan dijelaskan dalam tabel berikut:

No	Sepal Length	Sepal Width	Petal Length	Petal Width	Class
1	5.1	3.5	1.4	0.2	Iris-sentosa
2	4.9	3.0	1.4	0.2	Iris-sentosa
3	4.7	3.2	1.3	0.2	Iris-sentosa
4	7.0	3.2	4.7	1.4	Iris-versicolor
5	6.4	3.1	4.5	1.5	Iris-versicolor
6	6.9	3.1	4.9	1.5	Iris-versicolor
7	6.3	3.3	6.0	2.5	Iris-virginica
8	5.8	2.7	5.1	1.9	Iris-virginica
9	7.1	3.0	5.9	2.1	Iris-virginica

Data testing yang digunakan dijelaskan dalam tabel berikut:

No	Sepal Length	Sepal Width	Petal Length	Petal Width	Class
1	4.6	3.1	1.5	0.2	Iris-sentosa

- Menentukan bobot atribut secara random.

Bobot setiap partikel untuk atribut ditentukan secara random dan dijelaskan pada tabel berikut:

Partikel	Sepal Length	Sepal Width	Petal Length	Petal Width
1	0.2	0.5	0.8	0.2
2	0.5	0.5	0.5	0.5
3	0.6	0.7	0.5	0.3
4	0.9	0.1	0.6	0.7
5	0.1	0.5	0.4	0.2

- Menghitung  $P(C_i)$  setiap kelas.

$$P(Iris - sentosa) = \frac{3}{9} = 0.33$$

$$P(Iris - versicolor) = \frac{3}{9} = 0.33$$

$$P(Iris - virginica) = \frac{3}{9} = 0.33$$

- Menghitung  $\mu$  setiap atribut pada setiap kelas.

Menghitung  $\mu$  pada atribut Sepal Length:

$$\mu(Iris - sentosa) = \frac{5.1+4.9+4.7}{3} = 4.9$$

Kemudian untuk nilai  $\mu$  selanjutnya dijelaskan pada tabel berikut:

Atribut	Class	$\mu$
Sepal Length	Iris-sentosa	4.9

	Iris-versicolor	6.76
	Iris-virginica	6.4
Sepal Width	Iris-sentosa	3.23
	Iris-versicolor	3.16
	Iris-virginica	3.0
Petal Length	Iris-sentosa	1.36
	Iris-versicolor	4.7
	Iris-virginica	5.6
Petal Width	Iris-sentosa	0.2
	Iris-versicolor	1.46
	Iris-virginica	2.16

4. Menghitung  $\sigma$  setiap atribut pada setiap kelas.

Menghitung  $\sigma$  pada atribut Sepal Length:

$$\sigma(Iris - sentosa) = \frac{(5.1-4.9)^2 + (4.9-4.9)^2 + (4.7-4.9)^2}{3-1} = 0.04$$

Kemudian untuk nilai  $\sigma$  selanjutnya dijelaskan pada tabel berikut :

Atribut	Class	$\sigma$
Sepal Length	Iris-sentosa	0.04
	Iris-versicolor	0.1034
	Iris-virginica	0.43
Sepal Width	Iris-sentosa	0.06335
	Iris-versicolor	0.0034
	Iris-virginica	0.09
Petal Length	Iris-sentosa	0.0034
	Iris-versicolor	0.04
	Iris-virginica	0.25
Petal Width	Iris-sentosa	0
	Iris-versicolor	0.0034
	Iris-virginica	0.0934

5. Menghitung  $P(X_i | C_i)$  setiap atribut pada setiap kelas.

Perhitungan untuk  $P(4.6 | Iris - sentosa)$  adalah sebagai berikut:

$$P(4.6|Iris - sentosa) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp^{-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

$$= \frac{1}{\sqrt{2\pi(0.04)^2}} \exp^{-\frac{(4.6 - 4.9)^2}{2(0.04)^2}} = 6.08 \times 10^{-12}$$

Kemudian untuk perhitungan selanjutnya dijelaskan pada tabel berikut:

Atribut	Class	$P(X_i   C_i)$
Sepal Length	Iris-sentosa	$6.08 \times 10^{-12}$
	Iris-versicolor	$6.72 \times 10^{-95}$
	Iris-virginica	$1.45 \times 10^{-4}$
Sepal Width	Iris-sentosa	0.767
	Iris-versicolor	$2.79 \times 10^{-66}$
	Iris-virginica	2.391
Petal Length	Iris-sentosa	0
	Iris-versicolor	0
	Iris-virginica	$6.29 \times 10^{-59}$
Petal Width	Iris-sentosa	0.199
	Iris-versicolor	0
	Iris-virginica	$1.01 \times 10^{-95}$

6. Perhitungan Weighted Naïve Bayes setiap partikel  
Perhitungan untuk probabilitas Iris-sentosa pada partikel adalah sebagai berikut:

$$\begin{aligned}
 P(X_i | Iris - sentosa) \\
 &= 0.33 \times (6.08 \times 10^{-12})^{0.2} \times (0.767)^{0.5} \times (0)^{0.8} \\
 &\quad \times (0.199)^{0.2} = 0
 \end{aligned}$$

Kemudian untuk perhitungan selanjutnya dijelaskan pada tabel berikut:

Partikel	Class	Atribut	Weight	$P(X_i   C_i)$
1	Iris-sentosa	Sepal Length	0.2	0
		Sepal Width	0.5	
		Petal Length	0.8	
		Petal Width	0.2	
	Iris-versi-color	Sepal Length	0.2	0
		Sepal Width	0.5	
		Petal Length	0.8	
		Petal Width	0.2	
2	Iris-virgin-ica	Sepal Length	0.2	$2.42 \times 10^{-67}$
		Sepal Width	0.5	
		Petal Length	0.8	
		Petal Width	0.2	
	Iris-sentosa	Sepal Length	0.5	0
	Sepal Width	0.5		
	Petal Length	0.5		

		Petal Width	0.5	
3	Iris-versi-color	Sepal Length	0.5	0
		Sepal Width	0.5	
		Petal Length	0.5	
		Petal Width	0.5	
	Iris-virginica	Sepal Length	0.5	$1.56 \times 10^{-79}$
		Sepal Width	0.5	
		Petal Length	0.5	
		Petal Width	0.5	
4	Iris-sentosa	Sepal Length	0.6	0
		Sepal Width	0.7	
		Petal Length	0.5	
		Petal Width	0.3	
	Iris-versi-color	Sepal Length	0.6	0
		Sepal Width	0.7	
		Petal Length	0.5	
		Petal Width	0.3	
	Iris-virginica	Sepal Length	0.6	$1.56 \times 10^{-79}$
		Sepal Width	0.7	
		Petal Length	0.5	
		Petal Width	0.3	
5	Iris-sentosa	Sepal Length	0.9	0
		Sepal Width	0.1	
		Petal Length	0.6	
		Petal Width	0.7	
	Iris-versi-color	Sepal Length	0.9	0
		Sepal Width	0.1	
		Petal Length	0.6	
		Petal Width	0.7	
	Iris-virginica	Sepal Length	0.9	$4.9 \times 10^{-106}$
		Sepal Width	0.1	
		Petal Length	0.6	
		Petal Width	0.7	
5	Iris-sentosa	Sepal Length	0.1	0
		Sepal Width	0.5	
		Petal Length	0.4	
		Petal Width	0.2	
	Iris-versi-color	Sepal Length	0.1	0
		Sepal Width	0.5	
		Petal Length	0.4	

	Petal Width	0.2	
Iris-virginica	Sepal Length	0.1	$1.12 \times 10^{-43}$
	Sepal Width	0.5	
	Petal Length	0.4	
	Petal Width	0.2	

## 8.4.2 Implementasi

Berikut ini diberikan bagian kode program JAVA untuk implementasi Optimasi Naïve Bayes Classifier dengan Menggunakan Particle Swarm Optimization pada Data Iris.

Source Code 8.9 NBCPSO

```
Home.java

/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package nbcpso;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author UC ASUS X550V
 */
public class Home extends javax.swing.JFrame {

    /**
     * Creates new form Home
     */
    public Home() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of
     * this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
    // Code">/GEN-BEGIN:initComponents
```

```
private void initComponents() {  
  
    jLabel1 = new javax.swing.JLabel();  
    txtPopSize = new javax.swing.JTextField();  
    jLabel2 = new javax.swing.JLabel();  
    txtJmlIterasi = new javax.swing.JTextField();  
    jLabel3 = new javax.swing.JLabel();  
    txtInersia = new javax.swing.JTextField();  
    jLabel4 = new javax.swing.JLabel();  
    txtC1 = new javax.swing.JTextField();  
    jLabel5 = new javax.swing.JLabel();  
    txtC2 = new javax.swing.JTextField();  
    btnHitung = new javax.swing.JButton();  
    jSeparator1 = new javax.swing.JSeparator();  
    jLabel6 = new javax.swing.JLabel();  
    jLabel7 = new javax.swing.JLabel();  
    jLabel8 = new javax.swing.JLabel();  
    jLabel9 = new javax.swing.JLabel();  
    txtSepalLength = new javax.swing.JTextField();  
    txtSepalWidth = new javax.swing.JTextField();  
    txtPetalLength = new javax.swing.JTextField();  
    txtPetalWidth = new javax.swing.JTextField();  
  
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);  
    setTitle("Optimasi NBCPSO");  
  
    jLabel1.setText("popSize");  
  
    jLabel2.setText("Jumlah Iterasi");  
  
    jLabel3.setText("Inersia");  
  
    jLabel4.setText("C1");  
  
    jLabel5.setText("C2");  
  
    btnHitung.setText("Hitung");  
    btnHitung.addActionListener(new java.awt.event.ActionListener() {  
        public void actionPerformed(java.awt.event.ActionEvent evt) {  
            btnHitungActionPerformed(evt);  
        }  
    });  
  
    jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);  
  
    jLabel6.setText("Sepal Length");  
  
    jLabel7.setText("Sepal Width");  
  
    jLabel8.setText("Petal Length");  
  
    jLabel9.setText("Petal Width");  
  
    txtSepalLength.setEditable(false);
```

```
txtSepalWidth.setEditable(false);

txtPetalLength.setEditable(false);

txtPetalWidth.setEditable(false);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
..
..

pack();
}// </editor-fold>//GEN-END:initComponents

private void btnHitungActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnHitungActionPerformed
    ..
}//GEN-LAST:event_btnHitungActionPerformed

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    // <editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
```

```
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Home().setVisible(true);
        }
    });
}

// Variables declaration - do not modify//GEN-BEGIN:variables
..
// End of variables declaration//GEN-END:variables
}
```

### Bayes.java

```
package nbcpsos;

import java.sql.*;

/**
 *
 * @author UC ASUS X550V
 */
public class Bayes {

    private double inputSepalLength;
    private double inputSepalWidth;
    private double inputPetalLength;
    private double inputPetalWidth;

    private double[] bobot;

    public Bayes() {

    }

    public Bayes(double inputSL, double inputSW, double in-
putPL, double inputPW, double[] bobot) {
        this.inputSepalLength = inputSL;
        this.inputSepalWidth = inputSW;
        this.inputPetalLength = inputPL;
        this.inputPetalWidth = inputPW;
        this.bobot = bobot;
    }

    public Bayes(double[] input, double[] bobot) {
        this.inputSepalLength = input[0];
        this.inputSepalWidth = input[1];
        this.inputPetalLength = input[2];
        this.inputPetalWidth = input[3];
        this.bobot = bobot;
    }
}
```

```
}

public void setInput(double[] input) {
    this.inputSepalLength = input[0];
    this.inputSepalWidth = input[1];
    this.inputPetalLength = input[2];
    this.inputPetalWidth = input[3];
}

public void setBobot(double[] bobot) {
    this.bobot = bobot;
}

private double[][] getDataTrain(int kelas) {
    ..

    ..
    if(kelas == 1) {
        data = dataSentosa;
    }
    else if(kelas == 2) {
        data = dataVersicolor;
    }
    else if(kelas == 3) {
        data = dataVirginica;
    }

    ..
    for(int i=0; i<data.length; i++) {
        for(int j=0; j<data[i].length; j++) {
            dataTrain[i][j] = data[i][j] * this.bob-
bot[j];
        }
    }
    return dataTrain;
}

public double hitPrior(int kelas) {
    double[][] data = getDataTrain(kelas);
    double totalData = getDataTrain(1).length + get-
DataTrain(2).length + getDataTrain(3).length;
    return data.length / totalData;
}

public double hitMean(int atribut, int kelas) {
/*
    atribut 1 = sepal_length
    atribut 2 = sepal_width
    atribut 3 = petal_length
    atribut 4 = petal_width
*/
    double[][] data = getDataTrain(kelas);
    double jml = 0.0;
    for(int i=0; i<data.length; i++) {
        jml = jml + data[i][atribut-1];
    }
    return jml / data.length;
}
```

```
public double hitVarian(int atribut, int kelas) {  
    double[][] data = getDataTrain(kelas);  
    double mean = hitMean(atribut, kelas);  
    double jml = 0.0;  
    for(int i=0; i<data.length; i++) {  
        jml = jml + (Math.pow((data[i][atribut-1] -  
mean), 2));  
    }  
    return jml / (data.length - 1);  
}  
  
public double hitPosterior(double inputAtribut, int  
atribut, int kelas) {  
    ..  
}  
  
public double kaliPosteriorInput(int kelas) {  
    ..  
}  
  
public String hasilKlasifikasi() {  
    String hasil = "";  
    double sentosa = kaliPosteriorInput(1);  
    double versicolor = kaliPosteriorInput(2);  
    double virginica = kaliPosteriorInput(3);  
    double max = Math.max(Math.max(sentosa, versi-  
color), virginica);  
    if(max == sentosa) {  
        hasil = "sentosa";  
    }  
    else if(max == versicolor) {  
        hasil = "versicolor";  
    }  
    else if(max == virginica) {  
        hasil = "virginica";  
    }  
    return hasil;  
}  
  
private double[][] getDataTest() {  
    ..  
  
    ..  
    for(int i=0; i<dataTest.length; i++) {  
        for(int j=0; j<dataTest[i].length; j++) {  
            dataTest[i][j] = data[i][j] * this.bo-  
bot[j];  
        }  
    }  
  
    return dataTest;  
}  
}
```

### Partikel.java

```
package nbcpsos;

import java.util.Random;
import java.sql.*;
import java.util.Arrays;

/**
 *
 * @author UC ASUS X550V
 */
public class Partikel {

    ..

    public Partikel() {
        ..
    }

    public double[] getPosisi() {
        return posisi;
    }

    public double[] getKecepatan() {
        return kecepatan;
    }

    public double getFitness() {
        this.hitFitness();
        return fitness;
    }

    public double[][] getDataTest() {
        ..
        for(int i=0; i<dataTest.length; i++) {
            for(int j=0; j<dataTest[i].length; j++) {
                dataTest[i][j] = data[i][j] * this.posisi[j];
            }
        }

        return dataTest;
    }

    public void hitFitness() {
        double[][] dataTest = getDataTest();
        String[] hasilKlasifikasi = new
String[dataTest.length];
        for(int i=0; i<hasilKlasifikasi.length; i++) {
            hasilKlasifikasi[i] = new Bayes(dataTest[i],
posisi).hasilKlasifikasi();
        }

        int fit = 0;
        ..
        fitness = fit;
    }
}
```

```
}

    public void updateKecepatan(double inersia, double
konKecepatan1, double konKecepatan2, double[] pBest, dou-
ble[] gBest) {
    ..
    this.cekBatasKecepatan();
}

private void cekBatasKecepatan() {
    double posMax = 1.0;
    double posMin = 0.0;
    double k = 0.6;
    double vMax = k * 0.5;
    double vMin = -vMax;
    for(int i=0; i<kecepatan.length; i++) {
        if(kecepatan[i] > vMax) {
            kecepatan[i] = vMax;
        }
        else if(kecepatan[i] < vMin) {
            kecepatan[i] = vMin;
        }
    }
}

public void updatePosisi() {
    ..
    this.cekBatasPosisi();
}

private void cekBatasPosisi() {
    double max = 1.0;
    double min = 0.0;
    for(int i=0; i<posisi.length; i++) {
        if(posisi[i] > max) {
            posisi[i] = max;
        }
        else if(posisi[i] < min) {
            posisi[i] = min;
        }
    }
}

}
```

### PSO.java

```
package nbcpso;

import java.lang.Math;
import java.util.Random;
import java.sql.*;

/**
 *
```

```
* @author UC ASUS X550V
*/
public class PSO {

    ..

    public PSO(int jmlPartikel, int iterasi) {
        ..
    }

    public int getJumlahPartikel() {
        return jumlahPartikel;
    }

    public int getJumlahIterasi() {
        return jumlahIterasi;
    }

    public Partikel getPartikel(int p) {
        return swarm[p];
    }

    public void inisialisasiPBest() {
        this.pBest = this.swarm;
    }

    public void updatePBest() {
        ..
    }

    public void hitGBest() {
        ..
    }

    public Partikel[] getPBest() {
        return pBest;
    }

    public Partikel getGBest() {
        this.hitGBest();
        return gBest;
    }

    public double Rand(){
        double c = min +(double) (Math.random()*max);
        return c;
    }

    public static void main(String[] args) {
        PSO pso = new PSO(..., ...);
        ..
        pso.inisialisasiPBest();
        ..

        int jmlIterasi = pso.getJumlahIterasi();
        for(int i=0; i<jmlIterasi; i++) {
            System.out.println("Iterasi " + (i+1));
        }
    }
}
```

```
        Sys-
    tem.out.println("=====");
    System.out.println();
    ..
    pso.updatePBest();
    ..
}
}
```

#### Daftar Pustaka

- Abramowitz, M. and Stegun, I. A. (Eds.). Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. New York: Dover, p. 10, 1972.
- Ahmed, Zakir H., 2010. *Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator* dalam *International Journal of Biometrics & Bioinformatics (IJBB)* Volume (3): Issue (6) halaman 96 – 105.
- Amini, Mohsen and Majid Bazargan. 2013. Two Objective Optimization in Shell-and-Tube Heat Exchangers using Genetic Algorithm. Department of Mechanical Engineering, K. N. Toosi University of Technology, 15 Pardis St., Mollasadra St., Tehran 1999143344, Iran.
- Andri, S. W., 2013. Aplikasi Traveling Salesman Problem dengan Metode Algoritma Bee Colony.
- Andries P. Engelbrecht. 2007. *Computational Intelligence, An Introduction*. Second Edition. John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England.
- Caesar C.A., Hanum L., Cholissodin I., 2016. Perbandingan Metode ANN-PSO Dan ANN-GA Dalam Pemodelan Komposisi Pakan Kambing Peranakan Etawa (PE)

Untuk Optimasi Kandungan Gizi. JTIIK, Vol.3, No.3, September 2016, hlm. 216-225.

Cheng, H.A.L., 2013. Apply An Automatic Parameter Selection Method To Generalized Discriminant Analysis With RBF Kernel For Hyperspectral Image Classification, [e-journal] pp.253 – 258. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].

Cholissodin I., Farisuddin F., Santoso E., 2016. Klasifikasi Tingkat Resiko Stroke Menggunakan Improved Particle Swarm Optimization dan Support Vector Machine. Konferensi Nasional Sistem & Informasi 2016, At STT Ibnu Sina Batam , 11 – 13 Agustus 2016.

Cholissodin I., Khusna R.A., Wihandika R.C., 2016. Optimization Of Equitable Distributions Of Teachers Based On Geographic Location Using General Series Time Variant PSO. 2nd International Symposium on Geoinformatics (ISyG).

Cholissodin I., Dewi C., Surbakti E.E., 2016. Integrated ANN And Bidirectional Improved PSO For Optimization Of Fertilizer Dose On Palawija Plants. 2nd International Conference on Science in Information Technology (ICSI Tech).

D. Bratton, J. Kennedy, 2007. Defining a standard for particle swarm optimization, IEEE SwarmIntelligence Symposium, SIS'2007, 2007 Apr. 1–5, Honolulu, Hawaii, New Jersey, pp. 120–127.

Eliantara F., Cholissodin I., Indriati, 2016. Optimasi Pemenuhan Kebutuhan Gizi Keluarga Menggunakan Particle Swarm Optimization. Prosiding Seminar Nasional Riset Terapan (SNRT), Politeknik Negeri Banjarmasin, 9-10 Nopember.

E.P. & M., Z.Z. Kurniawan, 2010, Fuzzy Membership Function Generation Using Particle Swarm Optimization: International Journal Open Problems Computation Math, No 3.

- Faisal Amri, E. B., 2012. Artificial Bee Colony Algorithm untuk Travelling Salesman problem.
- Hahsler, Michael dan Hornik, Kurt., 2007. *Infrastructure for the Traveling Salesman Problem* dalam *Journal of Statistical Software Volume (23) Issue (2)*.
- HL Chen, at all. 2011, An Adaptive Fuzzy K-Nearest Neighbor Method Based on Parallel Particle Swarm Optimization for Bankruptcy Prediction, Part 1 LNAI 6634 Page 249-264, Springer-Verlag Berlin Heidelberg.
- Hong, W.C., 2010. Electric Load Forecasting by SVR with Chaotic Ant Swarm Optimization, [e-journal] pp.102 – 107. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- J. Blondin, 2009, Particle Swarm Optimization : A Tutorial, pp. 1-5.
- Jia Z., Gong L., 2008. Multi-criteria Human Resource Allocation for Optimization Problems Using Multi-objective Particle Swarm Optimization Algorithm. IEEE.
- Karaboga, D. B., 2007. A Powerful and Efficient Algotihm for NumericalFunction Optimization Artificial Bee Colony Algorithm (Vol. 39). Turkey: J Glob Optim.
- Kennedy, J and Eberhart, R.C., 1995. *Particle Swarm Optimization*. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*. IEEE Service Center, Piscataway.
- Kun, L., 2012. Rice Blast Prediction Based on Gray Ant Colony And RBF Neural Network Combination Model, [e-journal] pp.144 – 147. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Kuo, B.C., 2014. A Kernel-Based Feature Selection Method for SVM With RBF Kernel for Hyperspectral Image Classification, [e-journal] pp.317 – 326. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].

- Li, X., 2013. Coupling Firefly Algorithm and Least Squares Support Vector Regression for Crude Oil Price Forecasting, [e-journal] pp.80 – 83. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Lixing, D., 2010. Support Vector Regression and Ant Colony Optimization for HVAC Cooling Load Prediction, [e-journal] pp.537 – 541. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- M. Clerc, 2000. 'Discrete Particle Swarm Optimization Illustrated by the Traveling Salesman Problem', <http://www.mauriceclerc.net>.
- M. Hoffmann, et al., 2011. Discrete Particle Swarm Optimization for TSP: Theoretical Results and Experimental Evaluations, in Bouchachia, A. (Ed.), *Adaptive and Intelligent Systems: Second International Conference, ICAIS 2011, Klagenfurt, Austria, September 6-8, 2011. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 416-427.
- Mahmudy, W.F., 2013. *Algoritma Evolusi*.Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.
- Maickel Tuegeh, Soeprijanto, and Mauridhi H. Purnomo, 2009, Modified Improved Particle Swarm Optimization For Optimal, Seminar Nasional Aplikasi Teknologi Informasi.
- Maria, A., Sinaga, E. Y., & Helena, M., 2012. Penyelesaian Masalah Travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO).
- Marini, F., & Walczak, B., 2015. *Particle swarm optimization (PSO). A tutorial*. IEEE Chemometrics and Intelligent Laboratory Systems, 13.
- Martin, A., 2012. An Analysis on Qualitative Bankruptcy Prediction Using Fuzzy ID3 and Ant Colony Optimization Algorithm, [e-journal]. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].

- Meesad, P., 2013. Predicting Stock Market Price Using Support Vector Regression, [e-journal] pp.416 – 421. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Mirza Cilimkovic, Neural Networks and Back Propagation Algorithm. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin 15, Ireland.
- Mohanthy, R., 2014. Predicting Software Reliability Using Ant Colony Optimization, [e-journal] pp.496 – 500. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Mustafa Servet Kiran, H. I., 2012. The Analysis of Discrete Artificial Bee Colony Algorithm. Springer.
- N., Boukadoum, M. & Proulx, R. Nouaouria, 2013, Particle Swarm Classification: A Survey and Positioning: Pattern Recognition, Vol 46.
- Navalertporn, Thitipong, and Afzulpurkar, Nitin. 2011. Optimization Of Tile Manufacturing Process Using Particle Swarm Optimization. Industrial System Engineering, School of Engineering and Technology, Asian Institute of Technology. Thailand.
- Novitasari D., Cholissodin I., Mahmudy W.F., 2016, Hybridizing PSO With SA for Optimizing SVR Applied to Software Effort Estimation, TELKOMNIKA, Vol.14, No.1, March 2016, pp. 245-253.
- Ping, Y., 2011. On-line Adaptation Algorithm for RBF Kernel Based FS-SVM, [e-journal] pp.3963 – 3967. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Ratnaweera A., Halgamuge, SK., Watson HC., 2004. Self Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients. IEEE.

- Saidah, N. H., Er, M., & Soelaiman, R., 2014. Penyelesaian Masalah travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO).
- Sameh Otri B.Eng., M., 2011. Improving The Bees Algorithm For Complex Optimisation Problems. United Kingdom: Cardiff University.
- Soebroto A.A., Cholissodin I., Wihandika R.C., Frestantiya M.T., Arief Z.E., 2015. Prediksi Tinggi Muka Air (TMA) Untuk Deteksi Dini Bencana Banjir Menggunakan SVR-TVIWPSO. JTIIK, Vol. 2, No. 2, Oktober 2015 hlm. 79-86.
- Sun, C., 2010. Gas Bearing Capacity Forecasting Method Based on Ant Colony Optimization and Support Vector Regression, [e-journal] pp.387 – 390. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Syafiq M., Kadafi A.J.A, Zakiyyah R.H, Jauhari D., Luqyana W.A, Cholissodin I., Muflikhah L., 2016. Aplikasi Mobile (Lide) Untuk Diagnosis Tingkat Resiko Penyakit Stroke Menggunakan PTVPSO-SVM. JTIIK, Vol.3, No.2, Juni 2016, hlm. 147-155.
- Tianshi, L., 2014. Improved Ant Colony Optimization for Interval Pumping of Pumping Unit, [e-journal] pp.550 – 555. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Tyas, Y.S., 2013. Aplikasi Pencarian Rute Terbaik dengan Metode Ant Colony Optimization (ACO), [e-journal] pp.55 – 64. Tersedia melalui : IPI Indonesian Publication Index <[http://portalgaruda.org](http://http://portalgaruda.org)> [Diakses 11 September 2015].
- Vijayakumar, S., 1999. Sequential Support Vector Classifiers and Regression. In Proceedings of International Conference on Soft Computing (SOCO '99). pp. 610–619.
- Waliprana, W.E., 2009. Ant Colony Optimization, [e-journal]. Tersedia melalui : IPI Indonesian Publication Index

- <<http://portalgaruda.org>> [Diakses 12 September 2015].
- Wang, H., 2013. Electricity Consumption Prediction Based on SVR with Ant Colony Optimization, [e-journal] pp. 6928-6934. Tersedia melalui : IPI Indonesian Publication Index <<http://portalgaruda.org>> [Diakses 11 September 2015].
- Wardhani B.A.K., Rachmi I., Hasjidla N.F., Khaqiqiyah Z., Triatmaja I., Cholissodin I., 2016. Optimasi Penjadwalan Praktikum Menggunakan Modified Real Code Particle Swarm Optimization (Studi Kasus Fakultas Ilmu Komputer Universitas Brawijaya). JTIIK, Vol.3, No.4, Desember 2016, hlm.265-272.
- Xing, H., 2013. An New Strategy for Online Evaluation of Analog Circuit Performance based Adaptive Least Squares Support Vector Regression with Double Kernel RBF, [e-journal] pp.120 – 124. Tersedia melalui: IEEE Xplore Digital Library <<http://ieeexplore.ieee.org>> [Diakses 11 September 2015].
- Y Fukuyama, 2007, Fundamentals of Particle Swarm Optimization Techniques: K.Y. Lee & M.A. El-Sharkawi, John Wiley & Sons, Inc, Hoboken, NJ, USA.
- Y, Zhang., S, Huang. 2004. A Novel Multiobjective Particle Swarm Optimization for Buoy-arrangement Design. Shenyang Institute of Automation the Graduate School of Chinese Academy of Science. Senyang, China.
- Zou, Jifeng, Chenlong Li, and Qiao Li, 2015. Fault Prediction Method based on SVR of Improved PSO. School of Information Science and Engineering, Shenyang Ligong University, Shenyang 110159, China.



## Indeks

- ABC, i, iv, 3, 28, 30, 33, 35, 44  
acak, 7, 11, 13, 23, 15, 30, 32, 60, 71  
ACO, i, v, 3, 46, 56, 57, 58, 59, 60, 70,  
73, 74  
algoritma, ii, iv, vii, 3, 4, 6, 7, 8, 9, 11,  
16, 18, 19, 20, 27, 31, 33, 17, 26,  
35, 46, 58, 59, 73, 74  
Algoritma, iii, iv, v, vii, 2, 5, 7, 8, 9,  
10, 18, 19, 26, 33, 30, 46, 58, 59,  
77, 83  
Evolusi, 2, 18, 77  
Ant Colony Optimization, v, 46, 56,  
57, 73  
**Artificial Bee Colony**, iv, 28, 30, 31,  
35  
berhenti, v, 16, 17, 18, 29, 31, 34,  
47, 52, 61, 73  
berkelompok, ii, 1, 2  
Berkoloni, 2  
**Bobot inersia**, 8  
buatan, ii, 2, 30, 58  
*constraint*, 4  
*cost*, 4, 6, 46, 47, 52  
Data Latih, vi, 64, 65, 67, 68, 69  
Data Uji, vi, 64, 67, 68, 69  
Denormalisasi, vi, 68, 69  
dimensi, 11, 19, 20, 22, 34, 22  
**Employed Bee**, 29  
*error*, 61  
**Fase**, iv, v, 29, 30, 31, 33, 34, 35, 36,  
39, 43, 44  
feromon, 46, 47, 48, 52, 55, 57, 58,  
60, 61, 71, 72, 74  
*fitness*, 4, 6, 8, 13, 15, 16, 17, 21, 25,  
26, 27, 31, 32, 16, 18, 21, 28, 29,  
30, 31, 36, 40, 43, 45  
fitur, vi, 5, 63, 64, 65  
Flexible  
    Job-Shop, iv, 24  
**Flow-Shop**, iv, 21, 22  
FSP, iv, 21, 22  
fungsi, 6, 10, 11, 17, 19, 20, 15, 16,  
61, 67  
**Global Best**, iv, 8, 29  
*goal*, 4  
**HDPSO**, iv, 33, 17  
Hybrid, iv, v, 33, 17, 59, 60, 73  
individu, ii, 1, 2, 4, 5, 7, 8, 11, 22, 29,  
36, 39, 40, 43, 44  
Inisialisasi, iii, iv, v, 11, 12, 20, 21, 17,  
18, 28, 30, 35, 46, 48, 60, 61, 70  
interval, 13, 22, 23, 31  
iterasi, 8, 11, 12, 13, 15, 16, 17, 18,  
20, 21, 24, 26, 27, 31, 32, 34, 17,  
18, 20, 21, 31, 36, 44, 46, 47, 48,  
57, 60, 61, 71, 74  
jalur, 46, 47, 52, 56, 58, 60, 61, 71,  
73  
Jarak, vi, 17, 18, 20, 21, 35, 36, 40,  
47, 52, 53, 54, 55, 64, 65  
Job-Shop, iv, 23, 24  
kecepatan, 7, 8, 9, 11, 13, 14, 20, 22,  
31, 33, 35, 15, 17, 18, 26  
kecerdasan, ii, 1, 2  
Kecerdasan  
    Alami, 2  
**Koefisien akselerasi**, 8  
koloni, ii, 46, 58  
**kombinatorial**, 33, 16, 17, 24, 26, 44  
komplek, 2, 4, 16  
komponen  
    kognitif, 7  
    sosial, 7  
konvergen, 4, 16, 17, 18, 26, 15, 21  
konversi, 11, 12  
Local best, 29, 32  
*lower*, 13, 22  
maksimum, 10, 13, 17, 19, 22, 26,  
27, 60, 61, 62, 64, 70  
MAPE, vi, 59, 62, 69, 70, 72, 74  
matrik, 4, 47, 52, 55, 65, 66  
*meta-heuristics*, 4  
normalisasi, 62, 63, 64  
**Onlooker Bee**, v, 29, 30, 31, 32, 33,  
34, 39, 44  
optimasi, 2, 4, 7, 18, 19, 26, 16, 26,  
28, 44, 58, 70, 74  
*overfitting*, 61  
*Particle swarm optimization*, 7, 78  
**partikel**, 4, 7, 8, 11, 12, 13, 15, 20,  
21, 22, 24, 27, 29, 30, 31, 32, 33,  
34, 15, 17, 18, 20, 21, 22, 24, 26

- Pengkodean real, 19  
penjadwalan, 21  
Peramalan, vii, 59, 68, 70  
persamaan, 4, 9, 13, 15, 21, 27, 29,  
62, 64, 65, 66, 67, 69  
**Personal best**, 8  
populasi, 2, 4, 5, 6, 8, 11, 12, 13, 21,  
30, 28  
posisi, 4, 7, 8, 9, 12, 13, 15, 21, 22,  
23, 29, 30, 33, 34, 35, 15, 18, 20,  
26, 28, 29, 30  
problem, 4, 16  
PSO, i, iii, iv, vii, 3, 7, 8, 9, 10, 16, 18,  
19, 22, 26, 29, 30, 31, 32, 26, 59,  
75, 78, 80  
*random injection*, vii, 30, 31  
RCPSO, iii, 19, 74  
representasi, ii, 4, 8, 11, 22, 33, 15,  
26, 44  
Rumus, iv, 9, 29, 15, 20, 61  
rute, 16, 58  
Scheduling, iv, 21, 22, 23, 24  
**Scout Bee**, v, 29, 30, 31, 34, 43, 44  
sederhana, 4, 6, 10, 19, 59  
semut, ii, 3, 7, 46, 47, 48, 52, 55, 56,  
57, 58, 60, 61, 71, 72, 73, 74  
Simpul, vi, 17  
simulasi, 3, 71  
sistem  
    cerdas, ii, 2, 5, 30, 31, 34  
**solusi**, ii, 4, 5, 6, 7, 8, 11, 16, 17, 19,  
20, 22, 26, 27, 15, 23, 28, 31, 32,  
33, 34, 43, 44, 46, 47, 52, 58, 72  
string, 11  
Support Vector Regression, v, 61  
SVR, i, v, vii, 59, 60, 61, 62, 70, 73,  
74, 78, 79, 80  
*Swarm Intelligence*, ii, iii, v, vii, 1, 2,  
3, 4, 5, 6, 18, 58, 73  
testing, 62  
training, 61, 62  
TSP, iv, v, vii, 16, 22, 34, 46, 47, 57,  
77  
Tugas, iii, iv, v, 5, 6, 18, 31, 26, 44,  
56, 73  
Two-Stage Assembly, iv, 22  
Update, iii, iv, 5, 11, 13, 14, 15, 16,  
20, 22, 23, 24, 26, 34, 15, 18, 20,  
21, 29, 30, 47, 56, 61, 72, 73  
*upper*, 13, 22  
variabel, 7, 11, 19, 20, 31, 71

### Biografi Penulis



**Imam Cholissodin**, lahir di Lamongan pada tanggal 19 Juli 1985, telah menyelesaikan pendidikan S2 di Teknik Informatika FTIF ITS Surabaya pada Tahun 2011. Sejak Tahun 2012 telah aktif sebagai dosen pengajar di jurusan Teknik Informatika Program Teknologi dan Ilmu Komputer (PTIIK) Universitas Brawijaya (UB) Malang pada beberapa mata kuliah, seperti *Information Retrieval*, Pengolahan Citra Digital, Probabilitas dan Statistik, Grafika Komputer, *Decision Support System*, Kecerdasan Buatan, Data Mining dan Pengenalan Pola. Bidang Keminatan yang ditekuni peneliti adalah *Information Retrieval*, *Artificial Vision*, *Decision Support System*, *Image Processing*, dan *Cryptography*. Di samping mengajar, peneliti juga aktif dalam Riset Group *Image Processing* dan *Vision* (IMPROV) di dalam Laboratorium Komputasi Cerdas dan Visualisasi. Selain itu peneliti juga telah melakukan beberapa publikasi pada jurnal nasional dan internasional (IEEE). Riset pada tahun 2013-2014 yang sedang dilakukan sekarang bersama dengan beberapa tim dosen dan mahasiswa semester akhir adalah berfokus pada bidang *Information Retrieval* untuk melakukan analisis dokumen lembaga pendidikan secara *Real-time*, yaitu dengan tema “*Groups Decision Sentiment Analysis Untuk Klasifikasi Dokumen E-Complaint Kampus Menggunakan Additive Kernel SVM*” yang merupakan kombinasi dari dua lintas bidang keilmuan antara *Decision Support System* (DSS) dan *Information Retrieval* (IR). Motto : “We Are A Code, We Are The Best Code Of God”.



Efi Riyandani , lahir di Riau pada tanggal 20 April 1995. Pada tahun 2016 berhasil menyelesaikan studi S1 Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya Malang dengan tugas akhir yang berjudul “Optimasi Komposisi Pakan Sistem Polikultur Ikan dan Udang Menggunakan Algoritma Genetika”.