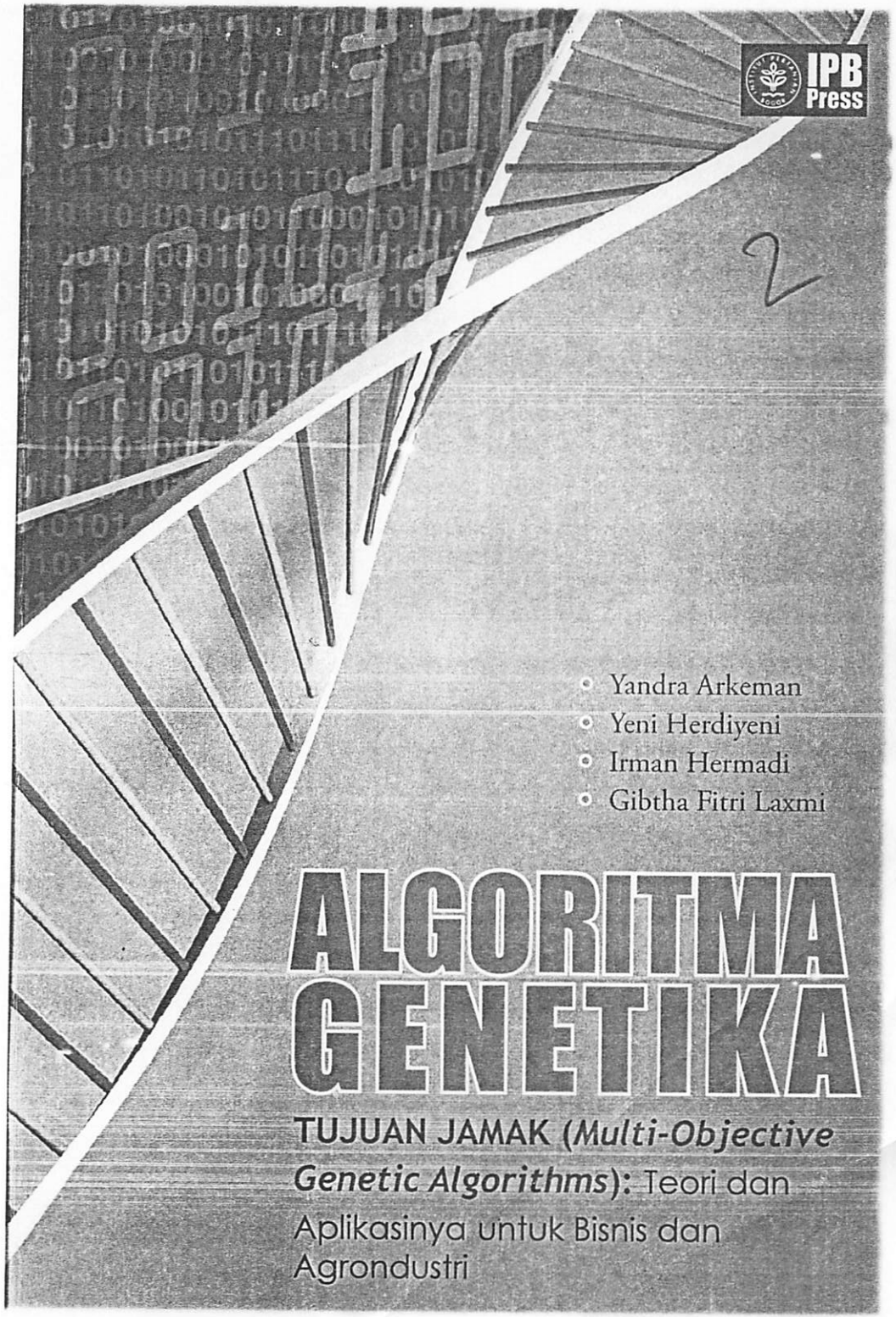

IPB Press
2
Yandra Arkeman
Yeni Herdiyeni
Irman Hermadi
Gibtha Fitri Laxmi
ALGORITMA GENETIKA
TUJUAN JAMAK (Multi-Objective Genetic Algorithms): Teori dan Aplikasinya untuk Bisnis dan Agrondustri

ALGORITMA GENETIKA

TUJUAN JAMAK (*Multi-Objective Genetic Algorithms*):

Teori dan Aplikasinya untuk Bisnis dan Agroindustri

ALGORITMA GENETIKA

TUJUAN JAMAK (*Multi-Objective Genetic Algorithms*):

Teori dan Aplikasinya untuk Bisnis dan Agroindustri

Yandra Arkeman
Yeni Herdiyeni
Irman Hermadi
Gibtha Fitri Laxmi

IPB Press

ALGORITMA GENETIKA
TUJUAN JAMAK (*Multi-Objective Genetic Algorithms*):
Teori dan Aplikasinya untuk Bisnis dan Agroindustri

Yandra Arkeman
Yeni Herdiyeni
Irman Hermadi
Gibtha Fitri Laxmi

Penyunting Bahasa : Nia Januarini
Desain Sampul & Tata Letak : Sani Etyarsah

PT Penerbit IPB Press
Kampus IPB Taman Kencana Bogor
Cetakan Pertama: Juli 2014

Dicetak oleh Percetakan IPB

ISBN: 978-979-493-625-2

KATA PENGANTAR

Algoritma Genetika Tujuan Jamak (*Multi-Objective Genetic Algorithms*) tengah menjadi perbincangan, baik di dunia akademisi maupun di kalangan para praktisi. Perkembangan teori dan aplikasinya telah banyak diterapkan di berbagai bidang ilmu pengetahuan, termasuk teknik, ekonomi, kedokteran, pertanian, bahkan robotika. Para peneliti mempelajari masalah optimasi multi-tujuan dari sudut pandang yang berbeda, sehingga terdapat filosofi solusi yang berbeda dan tujuan. Melalui prinsip-prinsip seleksi alam dan kelangsungan hidup, yang banyak menjadi fondasi pemikiran berbagai disiplin ilmu, algoritma genetik memberikan solusi atas permasalahan optimasi.

Sumbangan atas manfaat dan aplikasi *Multi-Objective Genetic Algorithm* hanyalah butiran debu dari kumpulan ilmu Allah yang diturunkan kepada manusia. Allah telah memberikan kepada manusia rahmat atas semesta alam yang penuh dengan aneka pengetahuan yang kita olah, pahami, dan manfaatkan untuk keberlanjutan hidup. Prinsip genetika tentang kromosom yang berisi *gen* adalah kunci utama dalam memahami terjadinya suksesi kehidupan, pelestarian sifat dari tetua makhluk ke generasi (turunan) berikutnya. Prinsip tersebut adalah suatu prinsip yang memberikan kontribusi dalam bidang komputasi cerdas (*intelligent computing*) yang terus berkembang progresif.

Buku ini merupakan salah satu publikasi terbaru penulis pertama yang didasarkan pada hasil-hasil penelitiannya di bidang *Multi-Objective Algorithm* yang telah dimulainya sejak tahun 2004 ketika melakukan *post doctoral research* di Osaka, Jepang dan dilanjutkan dengan *post doctoral*

research yang kedua di Amerika Serikat (Oktober 2009–Januari 2010). Selain itu, penulis-penulis lainnya juga memberikan kontribusi yang sangat besar terhadap buku ini sesuai dengan keahlian mereka masing-masing di bidang Algoritma Genetika pada umumnya dan *Multi-Objective Algoritms* pada khususnya.

Puji dan syukur kami panjatkan pada Tuhan Yang Maha Berilmu atas karunia dan izin-Nya, sehingga kami dapat menulis dan berbagi pengetahuan tentang teori dan aplikasi algoritma genetik dengan segala keterbatasan sebagai manusia biasa. Buku ini diterbitkan atas kerja sama dengan Direktorat Jenderal Pendidikan Tinggi (DIKTI), Kementerian Pendidikan Nasional dalam bentuk bantuan dana penelitian Kerja Sama Luar Negeri dan Publikasi Internasional untuk penelitian SMART-TIN©. Tiada ilmu bagi kita kecuali apa yang telah diajarkan-Nya kepada kita.

Akhir kata, kami sangat mengharap dan menyambut sumbangan ide, saran, dan kritik untuk menyempurnakan buku ini ke depan agar lebih bermakna dan bermanfaat.

Bogor, 6 Juni 2014

Penulis

DAFTAR ISI

BAB 1
KONSEP OPTIMASI

Optimasi sangat populer dalam berbagai bidang, di mana optimasi merupakan tujuan yang ingin dicapai. Dari setiap proses yang ada, mereka mengharapkan dapat berjalan secara efektif dan efisien dan hasilnya sesuai dengan idealnya mereka. Hasil dari optimasi itu sendiri dapat berupa nilai yang maksimal ataupun nilai yang minimal, misalnya dalam bidang ekonomi mereka ingin mengoptimalkan keuangan mereka dengan cara meminimalkan pengeluaran dan dapat juga dengan memaksimalkan keuntungan dalam perdagangan. Perlu diketahui bahwa optimasi termasuk ke dalam teknik perancangan. Hal ini disebabkan teknik perancangan berperan dalam bagaimana proses dalam sistem berjalan, sehingga menghasilkan keluaran yang diharapkan. Sebagai contoh permisalan tombol yang ada dalam sistem informasi, di mana tombol jika ditekan akan memunculkan data yang diharapkan. Data yang akan muncul dirancang sesuai prosesnya, yaitu mengambil data dari tabel mana, lokasinya yang mana, tipe data yang seperti apa, semua itu butuh proses yang jelas.

Model optimasi yang dikembangkan hingga saat ini semakin beragam, semuanya berawal dari teknik *evolutionary computation.* Perkembangan teknik optimasi di antaranya *genetic algorithm, particle swarm optimization, ant colony, genetic programming,* dan lain sebagainya.

ALGORITMA GENETIK (*GENETIC ALGORITHMS*)

Banyak persoalan dalam kehidupan manusia yang dikategorikan sebagai masalah *"searching"*, yaitu masalah untuk mencari satu pilihan yang paling baik (paling memuaskan) di antara beberapa kemungkinan yang ada. Suatu contoh sederhana, seseorang ingin pergi berlibur ke suatu tempat dengan banyak pilihan jenis pesawat, mobil, hotel, atau restoran yang tersedia. Ia tentu saja harus memutuskan satu kombinasi dari beberapa kombinasi yang tersedia untuk memuaskan keinginannya. Kadang-kadang masalah makin dipersulit karena adanya pertimbangan lain yang perlu diperhatikan. Contohnya, pada satu sisi ia ingin menghemat uang, sedangkan pada sisi lain ia ingin penerbangan yang nyaman.

Masalah *"searching"* juga banyak dijumpai di bidang keteknikan dan manajemen. Sebagai contoh, sebuah robot yang digunakan untuk transportasi barang di dalam pabrik akan melakukan *"searching"* untuk menentukan rute yang paling baik untuk menuju satu tempat tertentu. Contoh lain adalah menara pengatur lalu lintas penerbangan di bandara. Di menara ini, petugas dibantu oleh alat-alat berteknologi tinggi berusaha mencari urutan pendaratan pesawat yang paling aman dan efisien dari berbagai kombinasi urutan pendaratan yang ada setiap harinya. Masalah *"searching"* juga muncul dalam permainan catur atau bentuk permainan strategi lainnya. Dalam permainan catur misalnya, seseorang akan mencari urutan langkah terbaik untuk menaklukkan lawannya secepat mungkin.

Seperti diketahui, istilah *"searching"* dilakukan untuk mencari suatu objek dalam sekumpulan objek yang ada. Dalam masalah optimasi, objek yang dicari adalah solusi optimal atau terbaik untuk masalah yang dihadapi.

Sebagai contoh, untuk masalah penjadwalan produksi, teknik-teknik *"searching"* digunakan untuk mencari jadwal produksi yang optimal, yaitu jadwal yang dapat memberikan biaya paling kecil atau yang bisa meminimumkan waktu pembuatan barang.

Ada beberapa teknik *"searching"* yang dapat digunakan dalam bidang optimasi, di antaranya *hill-climbing search, gradient search, simulated annealing,* dan *genetic algorithms. Hill-climbing* dan *gradient search* adalah teknik *searching* tradisional yang menggunakan teori kalkulus dalam usahanya mencari titik maksimum atau titik minimum suatu fungsi dalam suatu *"search space"*. Teknik ini cocok untuk digunakan jika turunan (derivatif) fungsi yang dioptimasi dapat ditentukan terlebih dahulu. Dengan kata lain, jika fungsi yang dihadapi tidak diketahui fungsi turunannya, teknik-teknik tersebut tidak dapat digunakan.

Simulated annealing menggunakan teknik *"searching"* yang hampir sama dengan teknik *gradient search*. Perbedaan utamanya terletak pada parameter pengukur kesuksesan hasil *"searching"*. Dalam *simulated annealing*, ukuran yang digunakan adalah "temperatur". Penggunaan temperatur sebagai parameter *"searching"* ini adalah meniru teknik yang dipakai para ahli metalurgi dalam membuat metal dengan sifat tertentu. Dalam membentuk metal, bahan-bahan dasarnya dipanaskan dan didinginkan berkali-kali sampai menghasilkan metal unggul yang mempunyai sifat tertentu, misalnya tidak mudah patah karena guncangan yang keras. Prinsip *"searching"* untuk *simulated annealing* juga seperti itu; jika temperatur tinggi, perlu dibuat langkah yang besar (karena masih jauh dari titik optimum). Jika temperatur rendah, yang diperlukan langkah kecil saja untuk menuju titik optimal. Pada awalnya, temperatur akan tinggi

(karena masih jauh dari titik optimal). Namun selanjutnya, seiring dengan berjalannya waktu, temperatur akan turun perlahan-lahan sampai titik nol derajat, yang berarti titik optimum sudah tercapai. *Simulated annealing* telah banyak digunakan untuk menyelesaikan masalah optimasi jadwal dan rute transportasi di pabrik.

Berbeda dengan teknik-teknik sebelumnya, *genetic algorithms* atau algoritma genetik melakukan "*searching*" dengan melakukan simulasi proses evolusi makhluk hidup. Prinsip utamanya yaitu meniru proses seleksi alam dan prinsip-prinsip ilmu genetika. Dalam seleksi alam, individu-individu bersaing untuk mempertahankan hidup dan melakukan reproduksi. Individu-individu yang lebih "*fit*" akan mempunyai peluang untuk terus bertahan hidup (*survive*) dan melakukan reproduksi (menghasilkan keturunan). Sebaliknya, individu-individu yang kurang "*fit*" akan mati dan punah (prinsip ini dinamakan juga "*survival of the fittest*"). Selanjutnya, dalam proses seleksi alam ini, beberapa individu baru yang lebih "*fit*" dari kedua orang tuanya akan "dilahirkan" melalui proses yang disebut penyilangan (*crossover*) dan mutasi. Kedua proses ini terjadi pada kromosom-kromosom individu yang melakukan reproduksi. Proses seleksi dan reproduksi (penyilangan dan mutasi) ini berlangsung berulang kali sampai individu yang paling "*fit*" dihasilkan.

Penjelasan secara lebih terinci tentang algoritma genetik ini akan dibahas pada bab berikutnya. Pembahasan akan mencakup tentang konsep dasar, cara menentukan representasi kromosom, fungsi *fitness*, seleksi, penyilangan, dan mutasi. Pada bagian akhir bab tersebut juga diberikan satu contoh aplikasi algoritma genetik untuk ilustrasi.

PARTICLE SWARM OPTIMIZATION (PSO)

Particle Swarm Optimization (PSO) adalah sebuah populasi yang berdasarkan teknik optimasi stokastik. Dr. Eberhart dan Dr. Kennedy pada tahun 1995 terinsiprasi dari kelompok burung atau kelompok ikan untuk membuat teknik ini. Teknik PSO memiliki banyak kesamaan dengan teknik komputasi evalusioner, seperti Algoritma Genetika (GA). Di mana sistem diinisialisasikan dengan bilangan acak, dengan hasil yang optimum berdasarkan generasi yang dibentuk.

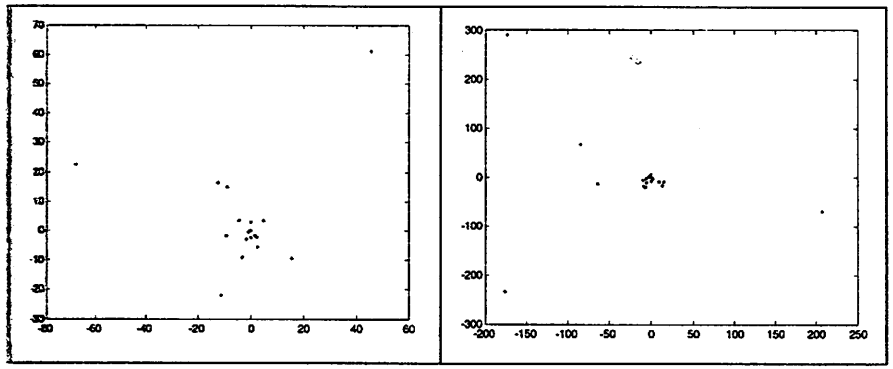
Perbedaan yang dimiliki PSO dari GA ialah tidak adanya proses genetika seperti pindah silang dan mutasi serta mudah diimplementasikan karena tidak adanya parameter yang harus digunakan. Solusi yang berpotensi dalam PSO disebut dengan partikel, setiap partikel akan melalui ruang masalah dan mengikuti partikel yang optimal. Sebagai contoh sekelompok ikan, apabila satu ikan melihat sasaran yang dituju dimisalkan adalah makanan, maka ikan yang lain akan segera ikut menuju ke lokasi yang sama, walaupun berada pada arah yang berlawanan.

Proses dari PSO itu sendiri diawali dengan pembentukan partikel ataupun populasi secara acak. Pembentukan secara acak merupakan pembentukan posisi pergerakan setiap partikel, sehingga setiap partikel memiliki kemampuan eksplorasi tertentu. Pergerakan partikel dilakukan pada ruang multi-dimensi yang setiap partikel memiliki posisi dan kecepatan. Perolehan nilai kecepatan atau *velocity* dalam proses percobaannya menggunakan persamaan sebagai berikut.

$$V = c_1 \times v_0 + c_1 \times rand \times (p_{best} - posisi_{partikel}) + c_2 \times rand \times (g_{best} - posisi_{partikel})$$

Setiap partikel akan bergerak melalui ruang masalah dan pergerakan tersebut memiliki dua kemampuan utama, yaitu ingatan terhadap posisi terbaik masing-masing partikel serta pengetahuan terhadap partikel yang terbaik. Kelompok partikel akan saling mengomunikasikan posisi terbaiknya dan akan menyesuaikan posisi serta kecepatannya berdasarkan posisi partikel yang terbaik antarpartikel. Setiap partikel akan menjaga *record* posisi yang diasosiasikan dengan nilai terbaik yang pernah dilewatinya.

Sebagai contoh, percobaan PSO pada Gambar 1 di mana pergerakan partikel akan mengikuti posisi partikel yang terbaik di antara partikel yang ada.



Gambar 1 Pergerakan partikel PSO

Kelebihan utama algoritma PSO jika dibandingkan dengan algoritma matematika dan teknik optimasi heuristik lainnya memiliki konsep yang sederhana, mudah diimplementasikan, serta lebih efisien dalam perhitungannya.

ANT COLONY ALGORITHM

Ant colony algorithm atau algoritma semut merupakan algoritma yang terinpirasi dari kumpulan/kelompok semut, di mana mereka menyelesaikan permasalahan yang ada dengan kerja sama *multi-agent* menggunakan komunikasi tidak langsung, akan tetapi dengan lingkungan sekitar. Semut saat berjalan melepaskan sejumlah feromon, semut senang berjalan mengikuti arah yang kaya akan feromon. Perilaku inilah yang membuktikan bahwa semut mampu menyesuaikan diri dengan perubahan lingkungan, seperti adanya hambatan baru yang mengganggu jalur terpendek yang mereka lalui.

Ilustrasi gangguan dapat dilakukan dengan diberikan benda pada jalur terpendek yang biasa mereka lalui, ketika benda tersebut diletakkan semut akan berjalan menyebar ke kanan dan ke kiri. Setelah beberapa lama, semut yang menemukan jalur terpendek akan membuat jejak dengan feromon dan dikuatkan kembali dengan jauh lebih cepat dari jalur yang lebih jauh. Hal inilah yang akan membuat semut lainnya lebih memilih untuk mengikuti semut yang telah menemukan jalur terpendek. Dari ilustrasi gangguan yang dialami semut inilah algoritma semut mulai dikembangkan untuk memecahkan permasalahan yang ada di dunia nyata. Optimasi algoritma semut merupakan teknik peluang untuk memecahkan masalah komputasi yang dapat dikurangi dengan mencari jalur yang tepat.

Proses utama yang ada dalam teknik algoritma semut ialah dengan simulasi berulang, di mana semut akan menyimpan jalur yang mereka lalui. Jalur yang jarang atau tidak dilalui akan menguap atau menghilang karena waktu yang ditempuh lebih lama (feromon mudah menguap). Jalur yang sering dilalui akan memiliki feromon kuat karena waktu tempuh yang

pendek, sehingga semut akan lebih memilih feromon yang kuat atau jalur terpendek. Mekanisme algoritma untuk memecahkan masalah bisa dikatakan terlalu kompleks untuk ditangani oleh semut tunggal. Sistem ini didasarkan atas umpan balik untuk menarik semut lain, sehingga akan memperkuat jalur. Sistem umpan balik akan mengantisipasi adanya jarak jalur yang sama di mana jalur akan lebih diperkuat, sehingga dapat memilih jalur yang akan dilalui. Proses penguapan atau hilangnya feromon yang akan menghindari kemungkinan terjebak pada lokal optimum.

BAB 2
OPTIMASI MULTI-OBJEKTIF

Optimasi pada dasarnya dikembangkan dari adanya permasalahan yang ingin diselesaikan secara efektif dan efisien. Solusi yang dimiliki bisa bernilai minimum atau pun maksimum. Akan tetapi permasalahan optimasi dalam kenyataannya tidak hanya ingin memiliki solusi yang meminimumkan atau memaksimumkan, bisa saja solusi yang diharapkan memiliki lebih dari satu fungsi, yaitu maksimum-maksimum, minimum-minimum, atau maksimum-minimum. Banyak fungsi tujuan harus diselesaikan secara simultan dan biasanya kebanyakan bertentangan. Pencarian kumpulan metode yang memiliki ukuran minimum telah dilakukan, akan tetapi akurasi yang dimiliki maksimum. Konsep dari optimasi multi-objektif ini dengan mendapatkan solusi dengan mengukur serta membandingkan antara satu solusi dan solusi yang lainya.

MULTI-OBJECTIVE OPTIMIZATION

Multi-objective optimization atau juga dikenal dengan *multi-criteria optimization* merupakan suatu persoalan optimasi dengan fungsi tujuan lebih dari satu (ganda), di antara fungsi-fungsi tujuan ini sangat mungkin terjadi konflik.

Keberadaan masalah optimasi multi-tujuan ini tidak dapat disangkal atau dihindari lagi. Ada begitu banyak permasalahan, baik di dunia nyata maupun secara matematis, yang secara alamiah memiliki lebih dari satu

tujuan yang ingin dioptimasikan. Bahkan, sangat mungkin terjadi konflik antarsatu atau beberapa tujuan dengan tujuan lainnya.

Upaya untuk menemukan solusi optimal dari suatu permasalahan multi-tujuan telah banyak dilakukan. Sayangnya, banyak di antara penelitian tersebut yang dalam proses optimasinya melakukan konversi multi-tujuan ke tujuan tunggal. Bahkan tidak jarang, penelitian justru berfokus pada perbaikan teknik konversi.

PARETO OPTIMAL

Pareto optimal atau Optimalitas Pareto adalah konsep optimalitas yang dikemukakan oleh Pareto. Konsep ini secara sederhana mengatakan bahwa sebuah titik mencapai optimalitas apabila tidak ada titik lain yang mampu memperbaiki optimalitas yang sudah tercapai.

Dalam bahasa matematis, untuk masalah optimasi multi-tujuan tanpa pembatas:

$$\min\left(f_1(x), f_2(x), \dots, f_k(x)\right),$$

dengan $k \geq 2$ dan $x \in X$, solusi $x^* \in X$ disebut optimal Pareto jika dan hanya jika:

$$\nexists x \in X, f_i(x) \leq f_i(x^*), \forall_i \in \{1,2,\dots,n\}$$

dengan $f_i(x) < f_j(x^*)$ berlaku untuk paling sedikit satu index-*j*.

Optimal Pareto menurut definisi ini dikenal sebagai bentuk tegas (*strict form*) atau bentuk kuat (*strong form*). Sementara bentuk lemah (*weak form*) jika yang berlaku adalah tanda "<", menggantikan tanda "≤" pada persamaan di atas:

$$\nexists x \in X, f_i(x) < f_i(x^*), \forall_i \in \{1,2,\dots,n\}$$

Dengan demikian, optimal Pareto yang tidak disebut khusus berarti bersifat tegas atau kuat, merupakan bagian (*subset*) dari optimal Pareto yang bersifat lemah (*weakly Pareto optimal*). Untuk kasus maksimasi, tanda tinggal dibalik.

DOMINASI

Dalam kasus meminimumkan, jika terdapat $a, b \in X$ dan $f(a) < f(b)$, dikatakan bahwa *b* didominasi *a* dan *f(b)* didominasi *f(a)*. pendominasian di sini bersifat tegas (*strictly domination*) atau kuat (*strongly domination*). Solusi *b* dikatakan didominasi secara lemah (*weakly domination*) oleh *a*, jika $f(a) \leq f(b)$. Jika solusi *a* mendominasi secara tegas solusi *b*, solusi *a* juga mendominasi secara lemah solusi *b*, tetapi tidak sebaliknya. Jika solusi *a* mendominasi solusi *b*, secara intuisi *a* lebih baik daripada *b*, untuk kasus maksimasi tanda tinggal dibalik (Deb *et al.* 2002; Deb 2001).

Himpunan (set) dari *non-dominated solutions* disebut *non-dominated set*. Himpunan ini memenuhi dua kondisi berikut.

1. Setiap pasang solusi dalam himpunan ini tidak boleh saling mendominasi.
2. Setiap solusi yang berada di luar himpunan didominasi paling sedikit oleh satu anggota himpunan (Deb 2004).

PARETO SET

Apabila *non-dominated set* yang diperoleh berasal dari keseluruhan ruang pencarian ($x \in X$), himpunan ini dikenal pula sebagai *Pareto set* atau *Pareto frontier*. Jadi, semua solusi *x* dalam *Pareto set* atau lengkapnya *set of*

pareto optimal solutions memenuhi konsep Optimalitas Pareto. Dengan demikian, seluruh anggota *Pareto set* adalah yang terbaik, yang paling optimal sesuai fungsi-fungsi tujuan yang ada.

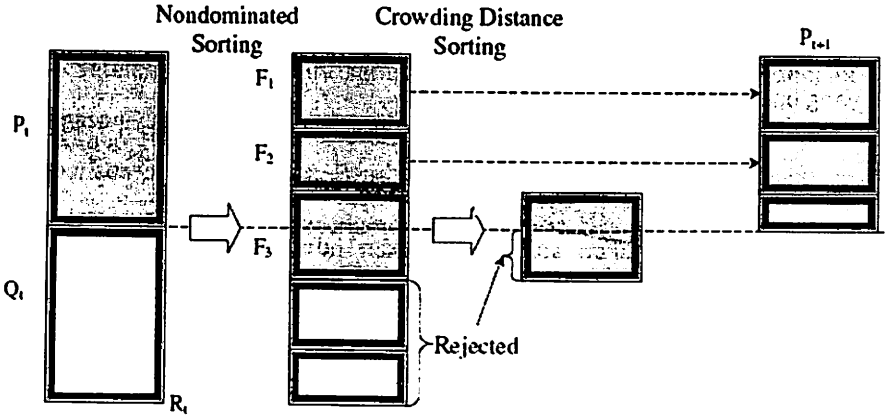
NON-DOMINATED SORTING GENETIC ALGORITHM II (NSGA-II)

Model NSGA-II ini efisien dari segi komputasi dan menggunakan operator *elitism* dan *crowded comparison* yang mempertahankan keragaman tanpa menggunakan berbagai parameter tambahan (Deb *et al.* 2002; Deb 2001).

Pada Gambar 2, generasi ke-*t*, populasi *offspring* Q_t dibentuk dari populasi P_t menggunakan operator-operator evolusioner. Kemudian P_t dan Q_t digabungkan menjadi satu populasi, yaitu R_t.

Setiap individu dalam populasi R_t kemudian diklasifikasikan berdasarkan *non-domination level* ke dalam kelas atau *front* yang berbeda: F_1, F_2, ..., F_n. Dengan demikian F_1 terdiri atas solusi-solusi atau individu-individu terbaik dalam populasi gabungan R_t. Sementara yang terburuk berada pada kelas paling buncit, yaitu F_n.

Selanjutnya NSGA-II menerapkan *niching* berupa penambahan *crowding distance* kepada setiap individu. Tujuannya adalah mempertahankan keragaman populasi dan membantu algoritma untuk mengeksplorasi ruang pencarian.



Gambar 2 Prinsip kerja NSGA-II (Deb 2001)

INISIALISASI POPULASI

Algoritma Genetika (GA) dimulai dengan sebuah grup dari kromosom yang disebut populasi. Populasi diinisialisasi berdasarkan kisaran permasalahan dan kendala jika ada. Populasi memiliki N_{pop} kromosom dan berbentuk matriks dengan ukuran N_{pop} x N_{bits} serta diisi dengan bilangan acak (Bagchi 1999).

NON-DOMINATED SORT

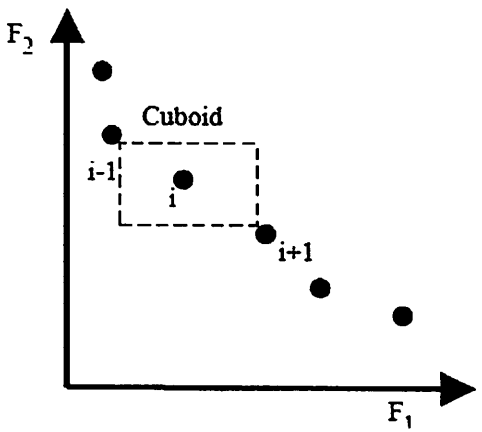
Setiap anggota seperti vektor yang tidak didominasi oleh anggota lainnya disebut *non-dominated.* Jika tujuannya adalah memaksimalkan fungsi, didefinisikan sebagai titik terdominasi jika komponen terkait tidak lebih baik dari titik tidak terdominasi. Solusi optimal yang dihasilkan dalam permasalahan *multi-objective optimization* adalah solusi *non-dominated.*

Ide di balik prosedure *non-dominated sorting* adalah metode seleksi *ranking* yang digunakan sebagai penegas titik yang baik dan metode *niche*

yang digunakan untuk menjaga kestabilan *subpopulations* berada pada titik yang baik.

CROWDING DISTANCE

Setelah dilakukan *sort* berdasarkan nondominasi, maka terdapat kumpulan individu yang mendominasi terhadap individu lainnya. Kemudian kumpulan individu-individu tersebut disebut sebagai *front*. Setelah didapatkan individu-individu di dalam satu *front*, langkah selanjutnya adalah melakukan proses *crowding*. *Crowding distance* merupakan kerapatan dari suatu *front* yang ada di sekitar tetangga terdekatnya, kemudian jarak dari masing-masing solusi tersebut berdasarkan tetangga terdekatnya. *Crowding distance* didapatkan dari solusi-solusi yang terbaik pada proses *non-dominated sort,* kemudian membandingkan *crowding distance* dengan dua individu pada *front* yang berbeda dan mempunyai nilai kurang dominan, berikut gambar tentang proses *crowding distance*.



Gambar 3 Proses *crowding distance*

SELEKSI

Setelah individu diurutkan berdasarkan nondominasi dan dengan penugasan *crowding distance*, seleksi dilakukan menggunakan *crowded-comparison-operator* ($\prec_n$). Pembanding antar-individu untuk dilakukan menyeleksi orang tua berdasarkan berikut ini.

(1) *Rank* nondominasi p_{rank} sebagai contoh individu dengan *front* F_i akan memiliki *rank*-nya yaitu $p_{rank} = i$.

(2) *Crowding distance* $F_i(d_j)$

$p \prec_n q$ jika :

- $p_{rank} < q_{rank}$ atau
- jika p dan q memiliki *front* F_i yang sama, $F_i(d_p) > F_i(d_q)$ yaitu nilai *crowding distance* pilih yang lebih besar.

Artinya, antara dua solusi dengan peringkat nondominasi berbeda maka yang dipilih adalah titik dengan peringkat yang lebih rendah. Jika tidak, yaitu kedua titik memiliki peringkat yang sama, maka kita lebih suka titik yang terletak di wilayah dengan jumlah yang lebih kecil dari titik (ukuran dari balok *inclosing* itu lebih besar) atau lebih tepatnya *crowding distance* besar. Proses seleksi individu ini termasuk ke dalam *binary selection tournament*.

GENETIC OPERATOR

Operator genetik yang digunakan setelah proses evaluasi tahap pertama membentuk populasi baru dari generasi sekarang. Operator-operator tersebut adalah pindah silang (*crossover),* mutasi, dan elitisme.

Pindah silang adalah sebuah operator genetik yang menggabungkan (*mate*) dua individu orang tua yang akan menghasilkan dua buah anak atau keturunan. Jenis pindah silang ada banyak disesuaikan dengan kebutuhan, seperti *one point crossover*, *two point crossover*, PMX, dan lain-lain. Menurut Deb *et al.* (1994 dan 2001), proses pindah silang pada Algoritma Genetika menggunakan SBX (*Simulated Binary Crossover*). Rumus yang digunakan ketika menggunakan SBX sebagai berikut.

$$b = \begin{cases} (2 * r)^{\frac{1}{\mu+1}} & jika\, r \leq 0.5 \\ \left(\frac{1}{2 * (1-r)}\right)^{\frac{1}{\mu+1}} & jika\, r > 0.5 \end{cases}$$

$$child_1(j) = \frac{1}{2}((1+b) * parent_1(j) + (1-b) * parent_2(j))$$

$$child_2(j) = \frac{1}{2}((1-b) * parent_1(j) + (1+b) * parent_2(j))$$

dengan:

r : *Random number* (bilangan acak [0 1])

μ : Operator pindah silang

j : Menggambarkan representasi individu

Ide di balik proses pindah silang ialah kromosom yang baru mungkin lebih baik dari orang tuanya jika kromosom yang baru mengambil karakteristik yang baik dari masing-masing orang tuanya.

Mutasi adalah proses genetik yang merubah nilai satu atau lebih gen di sebuah kromosom dalam populasi. Tipe proses mutasi dalam algoritma genetika pun banyak, seperti *bit flip mutation*, *boundary mutation*, *non-uniform*, *uniform*, dan lain-lain. Dalam kasus *multi-objective genetic algorithm*, proses mutasi yang biasa diterapkan oleh Deb *et al.* (2002) dan

Deb (2001), Raghuwansi dan Kakde (2004) ialah *polynomial mutation*. Rumus yang biasa digunakan dalam *polynomial mutation* adalah:

$$b = \begin{cases} (2 * r)^{\frac{1}{\eta+1}} - 1 & jika\, r \leq 0.5 \\ 1 - (2 * (1-r))^{\frac{1}{\eta+1}} & jika\, r > 0.5 \end{cases}$$

$$child(j) = parent(j) + d$$

dengan:

r : *Random number* (bilangan acak [0 1])

η : Operator mutasi

j : Menggambarkan representasi individu

Mutasi adalah bagian penting dalam pencarian genetik yang membantu mencegah populasi dari lokal optima yang tidak berubah.

REKOMBINASI DAN SELEKSI

Populasi anak yang dikombinasikan dengan populasi generasi sekarang dan seleksi dilakukan untuk mengatur individu-individu dari generasi berikutnya. Karena semua individu terbaik sebelumnya dan saat ini ditambahkan dalam populasi, maka elitisme akan terjamin. Populasi sekarang diurutkan berdasarkan nondominasi. Generasi baru ini diisi oleh *front* masing-masing kemudian sampai ukuran populasi melebihi ukuran populasi saat ini. Jika dengan menambahkan semua individu di *front* F_j populasi melebihi N, individu di *front* F_j dipilih berdasarkan *crowding distance* dalam urutan sampai ukuran populasi *N*.

BAB 3
CONTOH KASUS MASALAH OPTIMASI

Algoritma Genetika Multi-Objektif memiliki banyak perkembangan dalam algoritmanya, salah satunya menggunakan algoritma dominasi atau lebih dikenalnya *Non-Dominated Sorted Genetic Algorithm* (NSGA-II). Pada dasarnya, kebanyakan Algoritma Genetik Multi-Objektif (ALGEMO) menggunakan konsep dominasi pada ruang pencariannya, maka dari itu pada buku ini akan menggunakan NSGA-II untuk contoh aplikasinya.

INISIALISASI POPULASI

Inisialisasi Populasi berfungsi untuk membentuk populasi setiap individu dalam merepresentasikan bentuk kromosom yang akan mempermudah menemukan solusi optimasi yang diinginkan. Representasi kromosom merupakan cara untuk mengkodekan suatu alternatif solusi itu menjadi kromosom yang akan diproses menggunakan GA. Setiap kromosom memiliki nilai gen yang disebut *allele*, di mana *allele* akan menentukan bentuk hasil akhir yang diinginkan.

Banyak cara untuk mengimplementasikan fungsi inisialisasi populasi dalam program, baik dengan cara manual ataupun dengan fungsi *library* dari *software*. Pada buku ini menggunakan *software* MATLAB.

Fungsi inisialisasi populasi manual didapatkan dengan *listing* program berikut.

```
Function populasi = inisialisasi_populasi(ukpop,jmlgen)
%membentuk matrik bilangan random dengan ukuran ukpop x jmlgen
```

```
bil_rand = rand(ukpop,jmlgen);
for i = 1 : ukpop
      for j = 1 : jmlgen
      if bil_rand(i,j)>0.5
        populasi(i,j) = 1;
      else
        populasi(i,j) = 0;
      end
end
end
```

Fungsi inisialisasi populasi dapat juga menggunakan fungsi randint dari MATLAB yang dapat membangun bilangan skalar 0 atau 1 secara *random* dengan peluang yang sama.

```
Function populasi = inisialisasi_populasi(ukpop,jmlgen)
%%
%Membentuk populasi sebanyak ukuran populasi dan
%jumlah gen yang sudah ditentukan, dengan nilai gen %berisi bilangan
biner
%
%INPUT:
%ukpop : ukuran populasi, banyaknya individu dalam %populasi
%jumgen : jumlah gen dalam 1 kromosom
%
%OUTPUT
%populasi = kumpulan kromosom, dengan bentuk matriks
%ukuran ukpop x jumgen
%%

populasi = randint(ukpop,jmlgen);
```


Contoh Hasil Inisialisasi Populasi (5, 10)

1	0	1	1	1	1	1	1	0	0
1	1	0	0	0	1	1	1	0	1
1	0	1	1	1	0	0	0	1	0
0	0	0	0	1	0	1	1	1	0
0	0	1	0	1	0	0	1	0	1

DEKODE KROMOSOM

Fungsi dekode kromosom ini bertujuan untuk mengonversi nilai skalar biner 0 dan 1 menjadi bilangan *real* berdasarkan batas bawah dan atas yang ditentukan. Decode kromosom ini akan menghasilkan nilai variabel x1 dan variabel x2 dari setiap kromosom, dengan pembagian jumlah gen. Nilai variabel x1 dan x2 memiliki nilai batas berdasarkan fungsi objektif yang akan dilakukan, contoh nilai batas variabel (Arkeman dan Tamura 2007):

Keterangan: $0.1 \leq x_1 \leq 1$

$0 \leq x_2 \leq 5$

Hasil dari konversi nilai biner ke *real* akan dilakukan fungsi *min-max normalization* untuk menentukan nilai asli dari variabel tersebut. Fungsi *min-max normalization* yang digunakan:

$$newv = \frac{v - min_x}{max_x - min_x} \cdot (newmax_x - newmin_x) + newmin_x$$

Listing program dekode kromosom

```
Function dpopulasi = dekode_kromosom(populasi,nbit,a1,b1,a2,b2)
%% dekode kromosom merupakan fungsi konversi bilangan scalar 0 dan 1
%menjadi bilangan real dengan batas atau range nilai yang telah
ditentukan
%
%INPUT :
%populasi : populasi yang dibentuk dari inisialisasi
%populasi
%nbit : jumlah bit/gen untuk merepresentasikan 1 variabel
%b1 : batas bawah dari nilai variable x1
%a1 : batas atas dari nilai variable x1
%
%b2 : batas bawah dari nilai variable x2
%a2 : batas atas dari nilai variable x2
%
%OUTPUT:
%dpopulasi : hasil decode atau konversi nilai gen
%%

[m n] = size(populasi);
bobot_var = sort(2.^((1:nbit)-1),'descend');
minx = 0;
maxx = sum(bobot_var)
for i = 1 : m
     x1 = populasi(i,1:nbit).*bobot_var;
     x2 = populasi(i,nbit+1:n).*bobot_var;
     dpopulasi(i,1) = ((x1-minx)/(maxx-minx))*(a1-b1)+b1;
     dpopulasi(i,2) = ((x2-minx)/(maxx-minx))*(a2-b2)+b2;
   end
end
```


Contoh proses dan hasil dekode kromosom

Konversi Biner ke Bil. Real										Var 1 (x_1)	Var 2 (x_2)
16	0	4	2	1	16	8	4	0	0	23	28
16	8	0	0	0	16	8	4	0	1	24	29
16	0	4	2	1	0	0	0	2	0	23	2
0	0	0	0	1	0	8	4	2	0	1	14
0	0	4	0	1	0	0	4	0	1	5	5

	Var 1 (x_1)	Var 2 (x_2)
	0.87	4.50
Konversi	0.88	4.67
Var	0.87	0.17
Range	0.50	2.17
	0.57	0.67

EVALUASI INDIVIDU

Fungsi evaluasi individu ialah fungsi tujuan dari permasalah optimasi yang ingin dicapai. Fungsi ini merupakan salah satu tahap yang menentukan hasil dari algoritma genetika.

Fungsi evaluasi yang digunakan (Arkeman dan Tamura 2007):

Minimalisasi : $f_1(x) = x_1$

Minimalisasi : $f_2(x) = \frac{(1+x_2)}{x1}$

Keterangan : $0.1 \leq x_1 \leq 1$

$0 \leq x_2 \leq 5$

Hasil dari dekode kromosom yang merupakan nilai konversi variabel *range* akan dimasukan ke dalam fungsi evaluasi individu, di mana *listing* program yang digunakan:

```
Function f = evaluasi_individu(populasi,dpopulasi,jmlgen)
%% fungsi evaluasi individu berperan untuk
%
%INPUT :
%dpopulasi : populasi bilangan real yang dibentuk dari %hasil dekode
kromosom
```

```
%
%OUTPUT:
%fitness : hasil evaluasi setiap individu

[m n] = size(dpopulasi);
for i = 1 : m
   x1 = dpopulasi(i,1);
   x2 = dpopulasi(i,2);
   fitness(i,1) = x1;
   fitness(i,2) = (1+x2)/x1;
end
populasi(:,jmlgen+1:jmlgen+2) = fitness;
f = populasi;
```

Contoh hasil dari evaluasi individu:

X_1	X_2
0.87	4.50
0.88	4.67
0.87	0.17
0.50	2.17
0.57	0.67

Hasil perhitungan

$F(x_1)$	$F(x_2)$
0.87	6.35
0.88	6.42
0.87	1.35
0.5	6.33
0.57	2.94

NON-DOMINATED SORTING DAN *CROWDING DISTANCE*

Hasil evaluasi individu akan digunakan untuk mengetahui *rank* berdasarkan dominasinya. Kemudian hasil dominasi individu akan dilakukan perhitungan *crowding distance* dan akan menghasilkan urutan individu baru. *Crowding distance* dilakukan untuk memberikan kemampuan Algoritma Genetika untuk membedakan antar-individu walaupun dalam *rank* yang sama dan mencegah terjadinya keseragaman hasil.

SESHADRI (2009) merujuk pada Kanpur Genetic Algorithms Laboratory:

```
function f = non_domination_sort_mod(x, M, V)
[N, m] = size(x);
clear m
% inisialisasi awal front = 1
front = 1;

F(front).f = [];
individual = [];

%% Non-Dominated sort.

for i = 1 : N
    % Jumlah yang mendominasi individu ke i
    individual(i).n = 0;
    % Individu lainnya yang terdominasi Individu ke i
    individual(i).p = [];
    for j = 1 : N
        dom_less = 0;
        dom_equal = 0;
        dom_more = 0;
        for k = 1 : M
            if (x(i,V + k) > x(j,V + k))
                dom_more = dom_more + 1;
            elseif (x(i,V + k) == x(j,V + k))
                dom_equal = dom_equal + 1;
            else
                dom_less = dom_less + 1;
            end
        end
        if dom_less == 0 && dom_equal ~= M
            individual(i).n = individual(i).n + 1;
        elseif dom_more == 0 && dom_equal ~= M
            individual(i).p = [individual(i).p j];
        end
    end
    if individual(i).n == 0
        x(i,M + V + 1) = 1;
        F(front).f = [F(front).f i];
    end
```

```
end
% Proses mencari front selanjutnya
while ~isempty(F(front).f)
   Q = [];
   for i = 1 : length(F(front).f)
       if ~isempty(individual(F(front).f(i)).p)
      for j = 1 : length(individual(F(front).f(i)).p)
      individual(individual(F(front).f(i)).p(j)).n = ...
      individual(individual(F(front).f(i)).p(j)).n - 1;
       if individual(individual(F(front).f(i)).p(j)).n == 0
       x(individual(F(front).f(i)).p(j),M + V + 1) = ...
                        front + 1;
                    Q = [Q individual(F(front).f(i)).p(j)];
            end
          end
       end
   end
   front =  front + 1;
   F(front).f = Q;
end

[temp,index_of_fronts] = sort(x(:,M + V + 1));
for i = 1 : length(index_of_fronts)
    sorted_based_on_front(i,:) = x(index_of_fronts(i),:);
end
current_index = 0;

%% Crowding distance

% Proses mencari jarak crowding distance setiap individu pada
% tiap front
for front = 1 : (length(F) - 1)

    distance = 0;
    y = [];
    previous_index = current_index + 1;
    for i = 1 : length(F(front).f)
        y(i,:) = sorted_based_on_front(current_index + i,:);
    end
    current_index = current_index + i;
    % Mengurutkan berdasarkan nilai evaluasi objective
    sorted_based_on_objective = [];
```



```
    for i = 1 : M
        [sorted_based_on_objective, index_of_objectives] = ...
            sort(y(:,V + i));
        sorted_based_on_objective = [];
        for j = 1 : length(index_of_objectives)
            sorted_based_on_objective(j,:)                              =
y(index_of_objectives(j),:);
        end
        f_max = ...
            sorted_based_on_objective(length(index_of_objectives), V +
i);
        f_min = sorted_based_on_objective(1, V + i);
        y(index_of_objectives(length(index_of_objectives)),M + V + 1 +
i)...
            = Inf;
        y(index_of_objectives(1),M + V + 1 + i) = Inf;
         for j = 2 : length(index_of_objectives) - 1
            next_obj  = sorted_based_on_objective(j + 1,V + i);
            previous_obj  = sorted_based_on_objective(j - 1,V + i);
            if (f_max - f_min == 0)
                y(index_of_objectives(j),M + V + 1 + i) = Inf;
            else
                y(index_of_objectives(j),M + V + 1 + i) = ...
                      (next_obj - previous_obj)/(f_max - f_min);
            end
        end
    end
    distance = [];
    distance(:,1) = zeros(length(F(front).f),1);
    for i = 1 : M
        distance(:,1) = distance(:,1) + y(:,M + V + 1 + i);
    end
    y(:,M + V + 2) = distance;
    y = y(:,1 : M + V + 2);
    z(previous_index:current_index,:) = y;
end
f = z();
```

Contoh hasil kalkulasi menggunakan *non-dominated* dan *crowding distance*

indv 1		indv 2	result	
0.87	<	0.5	0	1
1.35	<	6.33	1	0
			equal	

Jika result $indv^{(1)} = indv^{(2)}$, maka *non-inferior / equal dominated*

Jika result $indv^{(1)} > indv^{(2)}$, maka *dominated*

Jika result $indv^{(1)} < indv^{(2)}$, maka *non-dominated*

indv 1		indv 2	indv 3	indv 4	indv 5
0.87	<	0.88	0.87	0.5	0.57
6.35	<	6.42	1.35	6.33	2.94
		nondominated	*dominated*	*dominated*	*dominated*

Contoh:

Indv 1		Indv3	Indv1	Indv 3
0.87	<	0.87	0	0
6.35	<	1.35	0	1
			Non-dominated	*dominated*

Sehingga bisa mendapatkan jumlah *dominated* hasil dari perbandingan antar individu.

$indv^{(1)} = \{indv^{(3)}, indv^{(4)}, indv^{(5)}\} = 3$

$indv^{(2)} = \{indv^{(1)}, indv^{(3)}, indv^{(4)}, indv^{(5)}\} = 4$

$indv^{(3)} = \{....\} = 0$

$indv^{(4)} = \{....\} = 0$

$indv^{(5)} = \{....\} = 0$

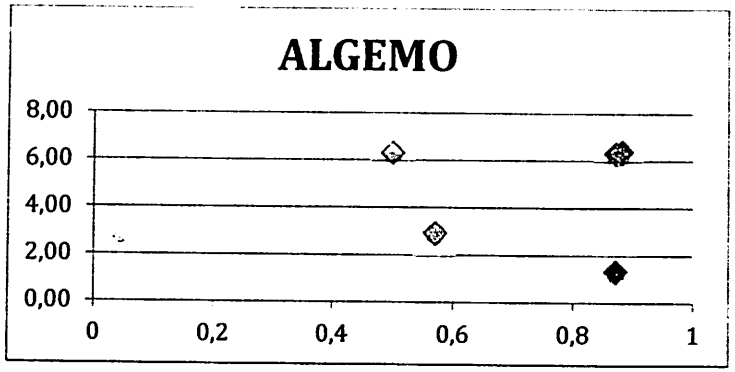
Jumlah *dominated* akan dilakukan *sorting* berdasarkan jumlah yang paling sedikit, sehingga didapatkan individu hasil *nondominated-sort.*

	$F(x_1)$	$F(x_2)$	Rank
$indv^{(1)}$	0.87	6.35	2
$indv^{(2)}$	0.88	6.42	3
$indv^{(3)}$	0.87	1.35	1
$indv^{(4)}$	0.5	6.33	1
$indv^{(5)}$	0.57	2.94	1

Sorting

	$F(x_1)$	$F(x_2)$	Rank
$indv^{(1)}$	0.87	1.35	1
$Indv^{(2)}$	0.5	6.33	1
$indv^{(3)}$	0.57	2.94	1
$indv^{(4)}$	0.87	6.35	2
$indv^{(5)}$	0.88	6.42	3

Proses *crowding distance* dilakukan pada setiap individu yang memiliki *rank* sama.



Gambar 4 Grafik *pareto front*

Perhitungan *crowding distance* tidak hanya didasarkan pada *rank* yang sama, tetapi juga berdasarkan posisi *frontier* individu, sehingga individu tersebut akan diurutkan kembali. Di mana pengurutannya ditentukan dari nilai x_1 yang menentukan posisi awal per *frontier.*

	$F(x_1)$	$F(x_2)$	Rank
$indv^{(1)}$	0.87	1.35	1
$indv^{(2)}$	0.5	6.33	1
$indv^{(3)}$	0.57	2.94	1
$indv^{(4)}$	0.87	6.35	2
$indv^{(5)}$	0.88	6.42	3

Sorting Based on frontier Position m=1

	$F(x_1)$	$F(x_2)$	Rank
$indv^{(1)}$	0.5	6.33	1
$indv^{(2)}$	0.57	2.94	1
$indv^{(3)}$	0.87	1.35	1
$indv^{(4)}$	0.87	6.35	2
$indv^{(5)}$	0.88	6.42	3

Berdasarkan Gambar 4 maupun contoh tabel, didapat bahwa $indv^{(2)}$ berada pada posisi pertama *crowding distance* yang kemudian menjadi $indv^{(1)}$, sedangkan individu yang berasal dari $indv^{(1)}$ merupakan posisi terakhir pada *frontier* yang sama, yaitu $indv^{(3)}$. $Indv^{(1)}$ dan $indv^{(3)}$ hasil pengurutan berdasarkan posisi akan memiliki jarak infinitif (∞). Jarak infinitif (∞) dalam *crowding distance* adalah jarak yang tidak terjangkau antara individu pada posisi pertama dan individu pada posisi terakhir dalam *frontier* yang sama.

$$1_{C.D} = l_{C.D} = \infty$$

Rumus *crowding distance* yaitu:

$$i_{C.D} = \sum_m \left(\frac{f[i+1]_m - f[i-1]_m}{f_m^{max} - f_m^{min}} \right)$$

Di mana perhitungan manual *crowding distance* sebagai berikut.

m = 1, yaitu fungsi $f(x_1)$

$$2_{C.D} = \frac{0.87 - 0.5}{0.87 - 0.5} = 1$$

dan

m = 2, yaitu fungsi $f(x_2)$, di mana:

Sorting Based on frontier Position m=2

	$F(x_1)$	$F(x_2)$	Rank
$indv^{(1)}$	0.87	1.35	1
$indv^{(2)}$	0.57	2.94	1
$indv^{(3)}$	0.5	6.33	1
$indv^{(4)}$	0.87	6.35	2
$indv^{(5)}$	0.88	6.42	3

$$2_{C.D} = \frac{6.33 - 1.35}{6.33 - 1.35} = 1$$

$1_{C.D} = 1 + 1 = 2$

sehingga *crowding distance* untuk $indv^{(3)}$ ialah 2.

Hasil populasi yang didapat dari *crowding distance* sebagai berikut.

	$F(x_1)$	$F(x_2)$	Rank	C.D
$indv^{(1)}$	0.87	1.35	1	∞
$indv^{(2)}$	0.5	6.33	1	∞
$indv^{(3)}$	0.57	2.94	1	2
$indv^{(4)}$	0.87	6.35	2	∞
$indv^{(5)}$	0.88	6.42	3	∞

SELEKSI

Seleksi dilakukan untuk memilih orang tua dari populasi yang terbaik. Fungsi seleksi yang dilakukan dalam kasus ini ialah menggunakan fungsi turnamen di mana prosesnya melihat dari kriteria *front/rank* dan jarak *crowding distance*. Individu yang terpilih secara acak akan dilakukan turnamen melihat minimum *front/rank*-nya, jika ternyata sama akan dilihat jarak maksimum *crowding distance* yang dimiliki.

Source code turnamen seleksi pada kasus ini:

```
for i = 1 : pool_size
    % memilih individu secara random dari ukuran tour_size
    for j = 1 : tour_size
        % memilih satu individu
        candidate(j) = round(pop*rand(1));
        % Array/list dimulai dari satu
        if candidate(j) == 0
            candidate(j) = 1;
        end
        if j > 1
            % memastikan tidak memilih individu yang sama
            while ~isempty(find(candidate(1 : j - 1) == candidate(j)))
                candidate(j) = round(pop*rand(1));
```

```
                if candidate(j) == 0
                    candidate(j) = 1;
                end
            end
        end
    end
    % kandidat
    % mengambil kriteria kandidat, front dan crowding distance
    for j = 1 : tour_size
        c_obj_rank(j) = chromosome(candidate(j),rank);
        c_obj_distance(j) = chromosome(candidate(j),distance);
    end
    % Mencari kandidat dengan rank terakhir
    min_candidate = ...
        find(c_obj_rank == min(c_obj_rank));
    % jika terdapat banyak kandidat
    % group individu yang memiliki jarak max.
    if length(min_candidate) ~= 1
        max_candidate = ...
        find(c_obj_distance(min_candidate)                    ==
max(c_obj_distance(min_candidate)));
        % ada banyak individu di rank yang sama dengan jarak max
        % berdasarkan jaraknya, hanya memilih satu
        if length(max_candidate) ~= 1
            max_candidate = max_candidate(1);
        end
        % menambah individu yang terseleksi ke mating pool
        f(i,:)                                                =
chromosome(candidate(min_candidate(max_candidate)),:);
    else
        % menambah kandidat individu pertama ke mating pool
        f(i,:) = chromosome(candidate(min_candidate(1)),:);
    end
end
```


Ilustrasi turnamen pada kasus ini sebagai berikut.

Populasi:

	$F(x_1)$	$F(x_2)$	Rank	C.D
$Indv^1$	0.87	1.35	1	∞
$Indv^2$	0.5	6.33	1	∞
$Indv^3$	0.57	2.94	1	2
$Indv^4$	0.87	6.35	2	∞
$Indv^5$	0.88	6.42	3	∞

Pemilihan acak

	Rank	C.D
$Indv^4$	2	~~∞~~
$Indv^5$	3	∞

	Rank	C.D
$Indv^3$	1	2
$Indv^2$	1	∞

Mating Pool

$Indv^4$
$Indv^3$

PINDAH SILANG DAN MUTASI

Pindah silang dan mutasi merupakan proses genetik yang ada pada algoritma genetika, di mana proses ini melibatkan parameter peluang pindah silang dan peluang mutasi. Konsep pindah silang yang digunakan pada contoh kali ini menggunakan konsep *one-point crossover*. Akan dipilih 2 induk hasil seleksi secara acak, yang kemudian akan ditentukan titik potong untuk dilakukan persilangan seluruh gen. Mutasi yang akan dilakukan ialah menggunakan *binary bitwise* di mana peluang yang dimiliki sebesar 1/M, M adalah banyaknya fungsi objektif yang digunakan.

Source code pindah silang dan mutasi pada kasus ini ialah:

```
function f  = genetic_operator(induk_chromosome, M, V, pc, pm)
 [N,m] = size(induk_chromosome);

p = 1;

%sebagai tanda proses genetika
was_crossover = 0;
was_mutation  = 0;
anak=[];

for i = 1 : N
    % berdasarkan peluang pindah silang
    if rand(1) < pc
        % inisialisasi awal anak = 0.
        anak_1 = [];
        anak_2 = [];
```

```
        % memilih secara acak induk 1
        induk_1 = round(N*rand(1));
        if induk_1 < 1
            induk_1 = 1;
        end
        % memilih secara acak induk ke 2
        induk_2 = round(N*rand(1));
        if induk_2 < 1
            induk_2 = 1;
        end
        % memastikan bahwa orang tua yang dipilih tidak sama
        while
isequal(induk_chromosome(induk_1,:),induk_chromosome(induk_2,:))
            induk_2 = round(N*rand(1));
            if induk_2 < 1
                induk_2 = 1;
            end
        end
      % mendapatkan nilai kromosom dari induk yang terpilih
        induk_1 = induk_chromosome(induk_1,:);
        induk_2 = induk_chromosome(induk_2,:);
        anak_1 = induk_1;
        anak_2 = induk_2;
        %menentukan titik potong secara acak, serta melakukan proses
pindah silang
        TP = 1 + fix(rand*(V-1));
        anak_1=[induk_1(1:TP) induk_2(TP+1:m)];
        anak_2=[induk_2(1:TP) induk_1(TP+1:m)];


        %memberi tanda terjadi proses pindah silang
        was_crossover = 1;
        was_mutation = 0;
    %dengan peluang mutasi yaitu 1/m
    else
        % memilih induk secara acak
        induk_3 = round(N*rand(1));
        if induk_3 < 1
            induk_3 = 1;
        end
        % mendapatkan nilai kromosom induk
        anak_3 = induk_chromosome(induk_3,:);
        % melakukan proses mutasi di induk yang terpilih, dengan
peluang mutasi gen pm = 1/m
        for j = 1 : V
           r(j) = rand(1);
           if r(j) < pm
               if anak_3(j) == 0
                   anak_3(j) = 1;
               else
                   anak_3(j) = 0;
               end
           end

        end

        % memberi tanda proses mutasi
        was_mutation = 1;
        was_crossover = 0;
    end
    % menggabungkan anak hasil mutasi dan pindah silang
    if was_crossover
```



```
        anak(p,:)   = anak_1;
        anak(p+1,:) = anak_2;
        was_cossover = 0;
        p = p + 2;
    elseif was_mutation

        anak(p,:)   = anak_3;
        was_mutation = 0;
        p = p + 1;
    end
end
f = anak;
```

Proses pindah silang dan mutasi dilakukan dengan menentukan nilai *random* awal. Nilai *random* tersebut kemudian dibandingkan dengan nilai peluang pindah silang. Jika nilai random lebih kecil dari peluang pindah silang, proses pindah silang akan dilakukan, tetapi jika tidak proses mutasi yang terlebih dahulu dilakukan.

Proses pindah silang yang dilakukan ilustrasinya dapat dilihat pada contoh berikut.

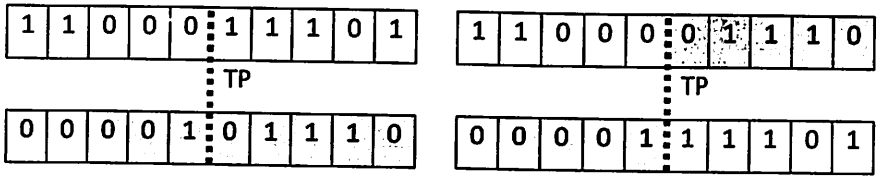
1	0	1	1	1	1	1	1	0	0
1	1	0	0	0	1	1	1	0	1
1	0	1	1	1	0	0	0	1	0
0	0	0	0	1	0	1	1	1	0
0	0	1	0	1	0	0	1	0	1

Memilih 2 induk secara *random*, dan pastikan tidak memilih induk yang sama

1	1	0	0	0	1	1	1	0	1

0	0	0	0	1	0	1	1	1	0

Proses pemilihan induk yang telah dilakukan akan dilanjutkan dengan proses pemilihan titik potong secara *random*. Titik potong digunakan untuk menentukan mulai dari urutan gen yang ditentukan akan dilakukan pertukaran gen atau pindah silang.



Awal mula proses mutasi pun hampir sama dengan proses pindah silang, yaitu dengan memilih satu induk secara acak. Individu induk yang telah terpilih akan dilakukan proses mutasi untuk tiap nilai gen yang terpilih (*allele*) secara *random*. Ilustrasi proses mutasi dapat dilihat pada contoh berikut.

ELITISME

Elitisme adalah proses pertahanan individu yang terbaik agar tidak terjadi penurunan kualitas nilai individu itu sendiri. Proses elitisme ini dilakukan dengan cara menjaga individu yang terbaik agar tidak mengalami proses genetik seperti pindah silang dan mutasi. Individu yang dipertahankan tersebut kemudian akan ditempatkan kembali ke dalam populasi hasil proses genetika, di mana proses penempatan kembali tersebut dapat dilakukan seperti *source code* berikut.

```
function f  = replace_chromosome(intermediate_chromosome, M, V,pop)
[N, m] = size(intermediate_chromosome);
% mendapatkan indeks dari populasi berdasarkan rank
[temp,index] = sort(intermediate_chromosome(:,M + V + 1));
clear temp m

% sort berdasarkan indeks
for i = 1 : N
    sorted_chromosome(i,:) = intermediate_chromosome(index(i),:);
end
```

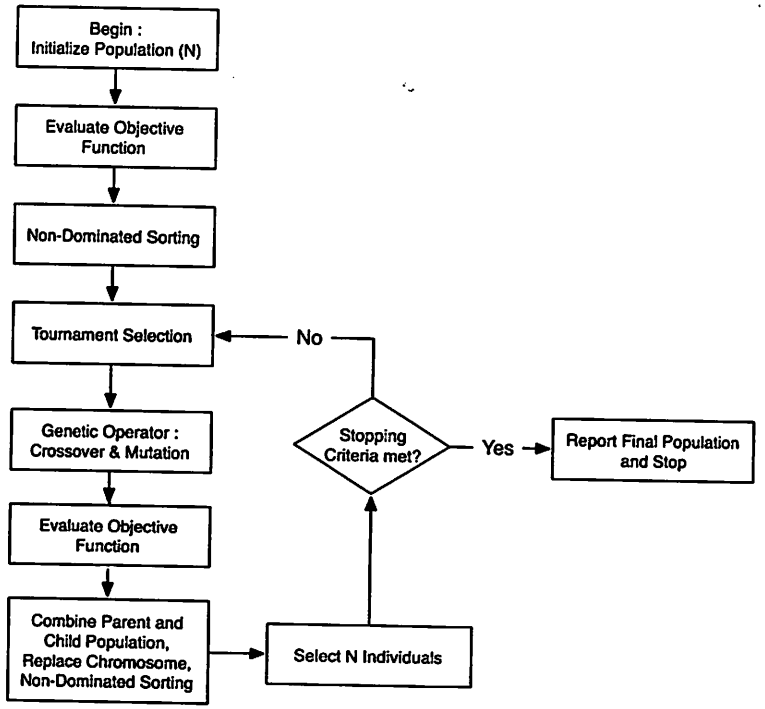


```
% menemukan maksimum rank dari populasi
max_rank = max(intermediate_chromosome(:,M + V + 1));

% menambahkan front berdasarkan rank dan crowding distance  sampai
populasi terisi

previous_index = 0;
for i = 1 : max_rank
    % mendapatkan index pada rank yang dituju misalkan yang terakhir,
elemen
    % yang terakhir dari variabel sorted_chromosome  dengan rank ke i.
    current_index = max(find(sorted_chromosome(:,M + V + 1) == i));

    % Mengecek, apakah populasi sudah terisi jika semua individual
dengan rank i
    % telah ditambankan ke dalam populasi
    if current_index > pop
% jika benar maka temukan jumlah individu di dalam rank ke i
        remaining = pop - previous_index;

        % Mendapatkan informasi mengenai individu di dalam rank ke i
        temp_pop = ...
            sorted_chromosome(previous_index + 1 : current_index, :);

% Mengurutkan individu dengan rank ke i dengan urutan descending
        % berdasarkan nilai crowding distance-nya
        [temp_sort,temp_sort_index] = ...
            sort(temp_pop(:, M + V + 2),'descend');

% Mulai mengisikan individual ke dalam populasi dengan cara
        %.descending sampai populasi tersebut terisi
        for j = 1 : remaining
            f(previous_index + j,:) = temp_pop(temp_sort_index(j),:);
        end
        return;
    elseif current_index < pop
% menambahkan semua individu rank ke i ke dalam populasi
        f(previous_index + 1 : current_index, :) = ...
            sorted_chromosome(previous_index + 1 : current_index, :);
    else
% Menambah semua individu rank ke i ke dalam populasi
        f(previous_index + 1 : current_index, :) = ...
```

```
            sorted_chromosome(previous_index + 1 : current_index, :);
        return;
    end

    % Mendapatkan indeks kemudian ditambahkan sebagai individu
terakhir
    previous_index = current_index;
end
```

NSGA-II

Pemanggilan fungsi-fungsi yang telah dibuat sebelumnya haruslah membuat program utama agar fungsi-fungsi tersebut dapat berjalan sesuai dengan tahapan dari NSGA-2 itu sendiri seperti pada Gambar 5.



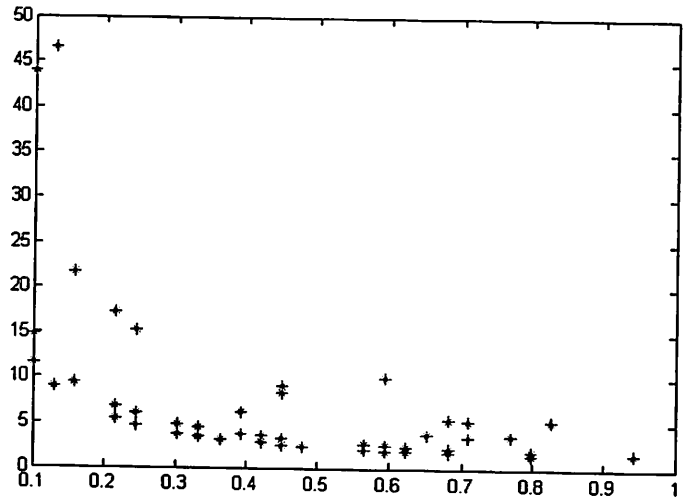
Gambar 5 Proses NSGA-II

Source code **yang digunakan untuk aplikasi NSGA-II sederhana dapat dilihat berikut.**

```
        function nsga2()

        %% inisialisasi variabel

        ukpop = 100;          %jumlah individu dalam 1 populasi
        nvar = 2;                    %jumlah variabel pada fungsi yang
dioptimasi
        nbit = 5;             %jumlah gen setiap variabel
        jmlgen = nvar*nbit;   %jumlah gen dalam setiap kromosom
        M = 2;                %jumlah objective
        Pc = 0.9;             %peluang pindah silang
        Pm = 1/M;             %peluang mutasi dalam 1 individu
        a1 = 1;               %batas atas variabel 1
        b1 = 0.1;             %batas bawah variabel 1
        a2 = 5;               %batas atas variabel 2
        b2 = 0;               %batas bawah variabel 2
        maxgenerasi = 500;    %jumlah generasi

        %% visualisasi grafik

        hfig = figure;
        hold on
        title ('Multi-Objective Genetic Algorithm');
        set(hfig,'position',[50,50,600,400]);
        set(hfig,'DoubleBuffer','on');
        axis([1 maxgenerasi 0 0.01]);
        hbestplot = plot(1:maxgenerasi, zeros(1,maxgenerasi));
        xlabel('x1');
        ylabel('x2');
        hold off
        drawnow

        %% inisialisasi populasi
        populasi = inisialisasi_populasi(ukpop,jmlgen);

        %% evaluasi individu
        x = dekode_kromosom(populasi,jmlgen,nbit, a1, b1, a2, b2);
        chromosome = evaluasi_individu(populasi,x,jmlgen);

        %% non-dominated
        chromosome = non_domination_sort_mod(chromosome, M, jmlgen);

        %% proses evaluasi
        for i = 1 : maxgenerasi
            %% seleksi
            matingpool = round(ukpop);
            tournament = 2;
            parent_chromosome     =     tournament_selection(chromosome,
matingpool, tournament);
            offspring_chromosome                                        =
genetic_operator(parent_chromosome,M,jmlgen, Pc,Pm);

            %% evaluasi individu
            x                                                           =
dekode_kromosom(offspring_chromosome,jmlgen,nbit,a1,b1,a2,b2);
            offspring_chromosome                                        =
evaluasi_individu(offspring_chromosome,x,jmlgen);

        %% populasi intermediate
```

```
            [main_pop,temp] = size(chromosome);
            [offspring_pop,temp] = size(offspring_chromosome);
            clear temp
            intermediate_chromosome(1:main_pop,:) = chromosome;
            intermediate_chromosome(main_pop + 1 : main_pop +
offspring_pop,:) = ...
                offspring_chromosome;

            %% non-dominated
            chromosome = non_domination_sort_mod(chromosome, M,
jmlgen);
            chromosome = replace_chromosome(intermediate_chromosome, M,
jmlgen, ukpop);
            if mod(i,100)== 0
                fprintf('%d generation completed\n',i);
            end

        end

        %% grafik
        save chromosome.mat chromosome
        plot(chromosome(:,jmlgen + 1),chromosome(:,jmlgen + 2), '*');
drawnow
```

Contoh grafik yang dihasilkan dari program tersebut dapat dilihat pada Gambar 6.



Gambar 6 Grafik NSGA2

BAB 4

APLIKASI ALGEMO UNTUK *CLUSTERING* WILAYAH

Algoritma Genetika Multi-Objektif pada *clustering* wilayah merupakan salah satu aplikasi yang terdapat dalam publikasi relatif baru yang ditulis oleh Arkeman Y, Wahanani NA, dan Kustiyo A (2012).

PENDAHULUAN

Reaktor Serba Guna G.A Siwabesi (RSG-GAS) yang dikelola Badan Tenaga Nuklir Nasional berada di kawasan Puspiptek, Serpong. Puspiptek merupakan kawasan perkantoran dan di sekelilingnya adalah perumahan serta fasilitas umum seperti sekolah dan tempat ibadah. Dalam pengelolaan fasilitas nuklir, setiap instalasi nuklir memerlukan data lingkungan yang diperbaharui setiap 5 tahun sekali. Informasi tersebut digunakan untuk perkiraan dampak radiologi terhadap komponen lingkungan dan masyarakat atas pengoperasian fasilitas nuklir. Bank Data Lingkungan Kawasan Nuklir Serpong antara lain berisi data rona lingkungan hidup, seperti radioaktivitas, cuaca, topografi, kualitas udara, air dan tanah, tata guna lahan, demografi, serta kesehatan masyarakat dalam wilayah radius 5 km dari tapak RSG-GAS. Data tersebut dapat digunakan baik untuk pelepasan zat radioaktif pada operasi normal maupun bila terjadi kecelakaan atau kedaruratan. Pengaruh terhadap kondisi lingkungan hidup harus diketahui secara dini yang merupakan umpan balik dalam pengelolaan kegiatan nuklir pada fasilitas nuklir Badan Tenaga Nuklir

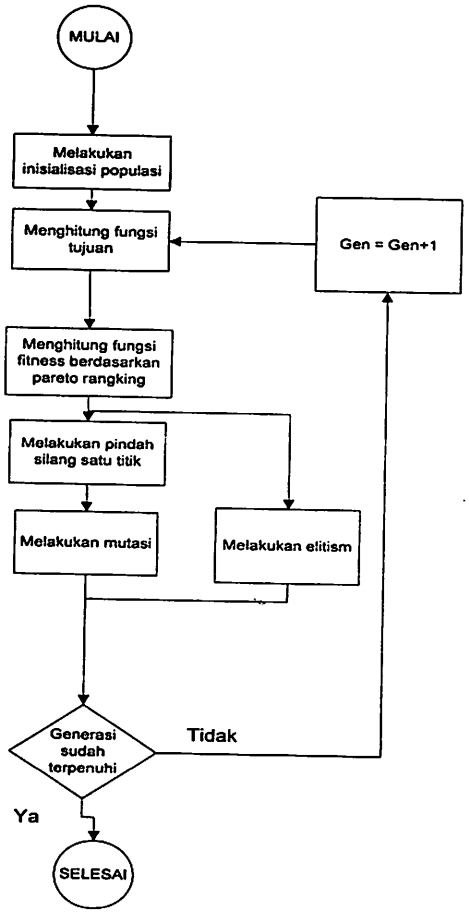
Nasional (Sumber: www.batan.go.id/datalingkungan). Pada penelitian ini dilakukan *clustering* wilayah dalam radius 5 km PPTN Serpong berdasarkan persentase 6 jenis penyakit yang diderita penduduk menggunakan algoritma genetika multi-objektif. Tujuan dilakukan pengelompokan ini adalah mengetahui pengaruh jauh-dekatnya lokasi kelurahan dari PPTN Serpong dengan tinggi-rendahnya persentase penderita penyakit pada suatu kelurahan.

Salah satu teknik *clustering partisioning* adalah K-Means, yang berusaha mempartisi data yang ada dalam bentuk dua atau lebih kelompok (Berkhin 2002). Pada proses iterasinya, K-Means tidak selalu menghasilkan nilai varian antar-*cluster* yang minimum dan varian antar-*cluster* yang maksimum. Kelemahan algoritma K-Means di antaranya sulit untuk mencapai optimum global, akan tetapi hanya bisa minimum lokal. Hal ini menyebabkan kualitas *cluster* yang dihasilkan kurang optimal. Kelemahan algoritma K-Means di antaranya sulit untuk mencapai optimum global, akan tetapi hanya bisa minimum lokal. Untuk meningkatkan kinerja *clustering* K-Means, salah satu metode yang bisa digunakan adalah algoritma genetika. Algoritma ini berbasis teori evolusi dan biologi, yang dapat melakukan pencaharian dalam ukuran besar dan nonlinier. Untuk mencapai kondisi konvergen digunakan operator genetik, yaitu persilangan, mutasi, dan elitism yang dievolusikan dalam fungsi *fitness*. Beberapa pendekatan yang bisa digunakan untuk evaluasi kuantitatif hasil *clustering* antara lain Hubert Statistic, Indeks Dun, dan Indeks Davies Bouldin. Pendekatan Indeks Davies Bouldin bertujuan untuk memaksimalkan jarak antar-*cluster* dan pada waktu yang sama mencoba untuk meminimalkan jarak antartitik dalam

sebuah *cluster* (Salazar *et al.* 2002). Metode yang digunakan adalah algoritma genetika multi-objektif dengan pendekatan *pareto rangking*. Hasil perbandingan metode *pareto rangking* dan NSGA II antara lain pareto unggul pada waktu komputasi (25 menit) dibandingkan dengan NSGA II (82 menit) dengan komputasi pararel menggunakan ukuran populasi 500, generasi 500. Himpunan penyelesaian yang dihasilkan *pareto rangking* lebih sedikit (153) dibanding menggunakan NSGA II (500).

METODE

Tahapan proses aplikasi Algoritma Genetik Multi-Objektif untuk *clustering* wilayah dapat dilihat pada Gambar 7.



Gambar 7 Tahapan proses ALGEMO

HASIL DAN PEMBAHASAN

Clustering wilayah dilakukan pada jumlah *cluster* sebanyak 2 sampai dengan 17. Dari keseluruhan hasil hanya *cluster* dengan jumlah 2, 3, dan 4 yang memiliki hasil *cluster* sesuai dengan jumlah yang diinginkan. Jumlah *cluster* di atas 4 memiliki hasil *cluster* yang kurang sesuai dengan yang diinginkan. Semakin banyak jumlah *cluster*, maka jarak antarrata-rata yang

dibangkitkan semakin kecil, sehingga objek memiliki peluang besar untuk berada dalam *cluster* yang sesuai. Pada Tabel 1 disajikan nilai varian dalam *cluster*, varian antar-*cluster*, serta indeks pada jumlah *cluster* 2, 3, dan 4. Dari hasil tersebut terlihat nilai indeks terkecil pada *cluster* sebanyak 2, 3, dan 4, yaitu 0.21, 0.23, dan 0.48. Perbedaan nilai varian dalam *cluster* pada setiap *cluster* relatif tidak terlalu jauh, sedangkan perbedaan nilai varian antar-*cluster* relatif jauh. Indeks naik seiring kenaikan jumlah *cluster*, kenaikan yang tajam pada *cluster* sebanyak 4.

Tabel 1 *Cluster* optimum

Cluster	Varian dalam *cluster*	Varian antar *cluster*	Indeks
2	0.02	0.05	0.21
3	0.01	0.11	0.23
4	0.01	0.09	0.48

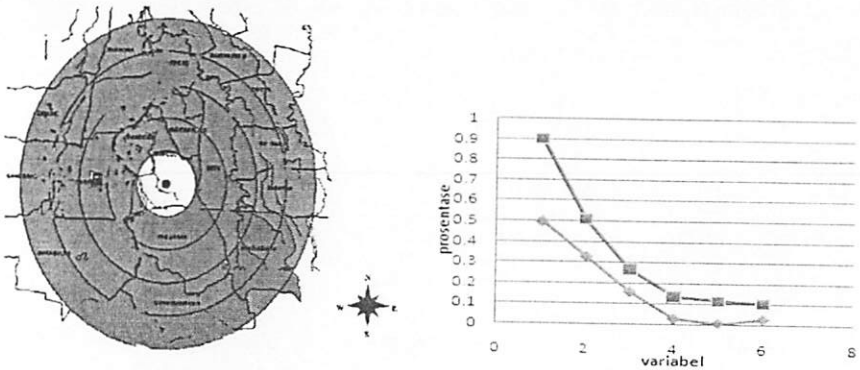
Hasil *clustering* wilayah sekitar PPTN berdasar persentase penduduk yang menderita 6 jenis penyakit dengan *cluster* sebanyak 2 adalah (Gambar 8 a):

Cluster 1 : Pengasinan (warna hijau)

Cluster 2 : Buaran, Ciater, Rawabuntu, Serpong, Dangdang, Suradita, Kranggan, Muncul, Setu, Babakan, Pademangan, Cibogo, Cisauk, Sampora, Gunung Sindur, Sukamulya, dan Pabuaran (warna merah).

Dari hasil *clustering* terlihat *cluster* 1 merupakan *cluster* yang anggotanya memiliki persentase yang rendah pada kombinasi 6 jenis penyakit tersebut, sedangkan *cluster* 2 adalah *cluster* yang anggotanya

relatif tinggi pada kombinasi 6 jenis penyakit tersebut. Sementara dari segi lokasi dari PPTN Serpong, Pengasinan relatif jauh dari PPTN. Namun terdapat lokasi yang sama jauhnya dari kedua tempat tersebut, seperti Gunung Sindur dan Sukamulya tidak menjadi anggota *cluster* 1 tersebut. Hal ini mengindikasikan tidak adanya relasi jauh-dekat lokasi terhadap penyebaran penyakit tersebut.



(a) (b)

Gambar 8 (a) *Clustering* wilayah sebanyak 2, (b) Plot titik pusat 6 variabel *cluster* 2

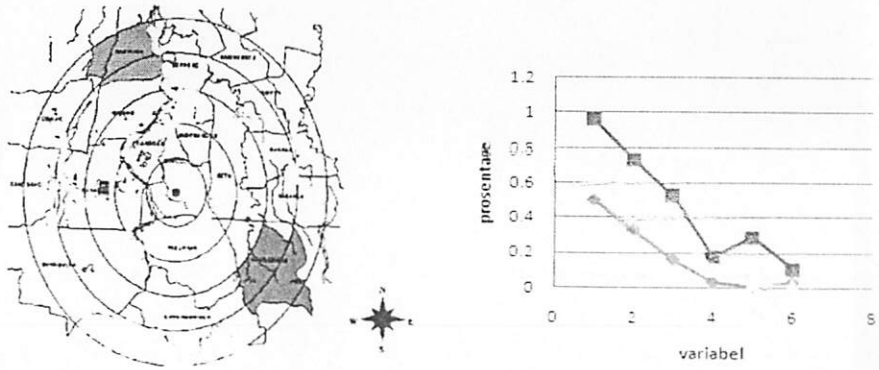
Pada *clustering* wilayah dengan *cluster* sebanyak 3 adalah (Gambar 9 a):

Cluster 1 : Pengasinan (warna hijau)

Cluster 2 : Sampora (warna merah)

Cluster 3 : Buaran, Ciater, Rawabuntu, Serpong, Dangdang, Suradita, Kranggan, Muncul, Setu, Babakan, Pademangan, Cibogo, Cisauk, Pengasinan, Gunung Sindur, Pabuaran, dan Sukamulya (warna kuning).

Dari hasil di atas terlihat *cluster* 1 memiliki persentase yang rendah untuk kombinasi 6 jenis penyakit tersebut, persentase sedang pada *cluster* 3 dan persentase tinggi pada *cluster* 2.



(a) (b)

Gambar 9 (a) Sebaran wilayah berdasar *cluster* 3, (b) Plot titik pusat 6 variabel *cluster* 3

Perbedaan persentase titik pusat variabel masing-masing *cluster*, untuk *cluster* sebanyak 3 seperti pada Gambar 8b. Persentase rendah ditunjukkan dengan warna hijau, persentase sedang warna kuning, dan persentase tinggi warna merah (Gambar 8b).

Berdasarkan hasil *cluster* sebanyak 2, 3, dan 4 terlihat bahwa keluarahan Pengasinan dan Sampora selama berada dalam *cluster* tersendiri karena Pengasinan mempunyai persentase yang rendah, sedangkan Sampora mempunyai persentase yang tinggi. Dari segi lokasi, Pengasinan berada jauh di sebelah tenggara, sedangkan Sampora berada di barat laut. Kedua kelurahan tersebut memiliki lokasi yang relatif jauh dari PPTN Serpong, tetapi keduanya memiliki tingkat persentase yang berbeda.

IMPLIKASI MANAJERIAL

Hasil *clustering* 18 kelurahan yang berada dalam radius 5 km PPTN Serpong berdasarkan 6 jenis penyakit yang diderita menunjukkan lokasi penderita menyebar, tidak bergantung jarak lokasi dari PPTN Serpong.

BAB 5
APLIKASI ALGEMO UNTUK *FLOWSHOP SCHEDULING*

Bab 5 membahas tentang algoritma genetika baru pada populasi yang heterogenus untuk menyelesaikan masalah *flowshop scheduling* multi-objektif. Aplikasi ini telah dilakukan oleh Arkeman Y dan Tamura H (2007) menggunakan 20 *job* dan 20 mesin dan menghasilkan solusi *pareto optimum* baru yang belum pernah ditemukan dari penelitian sebelumnya, yaitu oleh Varadharajan dan Rajendran (2005).

PENDAHULUAN

Pembentukan *shop scheduling* dengan pola alokasi tugas ke *resource* (mesin) merupakan suatu kendala kepuasan serta tujuan yang harus diselesaikan (Kusaik 1990; Pinedo 1995). *Shop scheduling* dapat diklasifikasi ke dalam masalah *flowshop, jobshop,* atau *openshop* (Bagchi 1999). Sebuah *flowshop* dikarakteristikan oleh pekerjaan dengan *flow* searah serta diproses secara berurutan dalam urutan yang sama, satu per satu. Tantangan yang dihadapi para proses *flowshop scheduling* ialah ketika menentukan urutan yang optimum, di mana setiap *job* harus diproses sehingga nilai *makespan* dan total waktu pekerjaan minimum.

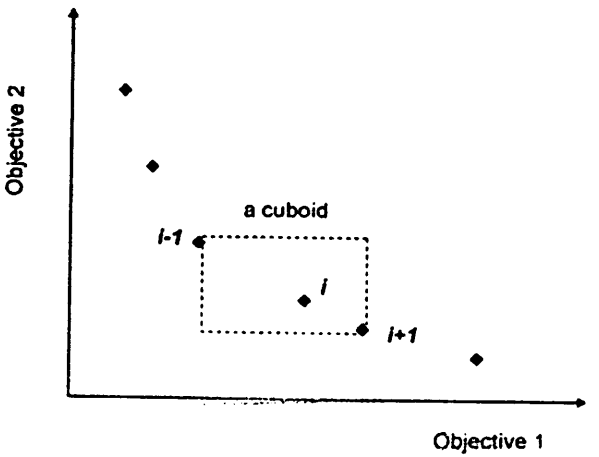
Biasanya penelitian sistem penjadwalan menggunakan permasalahan hanya satu (*single*) objektif. Akan tetapi, di dunia nyata permasalahan pada sistem penjadwalan memiliki lebih dari satu objektif (*multi-objective*). Sebagai contoh dua atau lebih optimasi objektif, yaitu nilai *makespan*, total *flowtime,* maksimum keterlambatan, dan total keterlambatan. *Makespan*

dan total *flowtime* terkait dengan memaksimalisasi sistem utilisasi dan meminimalisasi proses persediaan untuk masing-masing pekerjaan, sedangkan yang lainnya berhubungan dengan tanggal jatuh temponya pekerjaan. Dalam kasus ini menggunakan *makespan* dan total *flowtime* sebagai multi-objektif.

Metode yang digunakan untuk permasalahan *flowshop scheduling* dengan multi-objektif ialah algoritma genetika baru yang bernama *heterogeneous multi-objective genetic algorithm* (hMGA), di mana metode ini menggunakan kromosom yang identik atau duplikat dalam sebuah populasi. Implementasi untuk *testing* dan validasi menggunakan NSGA-II dan *multi-objective simulated annealing* (MOSA).

METODE PERMASALAHAN

Pengembangan algoritma genetika baru pada kasus ini digunakan untuk menyelesaikan masalah penjadwalan multi-objektif yang disebut *heterogeneous multi-objective genetic algorithm* (hMGA). hMGA pada umumnya memiliki proses yang sama dengan kebanyakan algoritma multi-objektif, dengan konsep *pareto optimality* untuk membandingkan solusi. Solusi tujuan (*fitness*) yang digunakan bergantung pada *non-dominated rank*. Pada kenyataannya, untuk membandingkan dua solusi *rank non-dominated* yang sama dengan benar menggunakan ide Deb (2002) yaitu menghitung jarak kerapatan *(crowding distance*) antarsolusi. *Crowding Distance* (CD) merupakan batas sebuah kubus yang merupakan solusi terdekat dengan *Pareto front* yang sama seperti pada Gambar 10.



Gambar 10 *Crowding Distance*

Fungsi $f_m^{(i-1)}$ dan $f_m^{(i+1)}$ akan membagi nilai ke dalam fungsi objektif *m* yang merupakan solusi terdekat di antara solusi *i* dengan $f_m^{(i-1)} \leq f_m^{(i)} \leq f_m^{(i+1)}$, di mana f_m^{max} merupakan nilai maksimum dan f_m^{min} merupakan nilai minimum untuk fungsi objektif *m,* dan akhirnya *M* adalah total dari jumlah fungsi objektif. Solusi *crowding distance* ke *i* dapat dihitung dengan formula:

$$CD(i) = \sum_{m=1}^{M} \frac{f_m^{(i+1)} - f_m^{(i-1)}}{f_m^{max} - f_m^{min}}$$

CD (*i*) dapat juga diisikan dengan ∞, jika solusi ke *i* adalah batas solusi dalam semua fungsi objektif yang terpilih. Sebuah solusi ke *i* dinyatakan mendominasi sebuah solusi ke *j* jika hanya:

$$rank(i) < rank(j) \textbf{ or } rank(i) = rank(j) \textbf{ and } CD(i) > CD(j)$$

Deb (2002) menjelaskan bahwa NSGA-II merupakan metode yang baik diterapkan untuk mengoptimalkan beberapa fungsi yang sulit. Akan

tetapi, algoritma ini memiliki kelemahan, yaitu memungkinkan dalam populasi memiliki kromosom yang identik (*duplicate*). Kromosom yang identik (di dalam fungsi objektif) akan diberi nilai CD tertentu sesuai rumus fungsi yang digunakan. Berdasarkan penelitian menunjukkan bahwa untuk fungsi kontinu (yang memiliki jumlah tak terbatas atau sangat besar Pareto solusi optimumnya) NSGA-II akan menemukan satu set baik solusi Pareto-optimal. Namun, untuk masalah-masalah yang hanya memiliki sejumlah solusi Pareto-optimal (kemungkinan kurang dari ukuran populasi yang digunakan dalam algoritma), seperti dalam penjadwalan *flowshop* permutasi, NSGA-II bisa terjebak untuk konvergensi prematur dan menghasilkan lokal Pareto-optimal solusi. Oleh karena itu untuk mengatasi permasalahan yang ada, penelitian ini menggunakan *multi-objective* GA yang baru untuk permasalahan *flowshop schedulling* yang memiliki populasi *heterogeneous*. Di dalam algoritma baru dibangun sebuah *heterogeneous populations procedure* (HPP) yang bertujuan untuk memeriksa struktur dari kromosom sebelum membentuk populasi yang baru. Kromosom identik yang ditolak untuk masuk di populasi dan akan diganti dengan kromosom baru hasil genetik manipulasi.

Pseudo code hMGA

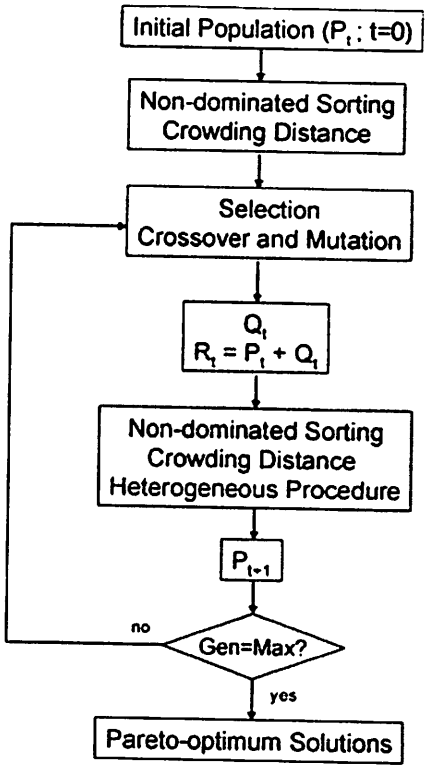
```
Input:
N (population size)•
Pc, Pm (probability of crossover and mutation)•
MaxGen (maximum number of generation as stopping criteria)•

Output:
NewPop3[1..N] (heterogeneous Pareto optimum solutions)

•begin
    (1) Initialization:
Generate an initial population OldPop[1..N] of string chromosome.
```



```
For each solution in OldPop compute the value of MSpan (makespan) and
      TFT(Total Flow Time);
    (2) Classification of initial population:
Classify OldPop using Non-dominated Sorting procedure (Deb 2002) and
      then obtain the list of Front1, Front2, . . . , Frontk.
Compute crowding distance value for each solution in all fronts using
      crowding sort procedure (Deb, 2002);
    (3) Set gen:= 0;
    (4) Selection, Crossover and Mutation:
gen:=gen+1;
repeat
    4.a Use binary tournament selection (Goldberg, 1989) and crowded
        comparison operator (Deb, 2002) to select two solutions from
        OldPop and store them into mate1and mate2
    4.b Crossover mate1and mate2by using Partially Matched Crossover
        (PMX) operator (Goldberg and Lingle, 1985) with probability
        Pcand obtain two new solutions child1 and child2
    4.c Mutate child1and child2using inversion method (Gen and Cheng,
        1997) with probability Pmto obtain child1' and child2'
    4.d Store child1' and child2' in NewPopuntil N/2 times;
    (5) Combination of population:
Combine New-Pop[1..N] with OldPop[1..N] to form UniPop[1..2N]
    (6) Classification of combined population:
Classify UniPop using Non-dominated sorting procedure and then obtain
      the list of Front1, Front2, . . . , Frontk.
Compute crowding distance value for each solution in all fronts using
      crowding sort procedure;
    (7) Population sortation:
Sort UniPop from the best to the worst solutions using crowded
      comparison operator. This sorted population is called New-
      Pop2[1..2N].
    (8) Heterogeneous procedure:
Discard all the duplicates of chromosomes in NewPop2 to form a fully
      heterogeneous population NewPop2' with size 2N-d, where d is the
      number of discarded solutions.
```

```
    (9) Pareto-optimum solutions formation:
Select the best N solutions from NewPop2'and then obtain a set of
      fully heterogeneous Pareto optimum solutions NewPop3
If gen = MaxGen
then Stop
else
set OldPop: =NewPop3 and go to Step 4
end;
```

Diagram alir proses hMGA dapat dilihat pada Gambar 11.



Gambar 11 Diagram alir hMGA

STRUKTUR KROMOSOM

Sebuah solusi populasi pada penelitian ini diwakili dalam sebuah kromosom *string*. Sebagai contoh:

9 *job*:

A	B	C	D	E	F	G	H	I

Sebuah kromosom dapat berupa:

B	F	G	A	C	I	E	H	D

Yang menjelaskan bahwa ***job*** dimulai dari ***B, F, G,*** dan seterusnya sampai *job D* selesai.

3 mesin:

M1	M2	M3

Keuntungan dari representasi kromosom *string* yang digunakan dalam penelitian ini adalah tidak memerlukan pembuatan jadwal untuk menerjemahkan urutan pengolahan menjadi jadwal yang sesuai.

FUNGSI *FITNESS*

Fungsi *fitness* atau tujuan dalam penelitian ini adalah mengetahui urutan pekerjaan meminimalkan waktu pekerjaan maksimum dan biasa disebut makespan adalah meminimalkan waktu pengerjaan secara keseluruhan.

Rumus umum untuk menghitung total waktu urutan pekerjaan dengan *n jobs* dan *m* mesin adalah:

$$C_{i0} = 0, C_{j0} = 0$$

$$C_{ij} = \max\{C_{i,j-1}, C_{i-1,j}\} + t_{ij}\text{, dengan } (i = 1, \dots, n) \; dan \; (j = 1, 2, \dots m)$$

Di mana C_{ij} adalah waktu penyelesaian *job* ke *i* di dalam mesin ke *j*. Dengan total waktu pekerjaan *F* diperoleh dengan menjumlah semua C_{im}.

$$F = \sum_{i=1}^{n} C_m$$

dan *makespan* maksimum (C_{max}) didapatkan dengan menemukan nilai maksimum dari semua C_{im}.

$$C_{max} = \max\{C_{im}\}\text{, dengan } (i = 1, \dots, n)$$

SELEKSI

Proses seleksi yang dilakukan adalah *Binary Tournament Selection* (BTS) dengan membandingkan operator kerapatan. Perbandingan operator kerapatan ($<_c$) membandingkan dua solusi dan mengembalikan pemenang turnamen, diasumsikan setiap solusi *i* memiliki dua atribut, contoh *non-domination rank* dan lokal *crowding distance* di dalam populasi. Sebuah solusi *i* memenangkan dari solusi *j* jika solusi *i* memiliki *rank* yang lebih baik atau *fitness* dan jika mereka memiliki *rank* yang sama, tetapi solusi *i* memiliki *crowding distance* yang lebih baik dari solusi *j*.

PINDAH SILANG DAN MUTASI

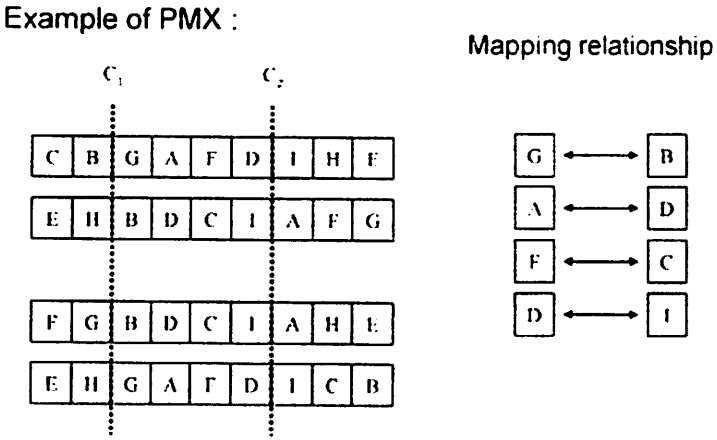
Pindah silang dan mutasi yang digunakan masing-masing adalah *Partially Matched Crossover* (PMX) (Goldberg 1989) dan *Inverse Mutation* (Gen dan Cheng 1997). PMX dapat dilihat sebagai perpanjangan dari dua titik pindah silang (*two point crossover*) untuk representasi permutasi *string* biner. PMX memiliki spesial perbaikan prosedur untuk menyelesaikan yang

tidak dapat dilakukan ketika menggunakan *two point crossover* yang sederhana.

Prosedur PMX ialah:

1. Pilih dua posisi sepanjang *string* secara rata dan acak. *Substring* pada dua titik posisi ini dikatakan sesi pemetaan (*mapping*).
2. Pertukaran dua *substring* antar-orang tua akan menghasilkan keturunan atau anak.
3. Menentukan hubungan pemetaan (*mapping*) antara dua sesi pemetaan.
4. Melegalkan keturunan dengan hubungan pemetaan.

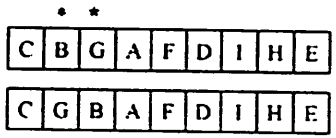
Ilustrasi PMX dapat dilihat pada Gambar 12.



Gambar 12 Ilustrasi PMX

Inverse mutation secara acak menukarkan *allele* dalam kromosom untuk menghasilkan keturunan atau anak baru yang disebut mutan. Ilustrasi *inverse mutation* dapat dilihat pada Gambar 13.

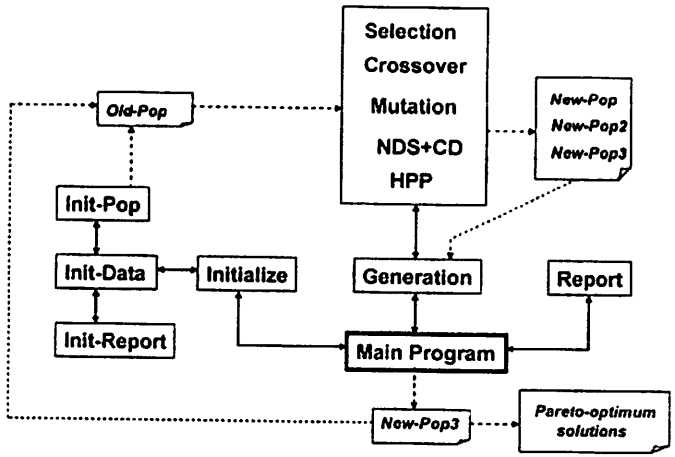
The mutation technique used is "inverse methods":



Gambar 13 Ilustrasi *inverse mutation*

IMPLEMENTASI DAN PENGUJIAN

Dalam sistem ini NSGA-II dan hMGA dibangun menggunakan Delphi 7, di mana struktur program hMGA dapat dilihat pada Gambar 14.

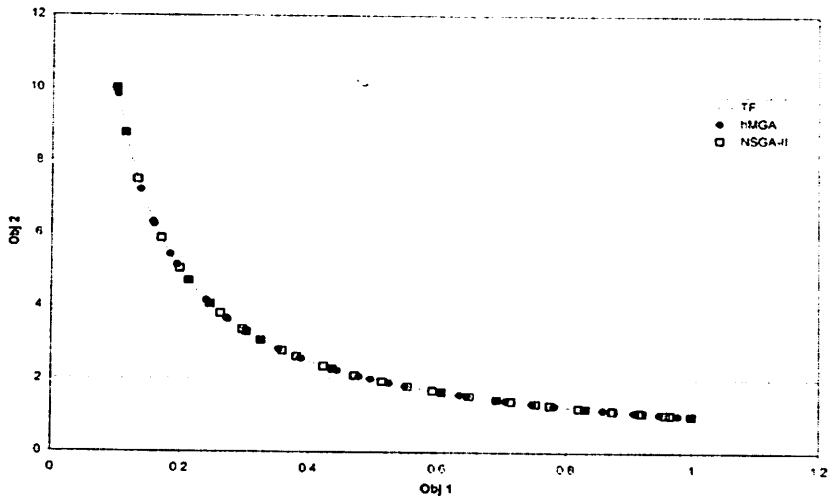


Gambar 14 Struktur program hMGA

Fungsi kontinu sederhana yang digunakan untuk melakukan pengujian pertama dalam sistem sebagai berikut.

$$\begin{aligned} \textit{Minimize} &: f_1(x) = x_1 \\ \textit{Minimize} &: f_2(x) = \frac{(1+x_2)}{x_1} \\ \textit{Subject to} &: 0.1 < x_1 < 1, \\ &\quad 0 < x_2 < 5 \end{aligned}$$

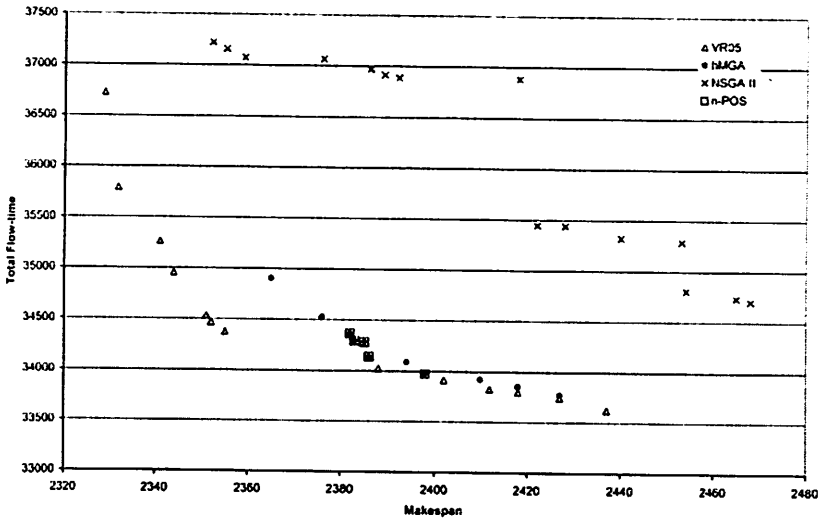
NSGA-II dan hMGA menggunakan ukuran populasi = 40, peluang pindah silang (*Pc*) = 0.9, peluang mutasi (*Pm*) = 0.01, dan maksimum generasi = 500. Variabel 1 dan variabel 2 masing-masing direpresentasikan dengan 24 bit dan 26 bit. Hasil solusi *Pareto-optimum* dari masing-masing NSGA-II dan hMGA dapat dilihat pada Gambar 15.



Gambar 15 Solusi *Pareto-optimum* dari NSGA-II dan hMGA pada fungsi kontinu sederhana (Deb 2001; Arkeman dan Tamura 2007)

HASIL NUMERIK

NSGA-II dan hMGA dibandingkan dengan algoritma Varadharajan dan Rajendran (2005) (VR05) dan metode sebelumnya, yaitu ENGA (Bagchi 1999), MOGLS (Ishibuchi dan Murata 1998), dan sebuah *posterior heuristic* (Framinen *et al.* 2002). Plot *makespan* dan total waktu pekerjaan yang didapatkan dari 2.000 generasi dari NSGA-II dan hMGA memiliki hasil solusi yang sama baiknya dengan hasil dari VR05 dan nPOS (Gambar 16).



Gambar 16 Solusi *Pareto-optimum* NSGA-II dan hMGA dibandingkan hasil dari VR05.

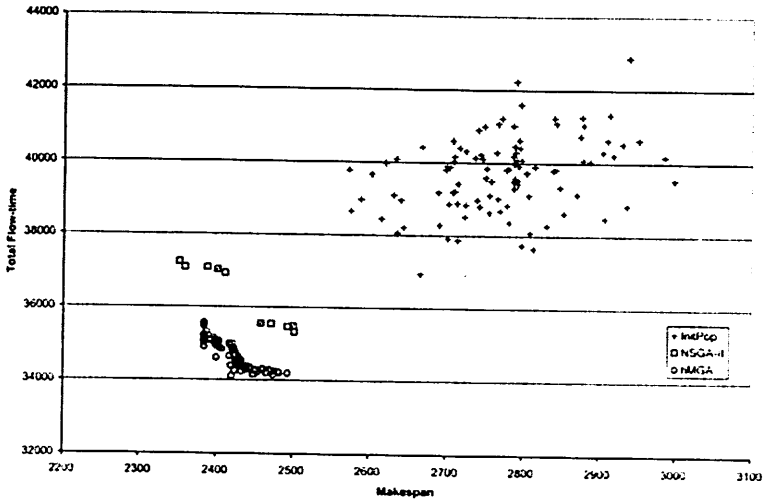
Gambar 16 menjelaskan bahwa hMGA memiliki hasil solusi Pareto-optimum yang lebih baik dari NSGA-II, selain itu hMGA menghasilkan hasil solusi yang identik dengan hasil solusi n-POS (Tabel 2). Performa hMGA terbukti sangat menjanjikan menghasilkan solusi *Pareto-optimal* terbaik.

Tabel 2 Solusi *Pareto-optimal* baru hasil hMGA

Makespan	Total flowtime
2382	34 366
2383	34 295
2385	34 281
2386	34 140
2398	33 976

Perbedaan antara hMGA dan NSGA-II dapat ditelusuri dengan melihat secara rinci pada solusi penengah mereka. Pada Gambar 16 dapat dilihat bahwa pada generasi ke-100, NSGA-II hanya menghasilkan sepuluh solusi (dengan 90 duplikat), sedangkan hMGA menghasilkan 100 solusi yang berbeda. Fitur heterogen membuat hMGA mengungguli NSGA-II dalam mencari solusi *Pareto-optimal* dalam hal kedekatan dengan benar *Pareto-front* serta keragaman solusi.

Perlu dicatat di sini bahwa algoritma yang diusulkan dengan beberapa modifikasi dapat menangani lebih dari dua fungsi objektif. Namun dalam banyak skenario kehidupan nyata, fungsi objektif mungkin memiliki efek interaksi dan menganalisis dua tujuan dalam satu waktu dari serangkaian tujuan yang lebih besar untuk membantu menyederhanakan evaluasi masalah. Selain itu, semua simulasi yang dilakukan telah dilaksanakan di ruang objektif-fungsi. Bahkan untuk masalah ini, penjadwalan tertentu tidak perlu untuk mencari dalam keputusan variabel ruang, seperti ruang variabel keputusan bersifat kualitatif (yaitu kombinasi jadwal), bukan kuantitatif. Namun untuk masalah lain, algoritma ini juga dapat diterapkan dalam ruang variabel keputusan, jika diinginkan.



Gambar 17 Perbandingan hasil populasi generasi ke 100 hMGA dan NSGA-II

BAB 6
APLIKASI *MULTI-OBJECTIVE GENETIC ALGORITHM* UNTUK IDENTIFIKASI DAUN TUMBUHAN OBAT

Bab 6 membahas tentang *multi-objective genetic algorithm* pada populasi berupa parameter yang digunakan untuk ekstraksi citra. Aplikasi ini telah dilakukan oleh Gibtha Fitri Laxmi, Dr. Yeni Herdiyeni, dan Dr. Yandra Arkeman serta menghasilkan solusi pemilihan parameter yang sesuai untuk mengekstraksi ciri citra daun.

PENDAHULUAN

Indonesia sebagai negara tropis memiliki keanekaragaman hayati yang sangat tinggi. Hal tersebut dapat dilihat dari beragamnya jenis flora di Indonesia dan masuknya Indonesia ke dalam sepuluh negara yang kekayaan keanekaragaman hayatinya tertinggi atau dikenal dengan *megadiversity country*. Keanekaragaman hayati di Indonesia memiliki lebih dari 38.000 spesies tanaman (Bappenas 2003), akan tetapi banyak dari tanaman itu sendiri tidak diketahui jenisnya, sehingga perlu dilakukan identifikasi pada tanaman tersebut. Proses identifikasi tanaman dapat dilakukan dengan berbagai cara, di antaranya dengan identifikasi manual menggunakan organ generatif buah dan bunga yang akan diperoleh hasil dalam waktu cukup lama. Teknologi identifikasi secara otomatis diperlukan untuk mempercepat proses identifikasi tersebut menggunakan organ vegetatif seperti daun yang paling mudah untuk ditemukan.

Kulsum (2010) telah membuat teknologi indentifikasi otomatis dengan melakukan ekstraksi ciri tanaman hias menggunakan *Local Binary Pattern* (LBP) *descriptor* dan proses klasifikasi menggunakan *Probabilistic Neural Network* (PNN). Herdiyeni dan Kusmana (2013) telah melakukan penelitian dengan mencoba beragam operator *Local Binary Pattern* (LBP) hingga menggabungkan operator tersebut.

Metode LBP memiliki kelemahan dalam *thresholding* pada nilai keabuan piksel yang membuat penyajian teksturnya sensitif terhadap *noise*. Hasil *threshold* pada *original* LBP terkadang menghasilkan pengodean pola biner yang tidak sesuai dengan kandungan nilai pikselnya. Hal ini disebabkan oleh ketidakpastian yang diakibatkan oleh adanya keragaman tekstur pada daun. Iakovidis *et al.* (2008) menggunakan *fuzzy logic* untuk mengatasi ketidakpastian pada representasi tekstur LBP, yang dikenal sebagai metode *Fuzzy Local Binary Pattern* (FLBP). Herdiyeni dan Wahyuni (2012) melakukan penelitian identifikasi daun tumbuhan obat menggunakan *Fuzzy Local Binary Pattern* (FLBP). FLBP menggunakan logika *fuzzy* untuk mengatasi masalah ketidakpastian pada representasi tekstur LBP (Iakovidis *et al.* 2008). Menurut Herdiyeni dan Wahyuni (2012), penentuan nilai operator FLBP dan *threshold* FLBP sangat berpengaruh terhadap akurasi pengenalan tumbuhan obat dan waktu komputasi. Penelitian tersebut menggunakan nilai operator dan *threshold* yang sama untuk semua data daun tumbuhan obat. Hal ini menyebabkan terjadinya kesalahan identifikasi pada beberapa spesies tumbuhan obat. Oleh karena itu diperlukan metode yang dinamis dan optimal untuk menentukan nilai operator FLBP dan *threshold* FLBP. Inilah yang memotivasi penggunakan

Multi-Objective Genetic Algorithm (MOGA) dalam penentuan nilai operator dan *threshold* dalam identifikasi daun tumbuhan obat.

Objek penelitian yang digunakan pada penelitian ini ialah citra daun. Ciri dari citra daun tersebut akan dilakukan proses klasifikasi. Proses klasifikasi yang akan digunakan ialah *Probabilistic Neural Network* (PNN) yang memiliki struktur sederhana dan proses *training* yang cepat tanpa harus memperbarui nilai bobot (Wu *et al.* 2007).

Tujuan penelitian ini adalah mengoptimasi pemilihan parameter dan operator pada FLBP menggunakan *Multi-Objective Genetic Algorithm* untuk meningkatkan hasil akurasi identifikasi citra daun. Data citra yang digunakan adalah berberapa tumbuhan di Indonesia serta operator FLBP yang digunakan ialah (8,1) dan (8,2).

METODE PENELITIAN

Penelitian ini menggunakan kombinasi dari operator LBP, nilai *threshold* FLBP, dan nilai klasifikasi PNN. Kombinasi operator LBP dan nilai *threshold* FLBP akan diproses menggunakan MOGA sehingga menghasilkan kumpulan kombinasi. Hasil kumpulan kombinasi tersebut memiliki operator LBP minimum dan nilai *threshold* FLBP minimum serta nilai peluang PNN yang maksimum.

DATA CITRA TUMBUHAN DAN PRAPROSES

Data yang digunakan dalam penelitian ini adalah citra digital daun tumbuhan obat sebanyak tiga puluh spesies. Total citra daun yang digunakan adalah 1.440 yang terdiri atas 30 jenis daun, depan dan belakang (masing-masing kelas 48 citra) diambil beberapa pada waktu yang berbeda

(pagi, siang, dan sore). Citra daun berformat JPG dan berukuran 270 × 240 piksel.

Mode warna citra keseluruhan dan citra daun kemudian diubah menjadi *grayscale* untuk proses ekstraksi selanjutnya. Hasil praproses data bertujuan mengurangi waktu pemrosesan data (*running time*).

EKSTRAKSI TEKSTUR DENGAN *FUZZY LOCAL BINARY PATTERN*

Ekstraksi tekstur pada citra daun hanya dilakukan pada piksel yang menyusun citra tersebut. Citra dikonversi ke *mode* warna *grayscale*. Selanjutnya membagi citra ke dalam beberapa *local region* sesuai dengan *sampling points* dan *radius* yang digunakan.

Tabel 3 Operator FLBP

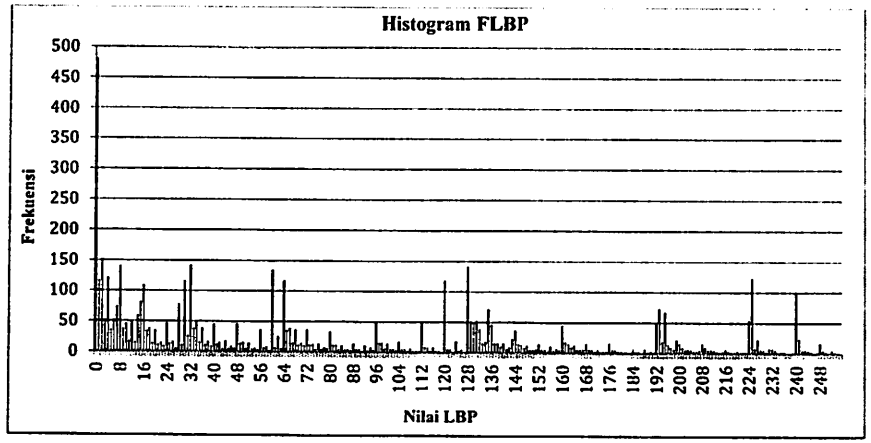
Operator (P,R)	Ukuran Blok (piksel)	Kuantisasi Sudut
(8,1)	3 x 3	45 derajat
(8,2)	5 x 5	45 derajat

Ekstraksi tekstur dilakukan dengan konvolusi menggunakan operator yang disajikan pada Tabel 3. Hasil perhitungan FLBP pun direpresentasikan dalam bentuk histogram.

Ciri citra yang dihasilkan kemudian diekstrak menggunakan nilai *threshold* FLBP fuzzifikasi (*F*) yang berbeda. Nilai *threshold* FLBP fuzzifikasi yang digunakan mulai dari $F = 1$ sampai $F = 10$. Di mana nilai *threshold* FLBP fuzzifikasi menentukan nilai biner yang dihasilkan. Jika terdapat sejumlah m

nilai Δp_t yang berada dalam rentang *fuzzy*, akan menghasilkan nilai biner sebanyak 2^m. Semakin besar nilai *threshold* FLBP fuzzifikasi, maka akan memberikan nilai rentang yang semakin besar yang berbanding lurus dengan nilai biner dan waktu komputasi dalam pembacaan nilai piksel.

Hasil ekstraksi FLBP menghasilkan frekuensi histogram yang ditunjukan Gambar 18, di mana histogram merupakan pertambahan C_{LBP} dari nilai LBP yang dihasilkan. Panjang *bin* yang dihasilkan pada histogram $FLBP_{P,R}$ bergantung pada jumlah *sampling points* (*P*) yang digunakan, yaitu 2^P. Pada penelitian ini, jumlah *P* yang digunakan adalah 8 sehingga jumlah *bin* pada histogram $FLBP_{P,R}$ sebanyak $2^8 = 256$ *bin*.



Gambar 18 Histogram FLBP pada citra

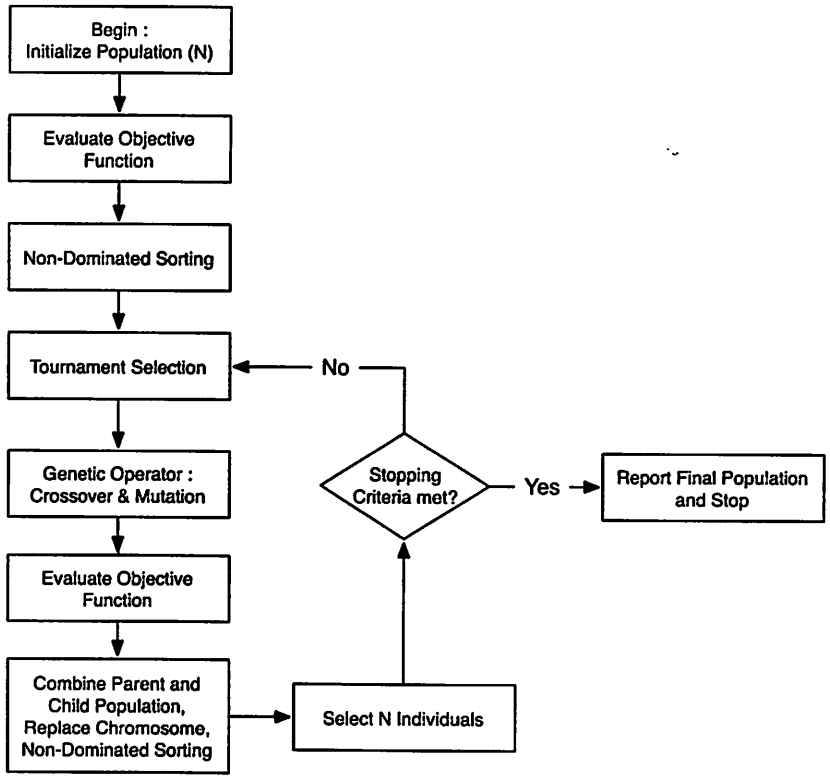
KLASIFIKASI DENGAN *PROBABILISTIC NEURAL NETWORK* (PNN)

Klasifikasi dilakukan pada vektor histogram untuk menentukan akurasi identifikasi tumbuhan. Klasifikasi dilakukan dengan membagi data

latih dan data uji dengan komposisi 80% dan 20% untuk data daun. Selanjutnya diperoleh model klasifikasi dari hasil pelatihan data.

MULTI-OBJECTIVE GENETIC ALGORITHM (MOGA)

Algoritma Genetika merupakan dasar dari pembentukan Algoritma Genetika Multi-Objektif, tahapan dalam proses genetika yang dilakukan sama dengan fungsi tujuan yang lebih dari satu atau disebut *multi-objective.* Algoritme Genetika Multi-Objektif pada penelitian ini menggunakan NSGA-II dengan skema prosesnya dapat dilihat pada Gambar 19.



Gambar 19 Proses NSGA-II

Inisialisasi Populasi merupakan tahap awal yang dilakukan untuk menentukan kombinasi variabel keputusan. Kombinasi populasi yang akan dibentuk berupa nilai *binary string* 0 dan 1. Banyaknya populasi yang digunakan sebanyak 20 individu yang dapat diubah sesuai keinginan pengguna.

Setiap individu memiliki 15 *bit*, dengan 10 *bit* pertama adalah variabel x_1 (parameter) dan 5 *bit* berikutnya ialah variabel x_2 (Operator LBP), dan memiliki ketentuan dibawah ini:

$$x_1,\ 1 \leq x_1 \leq 10$$

$$x_2,\ 1 \leq x_2 \leq 2$$

Sebuah kromosom yang terdiri atas bilangan biner akan dilakukan dekode nilai biner menjadi nilai desimal berdasarkan ketentuan di atas.

Kromosom-kromosom akan mengalami evolusi melalui sejumlah iterasi yang disebut generasi dan kromosom-kromosom tersebut akan dievaluasi menggunakan suatu fungsi yang disebut fungsi evaluasi. Fungsi evaluasi merupakan dimensi tujuan dari suatu keputusan, di mana terdapat 3 fungsi tujuan yang digunakan dalam penelitian ini, dengan meminimalkan fungsi pertama dan kedua, serta memaksimalkan fungsi yang ketiga. Fungsi pertama, kedua, dan ketiga masing-masing digunakan untuk menentukan nilai *threshold* FLBP, operator LBP, dan nilai peluang klasifikasi menggunakan fungsi PNN.

Fungsi evaluasi yang digunakan pada penelitian ini ditunjukkan di bawah ini:

$$Minimize : f_1(x) = x_1$$

$$Minimize : f_2(x) = x_2$$

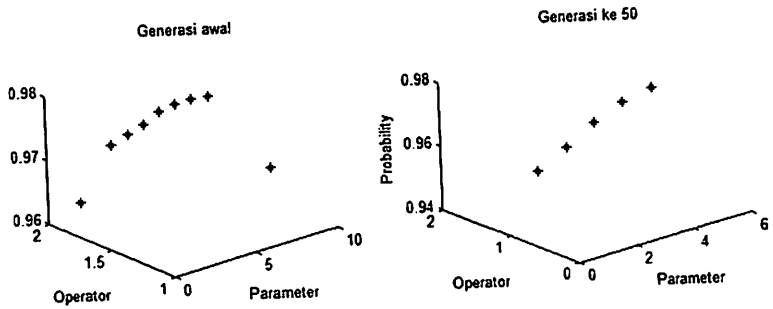
$$Maximize : f_3(x) = max\left(\frac{1}{n_i}\sum_{k=1}^{n_i} e^{\frac{\|x-x_{ik}\|^2}{\sigma^2}}\right), x = kueri\ fitur, x_k$$

$$= data\ latih$$

Non-Dominated Sorting Genetic Algorithm (NSGA-II) yang digunakan dalam penelitian ini menggunakan konsep tidak terdominasi satu sama lain dalam grup dan pe-*ranking*-an terdapat dalam tiap solusi kemudian dijadikan sebagai *front*.

Di dalam *front* setiap solusi akan memiliki sebuah pengukuran jarak yang relatif disebut *crowding distance*, di mana jarak rata-rata (di dalam ruang tujuan) dari dirinya sendiri ke *adjacent* solusi yang lain.

Non-dominated dan *crowding distance* akan menghasilkan konsep kumpulan beberapa titik yang merupakan solusi dari permasalahan. Ilustrasi peranan *non-dominated* dan *crowding distance* pada penelitian pada saat generasi pertama dan generasi terakhir terdapat pada Gambar 20.



Gambar 20 Ilustrasi *non-dominated* dan *crowding distance*

Hasil evaluasi lainnya menentukan kualitas individu digunakan untuk proses seleksi induk. Proses seleksi pada penelitian ini adalah *tournament*

selection. *Tournament selection* dapat menyesuaikan dan dapat beradaptasi dari perbedaan domain. Prosesnya dengan cara melakukan turnamen antar-*s* pesaing, dengan *s* merupakan ukuran turnamen, pemenang dari turnamen akan dimasukan ke dalam *mating pool* untuk proses genetik. Semakin banyak individu yang memiliki nilai *fitness* lebih tinggi, maka akan semakin banyak individu yang akan dipilh.

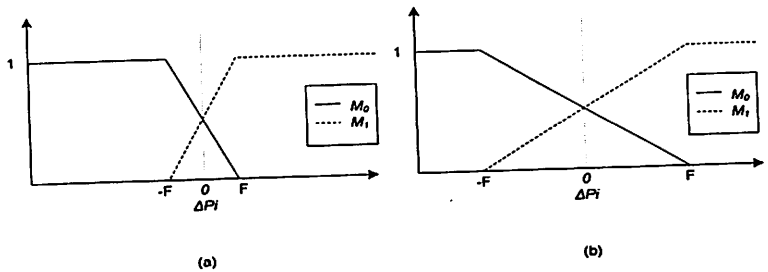
Proses pindah silang merupakan proses lanjutan dari proses seleksi yang menghasilkan calon-calon kromosom yang akan dipindahsilangkan. Proses pindah silang dilakukan menggunakan *one point crossover*. Di mana dalam prosesnya memerlukan dua kromosom induk dari hasil seleksi. Proses ini dilakukan dengan menggantikan semua nilai *allele* dari kromosom induk. Jumlah kromosom dalam satu populasi yang dipindahsilangkan bergantung pada nilai acak yang diambil kurang dari peluang pindah silang *Pc*. Peluang pindah silang *Pc* yang tinggi akan memungkinkan pencapaian alternatif solusi yang lebih bervariasi dan mengurangi kemungkinan menghasilkan nilai optimum yang tidak dikehendaki.

Proses mutasi penelitian ini menggunakan fungsi *binary bit flip*. Proses mutasi merupakan proses pengubahan nilai gen pada kromosom yang telah dipilih sebelumnya. Pemilihan gen tersebut ialah dengan cara mengambil nilai acak [0,1] yang kemudian nilai acak akan dibandingkan dengan peluang mutasi. Setelah itu *allele* dari setiap individu yang terpilih akan dimutasi menggunakan fungsi *binary bit flip* mutasi, mengganti nilai dari 0 menjadi 1 dan sebaliknya.

PENGUJIAN DATA

Data citra uji yang telah ditentukan akan dilakukan optimasi menggunakan NSGA-II untuk mendapatkan nilai *threshold* FLBP dan operator LBP. NSGA-II menghasilkan nilai *threshold* FLBP dan operator LBP yang berbeda-beda walaupun dalam kelas citra yang sama.

Nilai *threshold* FLBP yang digunakan ialah dari 1 sampai dengan 10. Nilai *threshold* FLBP merupakan nilai Δp_i (selisih antara piksel pusat dan piksel tetangga) di mana jika Δp_i besar, nilai *threshold* FLBP menggunakan nilai yang besar dan sebaliknya (Gambar 21).

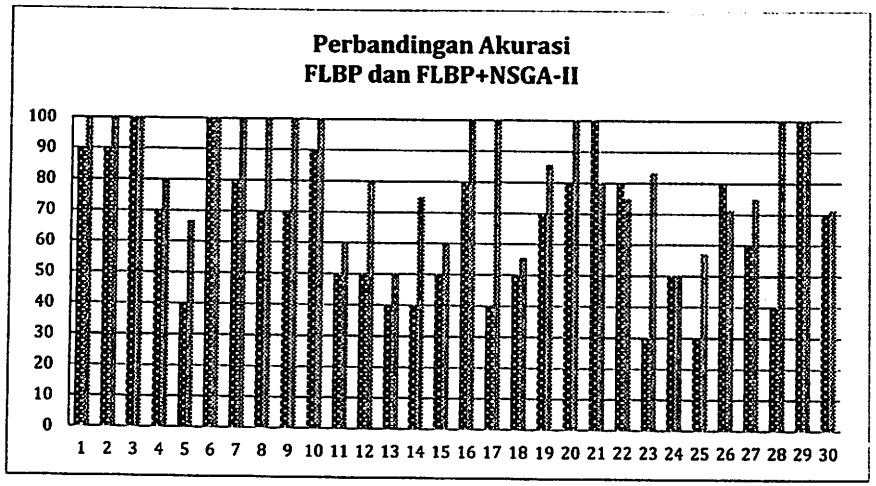


Gambar 21 Nilai *threshold* FLBP dan Δp_i, (a) nilai *threshold* FLBP kecil, nilai selisih kecil (b) nilai *threshold* FLBP besar, nilai selisih besar

Pemilihan nilai *threshold* FLBP dan operator LBP menggunakan NSGA-II pada identifikasi citra daun mampu menghasilkan nilai akurasi yang lebih baik, terlihat banyaknya kelas yang teridentifikasi 100%. Persentase akurasi yang dimiliki untuk pengujian data sebesar 66.3% untuk penggunaan Operator LBP (8,2) dan nilai *threshold* FLBP F = 4, sedangkan menggunakan pemilihan Operator LBP dan nilai *threshold* FLBP yang dilakukan metode NSGA-II memiliki persentase akurasi sebesar 82,54%. Hal ini membuktikan bahwa berbedanya nilai *threshold* FLBP dan operator LBP tiap citra

memengaruhi persentase akurasi yang dimiliki dengan adanya peningkatan sebesar 16%.

Perbandingan persentase akurasi Operator LBP dan nilai *threshold* FLBP tetap dengan pemilihan operator LBP dan *threshold* FLBP menggunakan NSGA-II dapat dilihat pada Gambar 22.



Gambar 22 Perbandingan akurasi FLBP dan FLBP + NSGA-II

Setiap kelas citra memiliki nilai *threshold* FLBP dan jumlah nilai *threshold* FLBP yang mampu mengidentifikasi dengan benar. Terlihat pada Tabel 4 bahwa citra pertama dan kedua merupakan kelas yang sama menggunakan 2 nilai *threshold* FLBP, yaitu F = 7 dan 10, terlihat bahwa rentang nilai *threshold* FLBP yang digunakan besar. Hal itu disebabkan nilai kontras citra yang dimiliki tinggi, maka akan lebih baik jika memiliki nilai *threshold* FLBP yang besar agar nilai tekstur yang direpresentasikan berada dalam rentang *threshold* FLBP. Contoh lain yang kontras citra uji yang berbanding lurus dengan nilai *threshold* FLBP ditunjukkan pada Tabel 4.

Tabel 4 Contoh hasil variabel objektif nilai *threshold* FLBP NSGA-II

Citra				
Nilai *Threshold* FLBP	7	10	2	10

Pemilihan Operator LBP yang tepat dapat memengaruhi nilai akurasi terlihat pada kelas 23 (Remak Daging) Gambar 23 yang awalnya memiliki akurasi sebesar 30% menggunakan operator LBP (8,2), mengalami peningkatan dengan operator LBP sebesar 83.33%. Perbedaan pemilihan operator dapat terlihat pada Tabel 5 bahwa jika citra memiliki tekstur yang kompleks, ia memerlukan operator LBP kecil, sedangkan jika citra memiliki keseragaman tekstur, operator LBP yang lebih besar menjadi pilihan yang tepat.

Kelas 23		Remak Daging (*Excecaria bicolor Hassk*)

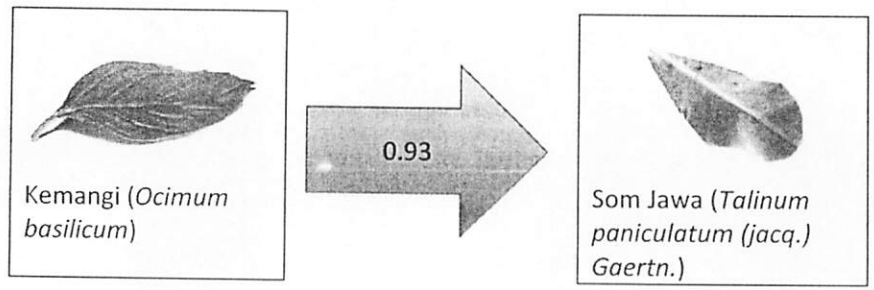
Gambar 23 Jenis tumbuhan kelas 23 remak daging

Tabel 5 Nilai operator LBP yang berbeda dalam kelas yang sama

Citra				
Operator LBP	(8,1)	(8,1)	(8,2)	(8,1)

Citra				
Operator LBP	(8,1)	(8,1)	(8,2)	(8,2)

Maksimalisasi objektif pada NSGA-II adalah nilai peluang pada fungsi PNN. Pengambilan nilai peluang yang dihasilkan dari fungsi PNN hanya berdasarkan nilai yang maksimal tanpa mempertimbangkan faktor apakah maksimal dengan kelas yang sama atau kelas yang lainnya. Sebagai contoh kelas 9 memiliki peluang sebesar 0.93 lebih besar dari yang lainnya akan tetapi masuk ke kelas yang salah.



SIMPULAN DAN SARAN

Berdasarkan hasil penelitian, dapat disimpulkan bahwa metode FLBP menggunakan NSGA-II mengalami peningkatan akurasi hingga sebesar 16%. Dari evaluasi dinyatakan bahwa operator LBP yang tepat diketahui bila tekstur citra yang memiliki tekstur kompleks, maka memerlukan operator LBP yang lebih detail, yaitu operator LBP (8,1). Evaluasi *threshold* FLBP menghasilkan pernyataan bahwa citra yang memiliki nilai kontras warna yang tinggi memerlukan nilai *threshold* FLBP yang tinggi pula. Nilai PNN yang digunakan untuk menentukan kelas bisa memiliki asumsi bahwa citra tersebut memiliki peluang yang salah dengan besarnya peluang yang citra tersebut miliki dibandingkan dengan kelasnya sendiri. NSGA-II mampu memilih operator LBP dan nilai *threshold* FLBP yang baik untuk setiap citra.

BAB 7
APLIKASI ALGEMO UNTUK PENGUJIAN JALUR PERANGKAT LUNAK

Dalam Bab 7 ini diberikan contoh studi kasus ALGEMO pada pengujian perangkat lunak (*software testing*) sampai pada tahap perancangan, sedangkan impelentasi dan evaluasi hasil diserahkan kepada pembaca untuk latihan dan eksplorasi. Aplikasi ini dipublikasikan dalam Hermadi I, Lokan CJ, dan Sarker RA (2010).

PENDAHULUAN

Dalam hal pengujian perangkat lunak, ada beberapa hal yang perlu disampaikan untuk mengingatkan kembali. Pertama, perangkat lunak yang telah lolos uji bukan berarti bebas dari kesalahan (*bug-free or error-free*), tetapi itu lebih menunjukkan bahwa perangkat lunak tersebut telah memperoleh kepercayaan untuk dapat berfungsi sesuai dengan yang diminta oleh penggunanya. Kedua, sebuah perangkat lunak hanya diuji terhadap kriteria tertentu saja, artinya belum tentu semua aspek pengujian dilakukan. Hal ini dilakukan karena beberapa keterbatasan berikut, misalnya waktu, tenaga, dan sumber daya lainnya, maka hanya kriteria yang dianggap penting dan relevan untuk tipe perangkat lunak tertentu saja akan dipilih.

PENGUJIAN JALUR PERANGKAT LUNAK

Seperti telah disampaikan pada subbab Pendahuluan, pengujian perangkat lunak memiliki beberapa kriteria. Secara umum, pengujian dibagi ke dalam dua kategori: kotak hitam (*black-box*) dan kotak putih (*white-box*). Pengujian kotak hitam melakukan pengujian aspek fungsionalitas dari perangkat lunak, di mana fungsi-fungsi yang harus disediakan akan diperiksa apakah telah sesuai dengan spesifikasi yang diberikan oleh (calon) penggunanya tanpa melihat proses apa yang terjadi di dalamnya, sehingga perangkat lunak diumpakan seperti kotak hitam. Sementara pengujian kotak putih, berlawanan dengan kotak hitam, perangkat lunak dianalogikan dengan kotak putih (tepatnya kotak transparan), di mana semua proses yang terjadi dapat terlihat dari luar. Jadi, pengujian kategori ini lebih pada pemeriksaan (kebenaran) struktur dan alur logika suatu proses dan disebut juga pengujian struktural (*structural testing*) atau pengujian kecakupan logika (*logic-coverage testing*) karena semua struktur di dalam perangkat lunak terlihat. Sebagai tambahan, dalam pengujian ini juga perlu disediakan kode sumber atau program (*source code*) dari perangkat lunak dan lebih baik lagi ada dokumen teknisnya untuk membantu proses pengujian.

Dalam kategori pengujian kotak putih, salah satu kelompok kriterianya adalah pengujian kecakupan (*coverage testing*). Kelompok pengujian ini terdiri atas beberapa tingkatan yang diurutkan dari yang terkecil ke yang terbesar lingkupnya (baca: yang lebih besar mencakup yang lebih kecil; *inclusivity character*): *statement, decision* (atau *branch*), *condition, decision-condition, multiple condition,* dan *path*. Jadi, dalam pengujian ini yang paling luas cakupannya adalah pengujian jalur (*path*).

Kriteria pengujian yang diangkat dalam kasus ini adalah pengujian jalur (*path testing*), di mana semua jalur logika dalam sebuah program harus dieksekusi minimal satu kali. Pada kasus program yang tidak memiliki pengulangan (*loops*; baik berupa pernyataan FOR, WHILE, maupun DO-WHILE), semua jalur logika dapat dituliskan satu persatu (*enumerated*). Sementara pada kasus program yang memiliki pengulangan, hal ini harus dibatasi agar banyaknya jalur tidak tak terhingga jumlahnya. Dalam hal ini, pada sebuah pengulangan, secara logika akan diuji ke dalam tiga jalur, yaitu jalur tanpa pengulangan, jalur dengan satu kali pengulangan, dan jalur dengan beberapa kali pengulangan yang diwakili dengan dua kali pengulangan saja.

Dalam pengujian jalur ini, beberapa hal yang harus dilakukan adalah konstruksi diagram aliran kendali (*CFG: Control Flow Graph*), pembangkitan jalur (*paths generation*), dan instrumentasi (*instrumentation*). Sementara dalam konteks ALGEMO, perlu ditentukan beberapa hal: fungsi *fitness*, inisialisasi populasi, seleksi, pindah silang, mutasi, dan kriteria pemberhentian. Semua penjelasan baik dari sisi pengujian jalur dan konteks ALGEMO akan diberikan pada subbab-subbab berikut.

GRAF ALIRAN KENDALI

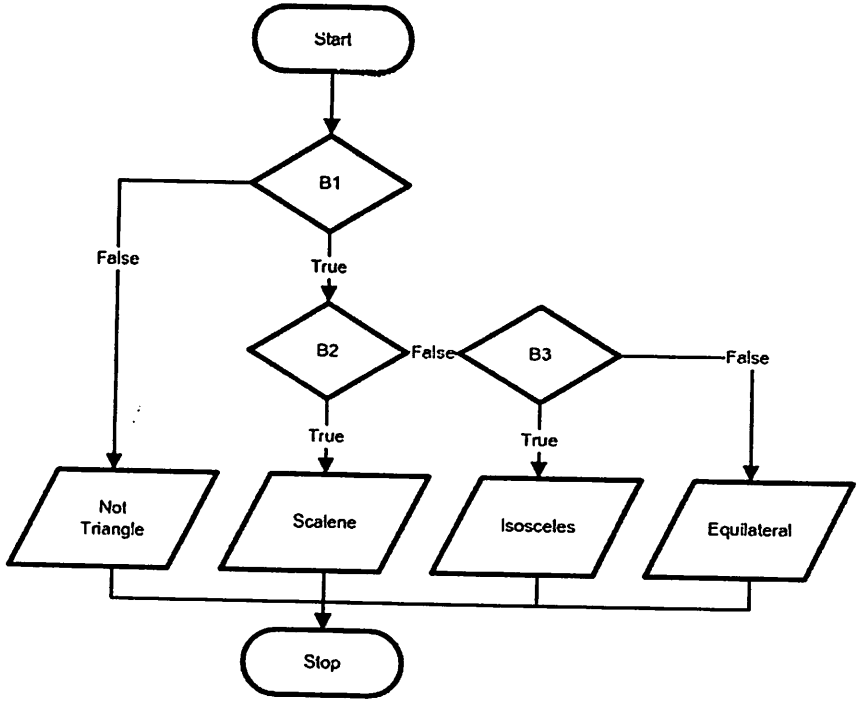
Pada kasus ini akan diberikan satu contoh program (dalam Matlab; C-*like language*) yang tanpa pengulangan (triangle.m). Program triangle.m adalah program yang berfungsi untuk menentukan apakah ketiga masukan (*input*) bilangan bulat yang diberikan membentuk segitiga atau tidak, kalau iya, apa tipe segitiganya: *scalene* (tidak sama sisi), *iscosceles* (sama kaki),

atau *equilateral* (sama sisi). Program ini memiliki tiga penyeleksian IF bersarang (*nested* IF *selection*). Berikut adalah program triangle.m (Gambar 24).

```
function [type] = triangle(sideLengths)
A = sideLengths(1); % First side
B = sideLengths(2); % Second side
C = sideLengths(3); % Third side
if ((A+B > C) && (B+C > A) && (C+A > B)) % Branch # 1
    if ((A ~= B) && (B ~= C) && (C ~= A)) % Branch # 2
        type = 'Scalene';
    else
        if (((A == B) && (B ~= C)) || ((B == C) && (C ~= A)) || ...
              ((C == A) && (A ~= B))) % Branch # 3
            type = 'Isosceles';
        else
            type = 'Equilateral';
        end
    end
else
    type = 'Not a triangle';
end
```

Gambar 24 Program triangle.m

Gambar 25 berikut adalah (*pseudo*) graf aliran kendali (GAK) dari program triangle.m, di mana dalam hal ini graf tersebut akan menampilkan aspek logika kendali yang terjadi dalam program. Setiap percabangan (penyeleksian atau keputusan) di dalam program diberi nama mulai dari yang dahulu muncul. Pada kasus ini, B1 (Branch # 1) adalah IF yang pertama (lihat Gambar 25), begitu pula dengan B2 dan B3. Setiap percabangan akan memiliki dua cabang yang bernilai ***True*** atau ***False***.



Gambar 25 (*pseudo*) Graf aliran kendali dari program triangle.m

PEMBANGKITAN JALUR

Setelah GAK terbentuk, langkah selanjutnya adalah melakukan pembangkitan jalur (dan pemilihan jalur jika memang diperlukan). Sebuah jalur logika dalam sebuah program direprsentasikan dalam bentuk sekuens pasangan cabang–keputusan (*a sequence of branch–decision pairs*) mulai dari awal masuk (*entry*) ke dalam program sampai keluar (*exit*) dari program. Sebagai contoh, lihat Gambar 2, satu jalur yang mungkin adalah **B1-F**(alse) atau ditulis lebih singkat [B1 0], di mana 1 adalah *True* dan 0 adalah *False*.

Seperti terlihat pada Gambar 2, maka program *triangle* ini seluruhnya memiliki 4 jalur berikut: [B1 0], [B1 1 B2 1], [B1 1 B2 0 B3 1], dan [B1 1 B2 0 B3 0]. Untuk program yang memiliki pengulangan, maka seperti telah dijelaskan di subbab sebelumnya, akan dibatasi pada tiga jenis pengulangan.

INSTRUMENTASI

Untuk melakukan *monitoring* jalur mana yang diambil oleh sebuah masukan pada saat eksekusi program, maka diperlukan tanda (atau *tag*) yang dapat memberikan informasi cabang mana yang diambil (1 atau 0) pada sebuah percabangan. Instrumentasi dilakukan dengan cara menyisipkan tanda-tanda tersebut ke dalam program, biasanya tepat sebelum sebuah percabangan. Selain untuk mengetahui cabang mana yang dilalui pada saat eksekusi, tanda-tanda tersebut juga digunakan untuk menghitung subnilai fitnes dari sebuah data masukan (*input data*).

Gambar 26 berikut menunjukkan program triangle.m yang telah diinstrumentasi. Pada gambar tersebut terlihat bahwa peubah traversedPath akan berupa larik (*array*) yang berisi pasangan cabang–keputusan.


```
function [traversedPath, type] = triangle(sideLengths)
traversedPath = [];
A = sideLengths(1); % First side
B = sideLengths(2); % Second side
C = sideLengths(3); % Third side
% instrument Branch # 1
traversedPath = [traversedPath 1 fitnessTriangle(1, A, B, C)];
if ((A+B > C) && (B+C > A) && (C+A > B)) % Branch # 1
    % instrument Branch # 2
    traversedPath = [traversedPath 2 fitnessTriangle(2, A, B, C)];
    if ((A ~= B) && (B ~= C) && (C ~= A)) % Branch # 2
        type = 'Scalene';
    else
        % instrument Branch # 3
        traversedPath = [traversedPath 3 fitnessTriangle(3, A, B, C)];
        if (((A == B) && (B ~= C)) || ((B == C) && (C ~= A)) || ...
              ((C == A) && (A ~= B))) % Branch # 3
            type = 'Isosceles';
        else
            type = 'Equilateral';
        end
    end
else
    type = 'Not a triangle';
end
```

Gambar 26 Program triangle.m yang telah diinstrumentasi

FUNGSI *FITNESS*

Fungsi *fitness* merupakan salah satu faktor utama yang ada dalam Algoritma Genetika yang paling menentukan kesuksesannya karena di sinilah terletak heuristik (informasi strategi pencarian) yang dimasukkan ke domain penyelesaian (*solution domain*) dari domain permasalahan (*problem domain*).

Dalam kasus ini, fungsi *fitness* terdiri atas dua tujuan (*multi-objective*), yaitu cakupan jalur (*path coverage*) dan minimalisasi pemakaian memori dinamis (*dynamic memory usage*). Untuk cakupan jalur, dua subfungsi digunakan, yaitu jarak cabang BD (*branch distance*) dan tingkat pendekatan AL (*approximation level*).

Jarak cabang dihitung menggunakan formula jarak Korel, seperti tercantum pada Tabel 6 berikut. Misal, sebuah kondisi cabang berisi pernyataan (X = 1), seandainya pada saat eksekusi X bernilai 10 dan cabang **True** yang ingin dilalui maka jaraknya adalah ABS(10-1) = 9 karena (X = 1) adalah tipe **Branch No 1**. Sementara tingkat pendekatan dihitung berdasarkan banyaknya cabang (*number of branches*) yang berbeda antara yang dilalui oleh sebuah masukan dengan target jalur yang diinginkan. Misal, pada Gambar 24 sebelumnya, target jalur adalah [B1 1 B2 0 B3 0], sedangkan masukan yang diberikan mengarahkan kendali pada jalur [B1 0], sehingga AL akan bernilai 2 karena berbeda pada cabang B1 dan cabang B2. Seandainya target jalur tetap sama, tetapi jalur yang dilalui oleh masukan adalah [B1 1 B2 0 B3 1], maka AL bernilai 0 karena semua cabang dilalui baik oleh jalur target maupun jalur masukan.

Jadi secara umum, fungsi *fitness* cakupan adalah membuat jarak cabang menjadi nol, begitu pula dengan tingkat pendekatan. Sementara fungsi *fitness* pemakaian memori adalah meminimalkan pemakaiannya, artinya tidak membuat jadi nol, tetapi mencari yang paling sedikit konsumsinya.

Tabel 6 Formula jarak Korel

No	Branch	Distance if path taken is different
1	A = B	ABS(A − B)
2	A ≠ B	K
3	A < B	(A − B) + k
4	A ≤ B	(A − B)
5	A > B	(B − A) + k
6	A ≥ B	(B − A)
7	X OR Y	MIN(Distance(X), Distance(Y))
8	X AND Y	Distance(X) + Distance(Y)

Pemakaian memori dinamis diketahui dengan cara mendefinisikan sebuah peubah pada awal program triangle.m dan menghitung berapa banyak *byte* yang diperlukan untuk mengeksekusi sebuah masukan. Proses untuk mendealokasikan peubah tersebut tidak diperhitungkan dalam pemakaian memori.

Gambar 27 berikut menunjukkan fungsi *fitness* fitnessTriangle.m untuk program triangle.m yang menggunakan fungsi jarak Korel. Fungsi tersebut menerima 4 masukan: pertama adalah nomor cabang dan yang ketiga berikutnya adalah masukan yang sama dengan triangle.m. Di dalamnya, ada tiga pilihan untuk pernyataan **SWITCH**, yaitu **case 1**, **case 2**, dan **case 3**. Setiap pilihan dalam **SWITCH** merepresentasikan masing-masing pemilihan (*selection*) dalam program triangle.m.

```
function branchVal = fitnessTriangle(branchNo, A, B, C)
    k = 1; % the smallest step for integer
    switch (branchNo)
      case 1,
         % branch #1: (A+B > C) & (B+C > A) & (C+A > B)
         term(1) = C - (A+B);
         term(2) = A - (B+C);
         term(3) = B - (C+A);
         for i=1:3
             if (term(i) < 0),
                term(i) = term(i) - k;
             else
                term(i) = term(i) + k;
             end
          end
          branchVal = sum(term);
          if (~((A+B > C) & (B+C > A) & (C+A > B)) & (branchVal < 0)), % Branch # 1
              branchVal = -branchVal;
          end
       case 2,
          % branch #2: (A ~= B) & (B ~= C) & (C ~= A)
          if (A ~= B), term(1) = 0; else term(1) = k; end
          if (B ~= C), term(2) = 0; else term(2) = k; end
          if (C ~= A), term(3) = 0; else term(3) = k; end
          branchVal = sum(term);
       case 3,
          % branch #3: ((A == B) & (B ~= C)) | ((B == C) & (C ~= A)) | ((C == A) & (A ~= B))
          if (A == B), subTerm(1) = 0; else subTerm(1) = abs(A-B); end
          if (B ~= C), subTerm(2) = 0; else subTerm(2) = k; end
          if (B == C), subTerm(3) = 0; else subTerm(3) = abs(B-C); end
          if (C ~= A), subTerm(4) = 0; else subTerm(4) = k; end
          if (C == A), subTerm(5) = 0; else subTerm(5) = abs(C-A); end
          if (A ~= B), subTerm(6) = 0; else subTerm(6) = k; end
          term(1) = subTerm(1) + subTerm(2);
          term(2) = subTerm(3) + subTerm(4);
          term(3) = subTerm(5) + subTerm(6);
          branchVal = min(term);
    end
end
```

Gambar 27 Fungsi *fitness* program triangle.m

INISIALISASI POPULASI

Populasi awal diinisialisasi dengan membangkitkan masukan secara acak sebanyak ukuran populasi *m* yang telah ditentukan, di mana setiap masukan adalah sebuah individu yang berupa tiga bilang bulat positif (panjang sisi segitiga tidak ada yang negatif) untuk program triangle.m. Dalam kasus ini, sebuah populasi dapat direpresentasikan sebagai matriks (baca: larik dua dimensi) dengan ukuran *m* x 3. Sebagai contoh, jika *m* = 3, maka akan ada 3 masukan, misal: [0 1 6], [100 23 33], dan [12 1034 234].

SELEKSI

Seleksi dapat dilakukan dengan metode Roda Rolet (*Roulette Wheel*), Turnamen, ataupun yang lainnya. Sangat disarankan untuk menggunakan elitisme pada proses seleksi ini, paling tidak dua individu terbaik atau proporsional terhadap ukuran populasi agar solusi yang paling baik pada generasi sekarang tidak serta merta menjadi hilang di generasi berikutnya.

Jarak antargenerasi (*generation gap*) untuk kasus ini bisa menggunakan 90% s.d. 95%, sesuai dengan keanekaragaman individu baru yang diinginkan. Individu-individu elite termasuk di dalam 5% s.d. 10% yang bukan jarak antargenerasi.

PINDAH SILANG

Tipe pindah silang (*crossover*) yang dapat dilakukan adalah satu sampai dua titikkarena maksimum hanya ada dua titik persilangan. Pindah silang dilakukan dengan mempertukarkan satu atau dua bilangan bulat positif dari masing-masing individu yang terlibat.

Laju pindah silang (*crossover rate*) yang dianjurkan adalah antara 0.8 sampai dengan 0.9. Ini memang sesuai dengan tujuan pindah silang, yaitu untuk melakukan eksplorasi terhadap domain masukan (*input domain*), sehingga ruang pencarian masukan program dapat dijelajahi secara eksploratif.

MUTASI

Mutasi dilakukan dengan cara mengurangi ataupun menambahkan bilangan bulat dalam jangkauan tertentu, misal 1 s.d. 5 terhadap salah satu komponen masukan (*input component*) yang dalam istilah Algoritma Genetika disebut dengan lokus (*locus*). Misal, jika diberikan sebuah individu berikut [3 100 78], maka salah satu mutasi yang mungkin adalah menambahkan komponen masukan kedua dengan 4 dan individu tersebut akan menjadi [3 104 78].

Seandainya hasil mutasi membuat sebuah individu menjadi tidak sah (*invalid*) untuk masukan program triangle.m, maka situasi ini harus dikembalikan ke domain individu yang sah dengan cara mengubah pengurangan menjadi penjumlahan (atau sebaliknya jika mencapai batas maksimum domain individu), memperkecil atau memperbesar bilangan bulat yang akan dikurangkan/ditambahkan, melipat/memutar kembali nilai-nya ke dalam agar masuk kembali ke domain individu yang sah, mengubah tandanya dari negatif menjadi positif. Misal, untuk individu [3 100 78], sebuah komponen individu pertama akan dikurangkan dengan 5, maka individu tersebut akan menjadi [-2 104 78] dan tidak sah. Dalam hal ini,

komponen tersebut bisa diubah menjadi +2 dan individu menjadi [2 104 78] atau membuat pengurangnya menjadi 2 dan individu menjadi [1 104 78].

KRITERIA PEMBERHENTIAN

Beberapa kriteria pemberhentian yang dapat dilakukan adalah cakupan jalur telah mencapai 100%, banyaknya generasi maksimum, durasi evolusi maksimum, laju kemajuan nilai *fitness* terbaik, dan banyaknya generasi stagnan.

Laju kemajuan nilai *fitness* terbaik adalah batas minimal kenaikan nilai *fitness* antargenerasi selama beberapa generasi. Misal, jika kenaikan nilai *fitness* kurang dari 0.03 selama lima generasi berturut-turut, proses evolusi dapat dihentikan.

Banyaknya generasi stagnan menunjukkan setelah beberapa generasi, misal 20 generasi berturut-turut tidak ditemukan lagi jalur baru yang belum pernah ditemukan sebelumnya.

IMPLEMENTASI

Kedua objektif cakupan jalur dan pemakaian memori dinamis dapat diagregasikan dengan teknik *weighted* GA ataupun pareto GA.

Untuk *weighted* GA, cakupan jalur dan pemakaian memori diberikan bobot sesuai dengan heuristik yang diberikan oleh masing-masing objektif. Misal 0.9 untuk cakupan jalur dan sisanya untuk pemakaian memori. Penentuan bobot yang optimal bisa didasarkan oleh kontribusi heuristik yang diberikan atau *trial-and-error*.

Sementara untuk pareto GA, dapat menggunakan NSGA II yang telah dikembangkan oleh Kalyanmov Deb untuk menggabungkan dua objektif tersebut.

KESIMPULAN

Penerapan AIGEMO untuk pengujian jalur perangkat lunak sangatlah menarik dan memberikan kontribusi yang besar dalam hal ini. Namun, perlu diperhatikan bahwa AIGEMO memerlukan setup yang tepat agar dapat berjalan baik.

Lebih jauh lagi, pengujian perangkat lunak biasanya membutuhkan waktu yang cukup lama, sehingga diperlukan percepatan dalam proses evolusinya. Salah satu tekniknya adalah dengan menerapkan paralel AIGEMO.

DAFTAR PUSTAKA

Adi WW. 2010. *Optimasi Jarak Cluster pada Data Intrusi Jaringan Menggunakan Multiobyektif Genetika Algoritma*. Jurusan Teknik Elektrik ITS.

Ahmed MA, Hermadi I. 2008. GA-based multiple paths test data generator. *Journal of Computers and Operations Research*. Vol. 35, pp. 3107–3124, Oct, Elsevier.

Alba E, Chicano F. 2007. Observations in using parallel and sequential evolutionary algorithms for automatic software testing. *Computers & Operations Research*. Vol. 35, pp. 3161–3183, Feb.

Aly AA. 2007. Applying genetic algorithm in query improvement problem. *International Journal of Information Technologies and Knowledge*. Vol. 1. 309–316.

Andria Y. 2007. *Optimasi Model Rantai Pasokan Agroindustri Cocodiesel dengan menggunakan Algoritma Genetika*. Bogor: Teknik Industri Pertanian, Institut Pertanian Bogor.

Andria Y, Arkeman Y, Gunawan H. 2009. *Optimization of Cocodiesel Supply Chain Model Using Genetic Algorithms*. Bogor: Indonesia Journal of Bioenergy.

Arkeman Y. 2013. SMART-TIN©: An Integrated And Intelligent System For The Design Of Adaptive Agroindustry (A Conceptual Framework). Bogor: Proceedings 2nd International Conference on Adaptive and Intelligent Agroindustry (ICAIA).

Arkeman Y, Kenneth De Jong. 2010. Development of Multiobjective Genetic Algorithms for Agri-Food Supply Chain Design by Considering Global Climate Change. Bogor: Proceedings AFITA 2010 International Conference.

Arkeman Y, Gunawan H, Satrio AB. 2010. Optimization of job-shop scheduling problem in agricultural product processing machineris industry using genetic algorithm. *Jurnal of Information Technology for Natural Resources Management*.

Arkeman Y, Tamura H. 2007. A New multiobjective genetic algorithm with heterogeneous population for solving flowshop scheduling problems. *International Journal of Computer Integrated Manufacturing*. Vol. 20. No. 5 :465–477.

Arkeman Y, Wahanani NA, Kustiyo A. 2012. Clustering K-means optimization with multi-objective genetic algorithm. *International Journal of Electrical Computer Science IJECS-IJENS*. Vol. 12. No. 05.

Arkeman Y, Yusuf A, Mushthofa, Laxmi GF, Seminar Kudang Boro. 2013. *The Formation of Optimal Portfolio of Mutual Shares Funds using Multi-Objective Genetic Algorithm*. Yogyakarta: Universitas Ahmad Dahlan.

Austin JE. 1992. *Agroindustrial Project Analysis; Critical Design Factors*. EDI Series in Economic Development. Baltimore and London: The Johns Hopkins University Press.

Bagchi TP. 1999. *Multiobjective Scheduling by Genetic Algorithms*. USA: Norwell, MAS, Kluwer Academic Publishers.

Bappenas. 2003. *Indonesia Boidiversity and Action Plan 2003–2020*. Jakarta: Bappenas.

Basuki APM, Arkeman Y. 2013. The Design and Implementation of Geographic Information System to Support Food and Energy Security. Bogor: Proceedings 2nd International Conference on Adaptive and Intelligent Agroindustry (ICAIA).

Berkhin P. 2002. Survey of Clustering Data Mining Technique." http://www.ce.ucr.edu/barth/EE242/clustering_survey.pdf.

Deb K, Agrawal RM. 1994. *Simulated Binary Crossover for Continuous Search Space*. IITK/ME/MD-94027. India.

Deb K. 2001. *Multi-Objective Optimization using Evolutionary Algorithm*. New York: John Willey and Son LTD.

Deb K, Pratap A, Agarwal S, Meyarivan T. 2002. A Fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans Evolutionary Comput*. Vol 6 : 182–197.

Dharmendra, Roy, Sharma L. 2010. Genetic K-Means clustering algorithm for data mixed numeric and categorical data sets. *IJAIA*. Vol. 1. No. 2.

Gen M, Cheng R. 1997. *Genetics Algorithms and Engineering Design*. Canada: John Wiley & Sons, Inc.

Genetic Algorithm based Test Data Generator, Master Thesis, KFUPM, Jun 2004.

Goldberg DE. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. New York : Addison Wesley.

Hadiguna RA. 2010. *Perancangan Sistem Penunjang Keputusan Rantai Pasok dan Penilaian Risiko Mutu pada Agroindustri Minyak Sawit Kasar*.

Haupt RL, Haupt SE. 2004. *Practical Genetic Algorithms*. New Jersey: John Willey.

Herdiyeni Y. 2010. Metodologi Pengukuran Kemiripan Citra Berbasis Semantik Menggunakan Representasi Tree [Disertasi]. Depok: Universitas Indonesia.

Herdiyeni Y, Kusmana I. 2013. Fusion of Local Binary Pattern Features for Tropical Medicinal Plants Identification. International Conference on Advance Computer Science and Information System. Bali, Indonesia. September 2013.

Herdiyeni Y, Wahyuni NKS. 2012. Mobile Application for Indonesian Medicinal Plants Identification using Fuzzy Local Binary Pattern and Fuzzy Color Histogram. International Conference on Advance Computer Science and Information System. Jakarta. December 2012.

Hermadi I, Ahmed MA. 2003. Genetic Algorithm Based Test Data Generator. Proceedings of the Congress on Evolutionary Computation 2003 (CEC2003), Canberra, Australia, 8–12 Dec 2003.

Hermadi I, Lokan CJ, Sarker RA. 2010. Genetic Algorithm Based Path Testing: Challenges and Key Parameters. Proceedings of the WCSE 2010 (2nd World Conference on Software Engineering), Wuhan, China, Dec 2010.

Hermawanto D. 2003. Algoritma Genetika dan Contoh Aplikasinya. Komunitas eLearning Ilmu Komputer.com. 2003–2007.

Iakovidis DK *et al*. 2008. *Fuzzy Local Binary Patterns for Ultrasound Texture Characterization*. Athens: University of Athens.

Jaya IK, Buono A, Arkeman Y. 2013. Precipitation Classification Using LVQ on Dry Season Based on Global Climate Indices Case Study in Indramayu District. Bogor: Proceedings 2nd International Conference on Adaptive and Intelligent Agroindustry (ICAIA).

Jogiyanto HM. 1998. *Teori Portofolio dan Analisis Investasi Edisi Pertama*. Yogyakarta: BPFE.

Kalyanmoy D. 1995. Simulated binary crossover for continuous search space. *Complex System*. 9:115 –148.

Keramidas EG *et al*. 2008. *Thyroid Texture Representation via Noise Resistant Image Feature*. Athens: University of Athens.

Klabbankoh B, Pinngern O. 1999. Applied Genetic Algorithms in Information Retrieval. Bangkok: Faculty of Information Technology King Mongkut, Institute of Technology Ladkrabang. *International Journal of the Computer, the Internet and Management*. Vol. 7. No. 3. September–December 1999.

Konak A, Coit DW, Smith AE. 2006. Multi-objective optimization using genetic algorithm: a tutorial. *Reliability Engineering and System Safety*. 91 :992–1007.

Korel B. 1990. Automated software test data generation. *IEEE Transactions on Software Engineering*. Vol. 16. No. 8, pp. 870–879, Aug 1990.

Kulsum LU. 2010. Identifikasi Tanaman Hias Secara Otomatis Menggunakan Metode Local Binary Patterns Descriptor dan Probabilistic Neural Network [Skripsi]. Institut Pertanian Bogor.

Kusmana I. 2011. Penggabungan Fitur Local Binary Pattern untuk Identifikasi Citra Tumbuhan Obat [Skripsi]. Institut Pertanian Bogor.

Lu Y, Lu S, Fotouhi F. 2004. FGKA: A Fast Genetic K-Means Clustering Algorithm. Proceedings of the 2004 ACM Symposium on Applied Computing. New York, USA: ACM. 622–623.

Markowitz H. 1952. Portfolio selection. *The Journal of Finance*. Vol. 7. No. 1, pp: 77–91. March.

Mitchell M. 1998. *An Introduction to Genetic Algorithms*. London: Massachusetts Institute of Technology.

Musaroh. 2007. *Jurnal: Kajian Perbandingan antara Reksa Dana Syariah dan Reksa Dana Konvensional sebagai Solusi Alternatif Perencanaan Investasi*.

Owais SSJ, Kromer P, Snasel V. 2005. Query Optimization by Genetic Algorithm. *DATESO*. 2005: 125–137.

Pinedo M. 1995. *Scheduling: Theory, Algorithms and Systems*. Englewood Cliffs, NJ: Prentice-Hall.

Pratomo EP, Ubaidillah N. 2004. *Reksadana: Solusi Perencanaan Investasi Era Modern*. Jakarta: Penerbit PT Gramedia Pustaka Utama.

Rahmanti, Farah Z, Entin M. 2010. *Pengelompokkan Gambar Berdasarkan Warna dan Bentuk Menggunakan FGKA Clustering*. Jurusan Teknik Informatika, Politeknik Elektronika Negeri Surabaya-ITS.

Salazar EJ, Velez AC, Parra MCM, Ortega LO. 2002. *A Cluster Validity Index for Comparing Non Hierarchical Clustering Methods*.

Srinivas N, Deb K. 1994. Multi-objective optimization using non-dominated sorting in genetic algorithm. *Journal of Evolutionary Computation*. Vol. 2 No. 3, pages: 221–248.

Suyanto. 2005. *Algoritma Genetika dalam Matlab*. Yogyakarta: Andi Yogyakarta.

Valerina F. 2012. Perbandingan Local Binary Pattern dan Fuzzy Local Binary Pattern untuk Ekstraksi Citra Tumbuhan Obat [Skripsi]. Institut Pertanian Bogor.

Utami TP, Ma'arif S, Arkeman Y, Hartoto L. 2013. Design Of Grouping Traditional Market Distribution Using Fuzzy Clustering And Design Of Routing Of Packaging Cooking Oil From Distribution Center To Traditional Market Using Traveling Salesperson Problem – Genetic Algorithm in Indonesia (Case-Jakarta). Bogor: Proceedings 2nd International Conference on Adaptive and Intelligent Agroindustry (ICAIA).

Utomo P. 2010. *Peluang dan Tantangan Pertumbuhan Reksadana di Indonesia*.

Widodo Y. 2008. *Pencarian Gambar Berdasarkan Fitur Warna Dengan GA-KMeans Clustering*. Jurusan Teknologi Informasi, Politeknik Elektronika Negeri Surabaya.

Wu SG *et al.* 2007. *A Leaf Recognition Algorithm for Plant Using Probabilistic Neural Network*. China: Chinese Academy Science.

Wulansari D, Entin M, Nana R. 2011. *Pengelompokkan Gambar berdasarkan Fitur warna dan tekstur dengan FGKA Clustering*.

Xu Rui. 2009. *Clustering*. John Wiley & Sons.

Zavaschi THH *et al.* 2011. Facial expression recognition using ensemble of classifier. *IEEE*. 1489–1492.

Zuhud EAM. 2009. Potensi hutan tropika Indonesia sebagai penyangga bahan obat alam untuk kesehatan bangsa. *Jurnal Bahan Alam Indonesia*. Vol. VI. No. 6. Januari 2009.

PROFIL PENULIS

Dr. Ir. Yandra Arkeman, MEng

Lahir di Payakumbuh 14 September 1965. Lulusan Jurusan Teknologi Industri Pertanian, Institut Pertanian Bogor (IPB) tahun 1989 dengan skripsi tentang *Computer Aided Quality* (CAQ) untuk Industri Minuman Ringan. Tahun 1994 penulis mulai melanjutkan studi untuk jenjang S-2/S-3 di Adelaide, Australia. Penulis meraih gelar Master (1996) dan Doktor (2000) dengan riset di bidang *Intelligent Manufacturing Systems* dari University of South Australia. Selanjutnya pada tahun 2004–2006 penulis melakukan penelitian tingkat *post-doctoral* di Applied System Design Laboratory, Department of Electrical Engineering and Computer Science, Kansai University, Osaka, Japan dengan topik *Multi-objective Genetic Algorithms for Agroindustrial Supply Chain Design*. Kemudian pada tahun 2009 penulis mendapat kesempatan untuk melakukan penelitian *post-doctoral* yang kedua selama 4 bulan di Department of Computer Science dan Krasnow Institute for Advanced Study, George Mason University, Virginia, USA dengan topik tentang *Parallel Genetic Algorithms for Agroindustrial System Design*. Pada tahun 2003 penulis mendirikan CIGARIS (Computational Intelligence Group for Advanced Research and Innovations in Supercomputing Technology) dengan alamat web: www.cigaris.com. Sejak tahun 2007, penulis menjadi peneliti pada Surfactan and Bioenergy Research Centre (SBRC), IPB dengan fokus penelitian pada Intelligent Modeling for Bioenergy Sustainability.

Penulis aktif di berbagai organisasi profesi tingkat nasional dan internasional sejak tahun 1994, di antaranya Institute of Industrial Engineers (IIE), Society of Manufacturing Engineers (SME), dan International Society of Systems Science (ISSS). Riwayat hidupnya dimasukkan di buku biografi *Who's Who in Science and Engineering 1998/1998* terbitan Marquis Publication, USA.

Penulis menikah dengan Ir. Titi Sulistyowati Rahayu dan dikarunia satu orang putra yang bernama Ryan Muhammad Khawarizmi, mahasiswa Teknik Fisika ITB tahun 2011. Selain melakukan penelitian, waktu luangnya diisi dengan bermain tenis dan *traveling*.

Dr. Yeni Herdiyeni, S.Si, M.Komp

(yeni.herdiyeni@ipb.ac.id). Lahir di Bogor, 23 September 1975. Penulis lulusan Departemen Ilmu Komputer FMIPA IPB pada tahun 1998. Setelah menyelesaikan studi S-1, sampai saat ini penulis menjadi staf pengajar di Departemen Ilmu Komputer, FMIPA IPB. Pada tahun 2002, penulis melanjutkan pendidikan S-2 di Magister Ilmu Komputer Fakultas Ilmu Komputer, Universitas Indonesia. Penulis meraih gelar Master pada tahun 2005 dengan judul tesis Metode Garis Wajah untuk Sistem Pengenalan Wajah 3D menggunakan Probability Principal Component Analysis dan Jaringan Syaraf Tiruan. Pada akhir tahun 2005 penulis melanjutkan studi S-3 di Program Doktor Fakultas Ilmu Komputer Universitas Indonesia dan meraih gelar Doktor pada awal 2010 dengan judul disertasi Metodologi Pengukuran Kemiripan Citra Berbasis Semantik Menggunakan Representasi Tree. Pada tahun 2011, penulis melanjutkan *post-doctoral* di Graduate School of Science and Engineering, Department of Information Science, Saga University, Japan dengan topik penelitian *Lung Cancer Identification using 3D Local Binary Pattern*. Sejak tahun 2013 penulis menjadi peneliti di Pusat Studi Biofarmaka LPPM IPB. Saat ini penulis sedang melakukan penelitian dalam bidang pengolahan citra digital untuk identifikasi tumbuhan dan pengembangan sistem *Biodiversitas Informatics* di IPB.

Dr. Irman Hermadi

Current: **Lecturer** at **Bogor Agricultural University**

Past: Research Publication Fellow at UNSW@ADFA, PhD Candidate at UNSW@ADFA, Casual teaching or laboratory demonstrator at UNSW@ADFA Education

- University of New South Wales
- King Fahd University of Petroleum & Minerals (KFUPM)
- Bogor Agricultural University

I have over 14 years of academic, research, and professional experience in computer science and software engineering disciplines. I have taught various courses in the area, been conducting research in evolutionary path testing, and provided several consultations on industrial-scale information and communication technology (ICT) projects. My involvement in the projects includes system analyst, database consultant, independent testing team member, and web-based software developer. This experience provides me some pedagogical expertise, ability to conduct research

independently, and professionalism knowledge. My achievements are reflected on the held positions, supervised graduates, received awards, research publications, and completed ICT projects. I am able to communicate effectively with other civitas academika, colleagues, software stake holders, software users, managerial staffs, and executives. Currently, I am research publication fellow with the SEIT UNSW Canberra and associate member of the Australian Computer Society (ACS).

Specialties

Software engineering, information technology, database analyst, business process analyst, optimization methods.

Gibtha Fitri Laxmi, S.Kom., M.Kom

Lahir di Jakarta, 12 Juli 1986 anak kedua dari pasangan Suroso dan Yulinar. Penulis lulusan S-1 Departemen Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan, Institut Pertanian Bogor. Judul skripsi yang diteliti adalah Implementasi Algoritma Genetika pada Temu Kembali Citra, di bawah bimbingan Dr. Yeni Herdiyeni, S.Si., M.Kom dan Dr. Ir. Agus Buono, M.Si., M.Kom. Setelah lulus, penulis mengikuti proyek Sistem Informasi Geografis di *Oil Palm Research*, Mekarsari. Tahun 2009 sampai saat ini penulis menjadi asisten Dr. Ir. Yandra Arkeman untuk melakukan penelitian mengenai optimasi menggunakan metode Algoritma Genetika. Pada tahun yang sama, penulis menjadi guru di SMPN 4 Bogor sampai 2011. Tahun 2010–2012 penulis melanjutkan studi S-2 di Pascasarjana, Institut Pertanian Bogor dengan

judul tesis yang diteliti Optimasi Pemilihan *threshold* dan operator FLBP menggunakan *Multi-Objective Genetic Algorithm,* dibimbing oleh Dr. Yeni Herdiyeni, S.Si., M.Kom. dan Dr. Ir. Yandra Arkeman, M.Eng. Sampai saat ini, penulis menjadi dosen tetap di Universitas Ibn Khaldun, Dosen Luar Biasa di Universitas Pakuan dan Universitas Trisakti dengan spesialis pada bidang kecerdasan buatan, sistem pakar, sistem penunjang keputusan. Penulis telah mendapatkan sertifikat SNI untuk multimedia. Penulis menyukai pemrograman, dengan menguasai pemrograman MATLAB, C, Java, VB, C++, dan XAML. Aktivitas lain yang digemari penulis adalah bermain musik (organ, piano, gitar), membaca, berolahraga (bersepeda, berenang, dan *jogging*).