PENERAPAN ALGORITMA GENETIKA UNTUK KOMPRESI CITRA FRAKTAL

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer
dalam bidang Ilmu Komputer

Oleh:
PUTU INDAH CIPTAYANI
0410960044 – 96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2008

LEMBAR PENGESAHAN SKRIPSI

PENERAPAN ALGORITMA GENETIKA UNTUK KOMPRESI CITRA FRAKTAL

Oleh:
PUTU INDAH CIPTAYANI
0410960044 – 96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 5 Agustus 2008
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Dosen Pembimbing I,	Dosen Pembimbing II,
Wayan F. Mahmudy, SSi, MT	Agus Wahyu Widodo,ST
NIP. 132 158 724	NIP 132 295 994

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Putu Indah Ciptayani
NIM : 0410960044-96
Jurusan : Matematika
Program Studi : Ilmu Komputer
Penulis tugas akhir berjudul :PENERAPAN ALGORITMA GENETIKA UNTUK KOMPRESI CITRA FRAKTAL.

Dengan ini menyatakan bahwa :

1. **Isi dari tugas akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Tugas Akhir ini.**
2. **Apabila dikemudian hari ternyata Tugas Akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 5 Agustus 2008
Yang menyatakan,

Putu Indah Ciptayani
NIM. 0410960044-96

UNIVERSITAS BRAWIJAYA

PENERAPAN ALGORITMA GENETIKA UNTUK KOMPRESI CITRA FRAKTAL

ABSTRAK

Kompresi citra fraktal merupakan salah satu teknik kompresi citra. Metode ini memanfaatkan konsep citra fraktal, yaitu suatu citra yang memiliki kemiripan terhadap dirinya sendiri dalam skala yang berbeda. Teknik kompresi citra fraktal membagi citra menjadi sejumlah blok yang berukuran sama dan tidak saling beririsan disebut dengan blok jelajah, kemudian untuk setiap blok jelajah dicari kemiripannya dengan bagian blok yang berukuran lebih besar disebut dengan blok ranah. Setelah didapatkan blok ranah yang paling mirip, kemudian diturunkan transformasinya yang disebut dengan transformasi *affine.* Transformasi *affine* akan disimpan ke dalam berkas. Permasalahan yang ditemui di dalam kompresi citra fraktal yaitu waktu yang dibutuhkan untuk mencocokan blok jelajah dengan blok ranah sangat lama. Karena itu dilakukan pendekatan dengan algoritma genetika untuk pencocokan blok jelajah dan blok ranah.

Algoritma genetika diterapkan dengan tiga metode *crossover* dan tiga metode mutasi. Jumlah individu di dalam populasi sebanyak 5, probabilitas mutasi 0.1, probabilitas *crossover* 0.5, 0.6, 0.7, 0.8, 0.9 dan 1.0. Maksimal generasi adalah sampai 500. Ukuran blok jelajah adalah 4 pada citra Lena1.bmp yang berukuran 117 x 149 *pixel* dan ukuran *file* 52 KB.

Hasil pengujian menunjukkan bahwa probabilitas *crossover* memberikan pengaruh yang berbeda terhadap masing-masing metode *crossover* dan mutasi. Sedangkan metode yang mampu menghasilkan transformasi *affine* terbaik adalah metode *crossover* dengan memperhatikan *fitness* induk dan metode mutasi dengan pencarian optimum lokal dengan probabilitas *crossover* 0.5. Ukuran *file* yang dihasilkan adalah 13 KB atau rasionya adalah 75,44%, waktu kompresi yaitu 10.7 detik, sedangkan MSE sebesar 0.158839.

IMPLEMENTATION OF GENETIC ALGORITHM IN FRACTAL IMAGE COMPRESSION

ABSTRACT

Fractal image compression is one of the image compression techniques. This method use fractal image concept, which an image have similarity to itself on different scales. Fractal image compression technique divide an image into a number of non overlapping block which have same size namely range block, then for each range block is searched its similarity with block that have larger size called domain block. After the most similar domain block is found, the transformation is derived and called affine transformation. Affine transformation is saved into a file. The problem is time needed to find the matching of range and domain block is too long. Because of it, necessary to get approach with genetic algorithm to find the matching of range and domain block.

Genetic algorithm is implemented by three crossover methods and three mutation methods. The number of individual in population is 5, mutation probability is 0.1, crossover probabilities are 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0. Then maximum generation is 500. The size of range block is 4, apply on image Lena1.bmp which size is 117 x 149 pixel and file size is 52 KB.

The result of examination show that crossover probabilities give different influence for each crossover and mutation method. The method which produce the best affine transformation is crossover method which concern the parent fitness and mutation which search the local optimum with crossover probability 0.5. File size after compression process is 13 KB or the ratio is 75,144%, compression time is 10.7 second, and MSE is 0.158839.

KATA PENGANTAR

Puji syukur Penulis panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat-Nya, sehingga Penulis dapat menyelesaikan tugas akhir ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer

Tugas akhir ini bertujuan untuk menerapkan algoritma genetika dalam kompresi citra fraktal dengan menggunakan perangkat lunak.

Pada penyusunan tugas akhir ini, Penulis mengucapkan terima kasih kepada :

1. Wayan Firdaus Mahmudy, S.Si., MT., selaku pembibing utama penulisan tugas akhir sekaligus sebagai Ketua Program Studi Ilmu Komputer, Jurusan Matematika, FMIPA Universitas Brawijaya.
2. Agus Wahyu Widodo, ST., selaku pembimbing pendamping dalam penulisan tugas akhir.
3. Achmad Basuki, ST., selaku penasehat akademik.
4. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
5. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan tugas akhir ini.
6. Orang tua Penulis atas dukungan materi dan doa restunya kepada Penulis.
7. Rekan-rekan di Program Studi Ilmu Komputer FMIPA Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan tugas akhir ini.
8. Dan semua pihak yang telah membantu dalam penyusunan tugas akhir ini yang tidak dapat Penulis sebutkan satu per satu.

Penulis menyadari bahwa masih banyak kekurangan dalam laporan ini, oleh karena itu Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi tugas akhir ini untuk kelanjutan penelitian serupa di masa mendatang.

Semoga laporan tugas akhir ini dapat bermanfaat.

Malang, 5 Agustus 2008

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR SOURCE CODE

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada umumnya representasi citra digital memerlukan memori penyimpanan yang besar. Misalnya citra dalam format bitmap yang berukuran 512 x 512 *pixel* membutuhkan memori sebesar 32KB untuk representasinya (Rinaldi Munir, 2004). Tentu saja jika ukuran citra semakin besar, maka memori yang dibutuhkan untuk menyimpan citra tersebut juga akan semakin besar. Citra yang memiliki ukuran penyimpanan yang besar tentu saja akan memerlukan waktu yang sangat lama untuk pengirimannya, sehingga sangat menyulitkan pengguna.

Untuk menangani besarnya memori yang digunakan dalam merepresentasikan citra, bisa dilakukan kompresi citra. Citra bisa dikompres atau dimampatkan untuk meminimalkan kebutuhan memori untuk merepresentasikan citra digital. Prinsip umum yang digunakan pada proses kompresi citra adalah mengurangi duplikasi data dalam citra sehingga memori yang dibutuhkan untuk merepresentaikan citra menjadi lebih sedikit daripada representasi citra semula.

Hingga saat ini sudah ada banyak metode kompresi citra. Metode kompresi citra diklasifikasikan menjadi dua yaitu metode *lossless* dan *lossy*. Metode *lossless* akan menghasilkan citra hasil dekompres yang tepat sama dengan citra semula, *pixel* per *pixel*, atau dengan kata lain tidak ada informasi yang hilang akibat kompresi. Sedangkan metode *lossy* menghasilkan citra hasil dekompresi yang hampir sama dengan citra semula. Ada informasi yang hilang akibat kompresi, tetapi dapat ditolerir oleh persepsi mata (Rinaldi Munir, 2004). Beberapa contoh metode kompresi yang bertipe *lossless* yaitu bitmap, *Portable Graymap Format* (pgm), *Portable Pixmap Format* (ppm), *Portable Network Graphics* (png), *Tagged Image File Format* (tiff), dan JPEG-2000. Sedangkan contoh kompresi *lossy* yaitu *Graphics Interchange Format* (gif), JPEG berbasis DCT, *Tagged Image File Format* (tiff), dan kompresi fraktal.

Metode kompresi fraktal didasarkan pada teori fraktal, di mana citra fraktal adalah sebuah citra memiliki kemiripan dirinya sendiri namun dalam skala yang berbeda (*self-similarity*). Sebuah citra alami hampir tidak ada yang berupa citra fraktal, namun citra alami

biasanya memiliki kemiripan lokal, yaitu ada bagian kecil dari citra yang mirip dengan bagian citra yang lebih besar. Jadi konsep kompresi fraktal adalah memanfaatkan kemiripan lokal tersebut dan menghitung transformasi yang memetakan bagian-bagian citra yang memiliki kemiripan tersebut yang disebut dengan transformasi *affine* (IFS lokal) w_i. Metode kompresi fraktal membagi citra menjadi sejumlah blok (bagian) yang berukuran sama dan tidak saling beririsan disebut blok jelajah (*range block*). Kemudian citra yang sama juga dibagi menjadi sejumlah blok dengan ukuran yang lebih besar daripada blok jelajah, dimana blok ini saling beririsan dan blok ini disebut dengan blok ranah (*domain block*). Semua blok ranah dikumpulkan, dankumpulan blok ranah ini disebut dengan pul ranah. Untuk setiap blok jelajah dihitung nilai kemiripannya dengan semua blok ranah di dalam pul ranah. Jika sudah didapatkan blok ranah yang paling mirip dengan blok jelajah yang sedang ditinjau, kemudian diturunkan transformasi *affine*-nya. Hasil dari semua pemasangan blok jelajah dengan blok ranah yang paling mirip disebut dengan *Partitioned Iterated Function System* (PIFS). Metode kompresi fraktal yang digunakan untuk citra abu-abu berbeda dengan metode kompresi fraktal yang dilakukan untuk citra berwarna. Metode yang dilakukan untuk citra berwarna adalah dengan mentransformasi model RGB ke model YIQ, XYZ atau IHS. Pemampatan fraktal cukup dilakukan terhadap komponen luminasinya (Y pada model YIQ, Y pada model XYZ, atau I pada model IHS). Dua komponen sisanya dimampatkan dengan metode berbeda (Rinaldi Munir, 2004).

Kompresi fraktal mampu menghasilkan rasio kompresi yang tinggi dan metode dekompresinya sederhana. Namun metode ini memiliki kelemahan yaitu lamanya waktu kompresi yang disebabkan oleh proses pencocokan blok jelajah dan blok ranah menggunakan algoritma *brute force*, yaitu mencoba berbagai macam kemungkinan. Seperti yang diungkapkan oleh Lifeng Xi dan Liangbin Zhang (2007) bahwa permasalahan utama dalam kompresi fraktal saat ini adalah bagaimana memilih dan mengoptimasi klasifikasi dari blok jelajah, menyeimbangkan kecepatan kompresi dan dekompresi, meningkatkan rasio kompresi dan memperbaiki kualitas citra setelah dekompresi. Permasalahan pencarian antara blok jelajah dengan blok ranah yang cocok jika dilakukan dengan metode *brute force* akan membutuhkan banyak waktu karena dilakukan evaluasi terhadap

semua kemungkinan yang ada. Hal ini akan sangat bermasalah jika ukuran citra semakin besar. Untuk itu, diperlukan metode yang bisa mengurangi kompleksitas waktu pencarian antara blok jelajah dan blok ranah.

Salah satu metode yang bisa dilakukan yaitu metode pendekatan heuristik dengan algoritma genetika. Algoritma genetika merupakan teknik pendekatan heuristik yang memanfaatkan konsep evolusi dan seleksi alam. Menurut Lifeng Xi dan Liangbin Zhang (2007), algoritma genetik adalah algoritma stokastik yang mensimulasikan proses evolusi alami, yang biasanya diaplikasikan untuk optimasi parameter terkontrol dan fungsi konstrain. Langkah pertama dalam algoritma genetik adalah membangkitkan beberapa solusi yang mungkin. Sejumlah solusi yang mungkin ini disebut dengan populasi. Suatu solusi di dalam populasi, nantinya akan disebut sebagai kromosom. Masing-masing kromosom akan memiliki suatu nilai tertentu yang disebut *fitness*. Untuk mencari nilai *fitness* tersebut, diperlukan sebuah fungsi *fitness*, di mana fungsi *fitness* ini akan ditentukan oleh permasalahan yang sedang diselesaikan. Untuk membentuk populasi yang baru, maka akan dilakukan perkawinan silang antar kromosom sesuai dengan peluang perkawinan silang yang bisa ditentukan oleh pengguna, peluang tersebut disebut dengan probabilitas perkawinan silang. Untuk memilih kromosom mana yang akan menjadi induk, maka dilakukan seleksi. Ada beberapa metode seleksi, salah satunya yaitu metode *roulette wheel* di mana dalam metode ini kromosom yang memiliki nilai *fitness* tertinggi akan memiliki peluang yang paling besar untuk terpilih. Hasil dari perkawinan silang yaitu berupa kromosom yang baru yang disebut dengan *offspring*. Nilai *fitness* dari *offspring* yang terbentuk akan dibandingkan dengan nilai *fitness* dari semua kromosom yang telah ada dalam populasi, jika nilai *fitness offspring* lebih besar daripada nilai *fitness* individu di dalam populasi, maka *offspring* tersebut akan masuk ke dalam populasi menggantikan individu dengan nilai *fitness* terkecil di dalam populasi. Untuk menghindari konvergenitas di awal, maka bisa dilakukan mutasi dengan peluang tertentu yang disebut dengan probabilitas mutasi. Langkah tersebut akan diulangi sampai kondisi yang diinginkan tercapai.

Dalam kompresi fraktal, algoritma genetik diaplikasikan untuk mencari blok jelajah dengan blok ranah yang cocok. Untuk setiap blok jelajah, strategi yang dilakukan algoritma genetika adalah

dengan mengkodekan posisi pencarian blok ranah, mendefinisikan *fitness* untuk jarak minimum pencocokan blok jelajah dengan blok ranah yang dicari. Dengan menerapkan algoritma genetika dalam kompresi citra fraktal, diharapkan waktu pencarian blok jelajah dan blok ranah bisa lebih cepat, sehingga kompresi citra memerlukan waktu yang lebih singkat dan waktu antara kompresi dengan dekompresi bisa menjadi seimbang.

1.2 Rumusan Masalah

Rumusan masalah dalam skripsi ini adalah :

1. Bagaimana menerapkan kompresi fraktal untuk mengkompresi dan mendekompresi citra?
2. Bagaimana menerapkan algoritma genetika untuk mengoptimalisasi waktu pencocokan blok jelajah dengan blok ranah dalam kompresi citra fraktal?

1.3 Batasan Masalah

Berikut ini adalah batasan masalah dari permasalahan di atas untuk menghindari terjadinya melebarnya pembahasan dari yang seharuanya :

1. Citra yang digunakan hanya citra abu-abu (*grayscale*).
2. Blok jelajah dan blok ranah berbentuk bujur sangkar.
3. Operasi aritmatika antara blok ranah dan blok jelajah tidak dievaluasi
4. Untuk proses seleksi digunakan metode *roulette wheel*.

1.4 Tujuan Penulisan

Tujuan penulisan skripsi ini adalah menerapkan algoritma genetik dalam mengoptimalisasi waktu pencocokan blok jelajah dengan blok ranah dalam kompresi citra fraktal.

1.5 Manfaat Penulisan

Menyediakan sebuah perangkat lunak untuk kompresi citra abu-abu dengan waktu kompresi yang lebih efisien.

1.6 Metodologi Penulisan

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah:

1. Studi Literatur

Mempelajari teori-teori yang berhubungan dengan konsep kompresi citra fraktal dan algoritma genetik dari berbagai referensi.

2. Pendefinisian dan analisis masalah

Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat dalam memanfaatkan algoritma genetika sebagi teknik pencarian heuristik untuk mengoptimasi waktu pencarian antara blok jelajah dengan blok ranah.

3. Perancangan dan implementasi sistem

Menentukan metode representasi kromosom yang digunakan, menentukan fungsi *fitness* dan menentukan teknik perkawinan silang yang digunakan untuk kompresi fraktal agar hasil yang didapatkan lebih baik. Kemudian membuat perancangan perangkat lunak dan mengimplementasikan hasil rancangan tersebut yaitu membuat perangkat lunak kompresi citra.

4. Uji coba dan analisa hasil implementasi

Menguji perangkat lunak, dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya, dan kemudian dievaluasi.

5. Penyusunan Laporan

Membuat laporan tertulis mengenai hasil skripsi ini

1.7 Sistematika Penulisan Skripsi

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang masalah penerapan algoritma genetika untuk meoptimasi waktu pencarian blok jelajah dan blok ranah, batasan masalah yang akan dibahas berkenaan dengan kompresi citra fraktal dengan algoritma genetik, tujuan penulisan, manfaat penulisan, metodologi penulisan dan sistematika penulisan.

BAB II DASAR TEORI

Bab ini berisi tentang teori-teori yang menjadi acuan untuk pelaksanaan penulisan skripsi yang meliputi teori tentang definisi fraktal, *Iterated Function System*, *Partitioned Iterated Function System* untuk kompresi citra fraktal, rekonstruksi citra fraktal,

definisi algoritma genetik, macam-macam pengkodean algoritma genetik, macam-macam metode seleksi, macam-macam metode perkawinan silang dan mutasi untuk masing-masing pengkodean, penggunaan algoritma genetika untuk kompresi citra fraktal.

BAB III METODOLOGI DAN PERANCANGAN SISTEM

Bab ini berisi tentang aliran proses atau alur algoritna genetika dalam menyelesaikan permasalahan kompresi fraktal sesuai dengan dasar teori yang telah dibahas di bab 2 dan studi kasus kompresi citra fractal, penyimpanan ke dalam berkas serta dekompresi citra.

BAB IV IMPLEMENTASI DAN UJI COBA SISTEM

Bab ini membahas tentang pengujian dan analisa kinerja dari perangkat lunak kompresi citra fraktal dengan algoritma genetik yang telah dibuat baik dari antar muka maupun proses serta pembahasan prosedur dari algoritmanya.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan dari pembahasan yang telah dilakukan di bab 4 serta saran pengembangan dari keseluruhan tahapan pembuatan skripsi ini .

BAB II
TINJAUAN PUSTAKA

2.1 Definisi Fraktal

Teknik kompresi citra fraktal diperkenalkan oleh Barnsley. Teknik kompresi fraktal memakai konsep citra fraktal yaitu sebuah citra yang memiliki bagian yang mirip dengan keseluruhan citra**.** Sedangkan konsep fraktal pertama kali diperkenalkan oleh Benoit Mandelbrot.

Menurut Rinaldi Munir (2004), fraktal dapat didefinisikan dari dua buah properti yaitu :

1. *Self similarity*

Fraktal adalah objek yang memiliki kemiripan dirinya sendiri namun dalam skala yang berbeda. Ini artinya, bagian-bagian dari objek akan tampak sama dengan objek itu sendiri bila diihat secara keseluruhan.

2. Matra (*dimension*)

Fraktal adalah obyek yang memiliki matra bilangan riil atau pecahan.

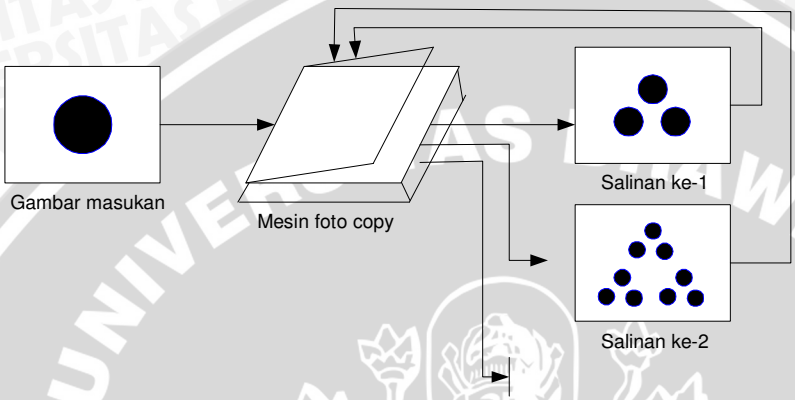
Beberapa contoh fraktal ditunjukkan oleh Gambar 2.1

Gambar 2.1 Segitiga Sierpinski, pakis Barnsley dan pohon fraktal

2.2 *Iterated Function System* (IFS)

Michael Barnsley merepresentasikan fraktal ke dalam model matematika yang dinamakan IFS (*Iterated Function System*) (Rinaldi Munir, 2004). IFS digambarkan sebagai sebuah mesin *foto copy* yang memiliki banyak lensa dan disebut dengan *Multiple Reduction Copy Machine* (MRCM). Setiap lensa dalam MRCM melakukan

pengecilan gambar dalam jumlah yang banyak. Gambar yang dihasilkan dari mesin *foto copy* dioperasikan kembali sebagai masukan untuk membuat salinan berikutnya. Ilustrasinya ditunjukkan oleh Gambar 2.2



Gambar 2.2 *Multiple Reduction Copy Machine*(MRCM)

Gambar apapun yang dimasukkan ke mesin MRCM sebagai inputan awal, akan konvergen ke gambar akhir yang sama, yaitu segitiga Sierpinski seperti ditunjukkan oleh Gambar 2.3.

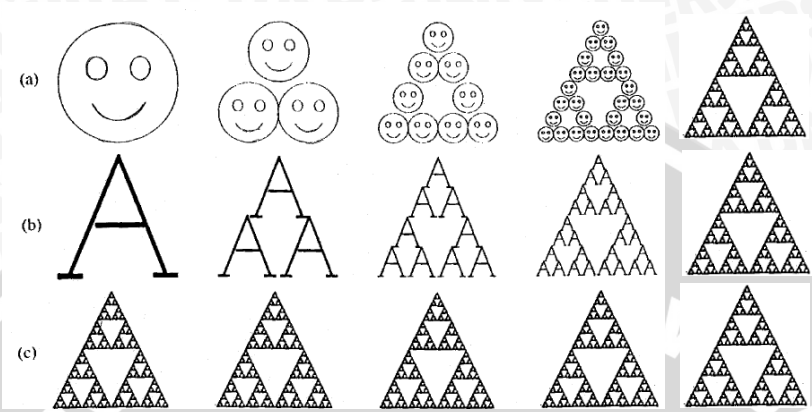
Sistem lensa pada MRCM dapat dinyatakan dengan sekumpulan transformasi *affine* $w_1,w_2,...,w_n$. Setiap transformasi w_i melakukan pencondongan, pemutaran, pengecilan dan pergeseran terhadap salinan (*copy*) citra masukan.

Menurut Rinaldi Munir (2004), setiap transformasi *affine* dinyatakan sebagai matriks dengan enam buah elemen :

$$w = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \tag{2.1}$$

Menurut Rinaldi Munir (2004), sembarang titik *(x,y)* pada gambar masukan ditransformasikan oleh *w* menjadi

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = w \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = Ax + t \tag{2.2}$$



Gambar awal Salinan 1 Salinan 2 Salinan 3 Salinan 20

Gambar 2.3 Contoh citra input dan salinannya pada MRCM

Setiap transformasi *affine* w_i akan menghasilkan salinan citra yang lebih kecil. Untuk sembarang citra awal *A* yang diinputkan ke dalam MRCM, akan dihasilkan salinan transformasi *affine* $w_1(A)$, $w_2(A)$, …, $w_n(A)$. Gabungan dari seluruh salinan tersebut adalah $W(A)$, sebagai keluaran dari MRCM,

$$W(A) = w_1(A) + w_2(A) + \ldots + w_n(A) \qquad (2.3)$$

W, adalah operator Hutchinson, yaitu gabungan (*collage*) dari sejumlah transformasi individual w_i, yaitu

$$W = w_1 \bigcup w_2 \bigcup \ldots \bigcup w_n = \bigcup_{i=1}^{n} w_i \qquad (2.4)$$

MRCM selalu menghasilkan salinan gambar yang ukurannya lebih kecil daripada ukuran gambar awal. Hal ini disebabkan karena setiap transformasi *affine* w_i memetakan dua buah titik menjadi lebih dekat. Sifat ini disebut dengan sifat kontraktif. Jika digunakan skema umpan balik terhadap citra awal A_0, maka akan diperoleh

$$A_1 = W(A_0) = \bigcup_{i=1}^{n} w_i(A_0)$$

$$A_2 = W(A_1) = W(W(A_0)) = W^2(A_0)$$

$$A_3 = W(A_2) = W(W(A_1)) = W^3(A_0)$$

$$\vdots$$

$$A_n = W^n(A_0)$$

Jika W secara seluruhnya kontraktif, maka untuk n $\rightarrow \infty$ uraiannya konvergen ke sebuah citra yang unik, A_∞. Citra A_∞ disebut titik tetap (*fixed point*) atau *invariant* dari proses penguraian, dan *attractor* dari W (Rinaldi Munir, 2004). Titik tetap adalah citra A_∞ sehingga

$$A_\infty = W(A_\infty) \qquad (2.5)$$

Jadi, jika A_∞ dipilih sebagai citra awal, maka tidak ada perubahan pada hasil transformasinya. Sedangkan *attractor* dari W adalah citra A_∞ sedemikian sehingga

$$A_\infty = \lim_{n\to\infty} W^n(A_0) \qquad (2.6)$$

Dari persamaan 2.6 dapat disimpulkan bahwa apapun citra awal yang diinputkan, limit penguraiannya selalu konvergen ke citra akhir yang sama.

Menurut Rinaldi Munir (2004), transformasi *affine* yang menghasilkan citra titik tetap segitiga Sierpinski adalah

$$w_1 = \begin{bmatrix} 0.5 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0.5 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 \end{bmatrix} \quad w_3 = \begin{bmatrix} 0.5 & 0.0 & 0.25 \\ 0.0 & 0.5 & 0.5 \end{bmatrix}$$

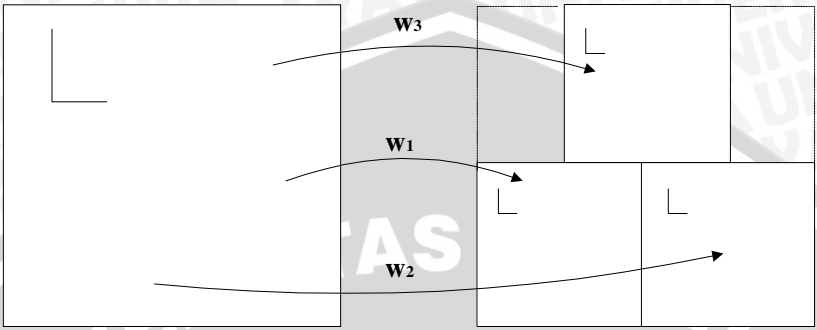
Skema pembentukan segitiga Sierpinski dengan ketiga transformasi w_1, w_2, dan w_3 ditunjukkan pada Gambar 2.4.

Jika jumlah transformasi *affine* meningkat menjadi empat dengan setiap w_i adalah sebagai berikut :

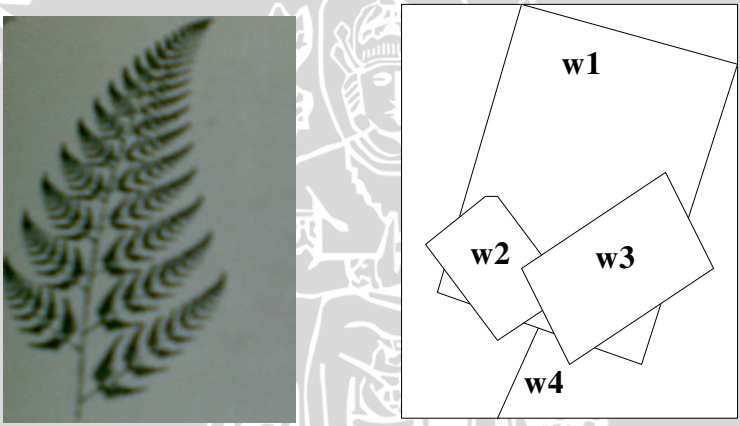
$$w_1 = \begin{bmatrix} 0.85 & 0.04 & 0.0 \\ -0.04 & 0.85 & 1.6 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0.20 & -0.26 & 0.0 \\ 0.23 & 0.22 & 1.6 \end{bmatrix}$$

$$w_3 = \begin{bmatrix} -0.15 & 0.28 & 0.0 \\ 0.26 & 0.52 & 0.44 \end{bmatrix} \quad w_4 = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.6 & 0.0 \end{bmatrix}$$

maka MRCM konvergen ke citra fraktal yang terkenal, yang dinamakan tanaman pakis Barnsley (*Barnsley's fern*). Di sini, w_1 mengendalikan keseluruhan bentuk, w_2 membangkitkan daun kiri, w_3 membangkitkan daun kanan, dan w_4 menghasilkan batang (Rinaldi Munir, 2004) seperti ditunjukkan oleh Gambar 2.5.



Gambar 2.4 Pembentukan segitiga Sierpenski oleh MRCM



Gambar 2.5 Pakis Barnsley dan empat buah transformasi *affine*-nya

2.3 Pengkodean Citra dengan IFS

Teknik kompresi yang dipakai oleh kompresi fraktal yaitu dengan menyimpan transformasi *affine*-nya, sehingga memori yang dibutuhkan untuk menyimpan citra menjadi jauh lebih sedikit. Teknik ini secara lebih spesifik dikenal dengan pengkodean citra dengan IFS (*Iterated Function System*), dengan metode ini, rasio kompresi yang dihasilkan oleh kompresi fraktal cukup tinggi.

"Salah satu contohnya yaitu Pakis Barnsley dibangkitkan dengan empat buah transformasi *affine* masing-masingnya terdiri atas enam buah bilangan riil (4 *byte*), sehingga dibutuhkan 4 x 6 x 4 = 96 *byte*

untuk menyimpan keempat transformasi itu. Apabila citra Pakis Barnsley disimpan dengan representasi *pixel* hitam putih (1 *pixel* = 1 *byte*) berukuran 550 x 480 membutuhkan memori sebesar 264.000 *byte*. Maka, rasio pemampatan citra pakis Barnsley adalah 264.000 : 96 = 2750 : 1. Hal tersebut menunujukkan bahwa pemampatan dengan fraktal mampu memberikan rasio pemampatan yang sangat tinggi" (Rinaldi Munir,2004).

Walaupun menghasilkan rasio kompresi yang cukup tinggi, namun teknik ini memiliki kesulitan yang cukup komplek. Kesulitan utamanya yaitu menemukan bagian citra yang mirip dengan keseluruhan citra. Untuk menemukan kemiripan tersebut, diperlukan keterlibatan manusia untuk menandai bagian citra yang mirip dengan keseluruhan citra. Selain itu, sangat jarang citra alami yang memiliki *self similarity*. Oleh sebab itulah pengkodean citra dengan IFS dirasa tidak mungkin untuk sembarang citra.

2.4 *Partitioned Iterated Function System* (PIFS)

Hampir semua citra alami bukan merupakan citra fraktal, sehingga citra tersebut tidak memiliki transformasi *affine* terhadap dirinya sendiri. Walaupun citra alami tidak pernah *self similar* secara keseluruhan, citra alami memiliki kemiripan lokal, yaitu terdapat bagian citra yang mirip dengan bagian lainnya dalam skala yang berbeda, misalnya citra berskala abu (*greyscale*) Lena pada Gambar 2.6 (bagian yang mirip ditandai di dalam kotak putih. Sebagai contoh bagian topi Lena mirip dengan bagian topi di dalam bayangan cermin, bagian bahu mirip dengan bagian bahu yang lebih besar, dan sebagainya).

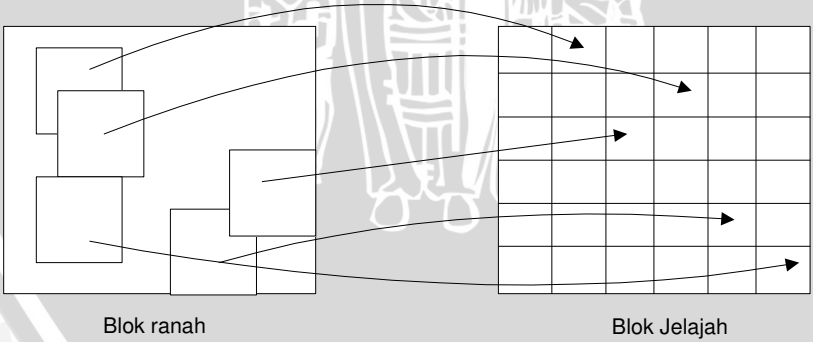
Kemiripan lokal yang banyak terdapat pada citra alami bersifat *self-transformability*, yaitu bagian citra yang lebih kecil dapat diperoleh dengan mentransformasikan bagian citra yang lebih besar namun mirip dengan bagian citra yang lebih kecil itu (Rinaldi Munir, 2004). Setiap transformasi itu disebut IFS lokal (LIFS). Gabungan keseluruhan transformasi tersebut merupakan sebuah citra yang menyerupai atau hampir mirip dengan citra awal.

Kompresi citra fraktal dilakukan dengan membagi sebuah citra menjadi sejumlah blok dengan ukuran tertentu yang sama dan tidak saling beririsan, disebut sebagai blok jelajah (*range block*). Kemudian untuk citra yang sama, akan dibagi menjadi sejumlah blok dengan ukuran lebih besar daripada blok jelajah dan saling beririsan

yang disebut sebagai blok ranah (*domain block*). Untuk lebih menyederhanakan permasalahan, maka blok jelajah dan blok ranah diambil berbentuk bujur sangkar. Untuk setiap blok jelajah, dicari blok ranah yang paling mirip dengan blok jelajah tersebut. Ilustrasinya ditunjukkan seperti Gambar 2.7.

Gambar 2.6 Kemiripan lokal pada citra Lena

Setelah menemukan blok jelajah dengan blok ranah yang cocok, kemudian diturunkan transformasi *affine* (IFS lokal) w_i yang memetakan blok ranah ke blok jelajah. Hasil dari semua pemasangan ini adalah *Partitioned Iterated Function System* (PIFS) (Rinaldi Munir, 2004).



Gambar 2.7 Pemetaan dari blok ranah ke blok jelajah

Kemiripan antara dua buah (blok) citra diukur dengan metrik jarak. Metrik jarak yang banyak digunakan dalam praktek adalah metrik rms (*root meas square)* :

$$d_{rms} = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(z'_{ij} - z_{ij})^2} \qquad (2.7)$$

dengan z dan z' adalah nilai intensitas *pixel* dari dua citra, dan n adalah jumlah *pixel* di dalam citra (Rinaldi Munir, 2004).

Untuk lebih menyederhanakan persoalan, ukuran blok ranah diambil dua kali ukuran blok jelajah. Misalnya dalam citra berukuran 400x600, jika diambil ukuran blok jelajah 2x2 dan blok ranah 4x4, maka akan terdapat 60000 blok jelajah dan 237009 buah blok ranah. Himpunan blok ranah akan dimasukkan ke dalam pul ranah (*domain pool*). Semakin besar ukuran pul ranah, maka kualitas pemampatan citra yang dihasilkan akan semakin baik, namun akan memerlukan waktu yang lama untuk proses pencarian antara blok jelajah dan blok ranah.

Ukuran blok ranah dan blok jelajah yang berbeda menyebabkan perhitungan kemiripan dengan matrik d_{rms} pada persamaan 2.7 sulit dilakukan. Namun hal ini bisa diatasi dengan melakukan penyekalaan terhadap blok ranah. Menurut Rinaldi Munir (2004), penyekalaan dilakukan dengan menjadikan 2x2 buah *pixel* menjadi satu buah *pixel*. Nilai satu buah *pixel* tersebut adalah rata-rata nilai keempat *pixel*.

Transformasi *affine* w_i untuk citra berskala abu-abu disusun oleh bagian spasial yang memetakan posisi *pixel* di blok ranah D_i ke posisi *pixel* di blok jelajah R_i, dan bagian intensitas *pixel*. Titik (x,y) dengan intensitas z yang termasuk di dalam blok ranah dipetakan oleh w_i menjadi :

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \qquad (2.8)$$

(Rinaldi Munir, 2004).

Dengan pemetaan w_i di atas, intensitas tiap *pixel* juga diskalakan dan digeser, yaitu

$$z' = s_i z + o_i \qquad (2.9)$$

”Parameter e_i dan f_i menyatakan pergeseran sudut kiri blok ranah ke sudut kiri blok jelajah yang bersesuaian. Parameter s_i menyatakan faktor kontras *pixel* (seperti tombol kontras di TV). Bila s_i bernilai 0, maka *pixel* menjadi gelap, bila s_i sama dengan 1 kontras tidak berubah; antara 0 dan 1 kontras *pixel* berkurang, di atas 1 kontras bertambah. Parameter o_i menyatakan ofset kecerahan (*brightness) pixel* (seperti tombol kecerahan di TV. Nilai o_i positif mencerahkan gambar dan nilai o_i negatif menjadikan gambar gelap. Kedua parameter tersebut dapat memetakan secara akurat blok ranah berskala abu ke blok jelajah berskala abu” (Rinaldi Munir, 2004).

Karena perbandingan ukuran antara blok jelajah dengan blok ranah adalah 1:2, maka transformasi *affine* pada persamaan 2.8 akan menjadi lebih sederhana, yaitu

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \tag{2.10}$$

Parameter s_i dan o_i dihitung dengan menggunakan rumus regresi berikut : Diberikan dua buah blok citra yang mengandung n buah *pixel* dengan intensitas d_i, d_2, .., d_n (dari blok ranah D_i) dan r_1, r_2, …, r_n (dari R_i). Nilai s dan o diperoleh dengan meminimumkan total kuadrat selisih antara intensitas pixel blok D_i yang diskalakan dengan s lalu digeser sejauh o dan intensitas *pixel* blok R_i :

$$E = \sum_{i=1}^{n} (d'_i - r_i)^2 = \sum_{i=1}^{n} (s.d_i + o - r_i)^2 \tag{2.11}$$

Nilai minimum E terjadi bila turunan parsialnya terhadap s dan o adalah nol, yang terjadi bila turunan pertama E sama dengan 0, atau

$E' = 0$

yang dipenuhi oleh

$$s = \frac{\left[n\sum_{i=1}^{n} d_i r_i - \sum_{i=1}^{n} d_i \sum_{i=1}^{n} r_i \right]}{\left[n\sum_{i=1}^{n} d_i^2 - \left(\sum_{i=1}^{n} d_i \right)^2 \right]} \tag{2.12}$$

dan

$$o = \frac{1}{n}\left[\sum_{i=1}^{n} r_i - s\sum_{i=1}^{n} d_i\right] \qquad (2.13)$$

Jika $n\sum_{i=1}^{n} d_i^2 - (\sum_{i=1}^{n} d_i)^2 = 0$ maka $s = 0$ dan $o = \frac{1}{n}\sum_{i=1}^{n} r_i$.

Dengan nilai *s* dan *o* yang telah diperoleh, maka kuadrat galat antara blok jelajah dan blok ranah adalah

$$E = \frac{1}{n}\left[\sum_{i=1}^{n} r_i^2 + s\left(s\sum_{i=1}^{n} d_i^2 - 2\sum_{i=1}^{n} d_i r_i + 2o\sum_{i=1}^{n} d_i\right) + o\left(no - 2\sum_{i=1}^{n} r_i\right)\right]$$

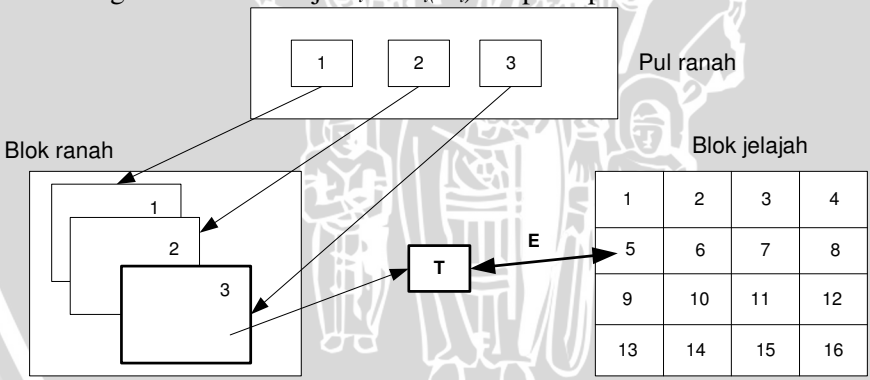
(2.14)

Sedangkan matrik *rms,* d_{rms} adalah

$$d_{rms} = \sqrt{E}/n \qquad (2.15)$$

(Rinaldi Munir, 2004)

Selanjutnya, transformasi *affine* w_i diuji terhadap blok ranah D_i untuk menghasilkan blok uji $T_i = w_i(D_i)$ seperti pada Gambar 2.8



Gambar 2.8 Blok jelajah 5 dibandingkan dengan blok ranah 3 di dalam pul ranah. Transdformasi *w* ditentukan, lalu blok ranah 3 ditransformasikan dengan *w* menghasilkan *T*. Jarak antara *T* dengan blok jelajah 5 diukur

Jarak antara T dan R_i dihitung dengan persamaan 2.15. Transformasi *affine* yang terbaik ialah transformasi yang meminimalkan jarak antara R_i dan T (Rinaldi Munir, 2004).

Pencarian kemiripan dilakukan untuk setiap blok jelajah. Hasil dari proses kompresi ini adalah sejumlah IFS lokal yang disebut PIFS. Parameter PIFS yang akan disimpan ke dalam sebuah berkas eksternal yaitu e_i, f_i, s_i, o_i , dan jenis operasi simetri untuk setiap blok jelajah. Biasanya parameter e_i dan f_i diganti dengan posisi blok ranah yang dipetakan ke blok jelajah.

2.5 Rekonstruksi Citra

Rekonstruksi (*decoding*) citra dilakukan dengan menguraikan PIFS dari citra awal sembarang. Karena setiap IFS lokal(LIFS) kontraktif, baik kontraktif dalam matra intensitas maupun kontraktif dalam matra spasial maka uraiannya akan konvergen ke citra semula, sedangkan kontraktif spasial berguna untuk membuat rincian pada citra untuk setiap skala. Jika PIFS yang ditemukan selama proses pemampatan bagus, yaitu gabungan dari transformasi seluruh blok ranah dekat dengan citra semula (diukur dengan persamaan 2.7), maka titik tetap PIFS juga dekat dengan citra semula tersebut (Rinaldi Munir, 2004).

Ketika proses dekompresi,setiap IFS lokal mentransformasikan sekumpulan blok ranah menjadi sekumpulan blok jelajah. Citra yang akan dihasilkan dari proses dekompresi ini akan berupa titik tetap yang menyerupai citra semula, karena blok jelajah tidak saling beririsan dan mencakup seluruh *pixel*.

Konvergensi ke citra titik tetap berlangsung cepat. Konvergensi umumnya dapat diperoleh dalam 8 sampai 10 kali penguraian. Karena transformasi *affine* kontraktif dalam arah spasial, maka semakin banyak rincian citra yang dibuat pada setiap penguraian (Rinaldi Munir, 2004).

Salah satu keunggulan citra hasil dekompresi fraktal yaitu rasio kompresi yang tinggi serta citra hasil yang menghampiri citra asli. Menurut *Anonymous* (1997), citra hasil dekompresi tidak memiliki ukuran yang asli, citra tersebut dapat didekompresi dalam ukuran berapapun. Hasil yang sangat detil untuk ukuran dekompresi yang sangat besar dibangkitkan secara otomatis oleh transformasi yang dihasilkan pada saat kompresi.

2.6 Algoritma Genetika

Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis (Sri Kusumadewi & Hari Purnomo, 2005). Menurut Wahyu Nurjaya algoritma genetika sangat tepat digunakan untuk penyelesaian masalah optimasi yang kompleks dan sukar diselesaikan dengan menggunakan metode yang konvensional.

Algoritma ini dimulai dengan sejumlah solusi yang mungkin. Sejumlah solusi yang mungkin ini disebut sebagai populasi. Masing-masing solusi yang mungkin di dalam sebuah populasi disebut dengan kromosom (*Chromosome)*. *Chromosome* tersusun atas gen-gen yang merupakan representasi nilai variabel dalam suatu fungsi yang ingin dicari solusinya. *Chromosome* dapat dibentuk dari bilangan numerik, biner ataupun karakter (Denny Hermawanto, 2006). Setiap kromosom diberi sebuah nilai *fitness* berdasarkan pada fungsi *fitness*. Solusi dari sebuah populasi diambil dan dipergunakan untuk membentuk populasi yang baru. Hasil yang diharapkan adalah populasi baru yang terbentuk akan memiliki nilai *fitness* yang lebih tinggi dibandingkan dengan populasi sebelumnya. Suatu solusi bisa terpilih untuk membentuk solusi yang baru (*offspring*) tergantung pada nilai *fitness*-nya, semakin besar nilai *fitness* suatu solusi, maka semakin besar pula peluang terpilihnya solusi tersebut untuk bereproduksi. *Offspring* akan menggantikan populasi lama. Proses ini diulangi sampai kriteria yang diinginkan ditemukan. Misalnya, reproduksi akan berakhir apabila generasi sudah mencapai 200.

Menurut Obitko (1998), berikut ini adalah garis besar algoritma genetika secara dasar :

1. [Mulai] : Membangkitkan populasi acak (random) sejumlah n kromosom (solusi yang cocok dengan permasalahan).
2. [Fitness] : Mengevaluasi *fitness f(x)* dari setiap kromosom *x* di dalam populasi
3. [Populasi baru] : Membuat populasi baru dengan mengulangi langkah-langkah berikut sampai populasi baru lengkap
   3.1. [Seleksi] : Memilih dua kromosom induk dari sebuah populasi sesuai dengan *fitness*-nya (*fitness* terbaik akan memiliki peluang terbesar untuk terpilih)
   3.2. [Perkawinan silang] : Perkawinan silang antara dua induk bertujuan untuk membentuk *offspring* (anak) baru. Perkawinan silang terjadi dengan probabilitas tertentu. Jika

tidak terjadi perkawinan silang, maka *offspring* yang terbentuk adalah salinan dari induknya.

3.3. [Mutasi] : Mengubah beberapa atribut dari *offspring* baru di dalam lokus (posisi di dalam kromosom) dengan probabilitas tertentu. Jika tidak terjadi mutasi, maka *offspring* merupakan hasil secara langsung dari perkawinan silang, atau salinan dari salah satu induk.

3.4. [Penerimaan] : Menempatkan *offspring* baru ke dalam populasi baru.

4. [Ganti] : Menggunakan populasi yang baru dibentuk untuk proses selanjutnya
5. [Tes] : Jika kondisi terakhir memenuhi, berhenti dan hasilnya adalah solusi terbaik dari populasi saat itu.
6. [Ulangi] : Kembali ke langkah 2

2.7 Pengkodean Algoritma Genetika

Langkah pertama di dalam algoritma genetika adalah merepresentasikan permasalahan ke dalam hubungan biologi. Format dari kromosom disebut sebagai pengkodean atau representasi. Menurut Obitko (1998), ada empat cara yang secara umum digunakan yaitu :

2.7.1. Pengkodean Biner

Pengkodean biner adalah pengkodean yang paling sederhana dan umum. Di dalam pengkodean biner, setiap kromosom merupakan barisan string bit, 0 atau 1. Contoh pengkodean biner ditunjukkan oleh Gambar 2.9.

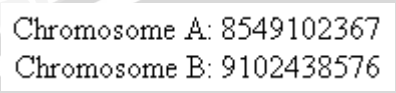
Chromosome A: 0101101100010011
Chromosome B: 1011010110110101

Gambar 2.9. Contoh representasi kromosom dengan pengkodean biner

2.7.2. Pengkodean Permutasi

Pengkodean permutasi dapat digunakan di dalam permasalahan pengurutan, seperti misalnya *Traveling Salesman Problem* (TSP) atau permasalahan pengurutan tugas. Di dalam pengkodean

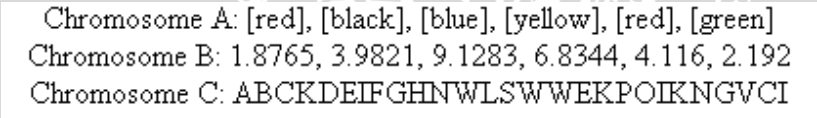
permutasi, setiap kromosom adalah sebuah angka, yang mana merepresentasikan angka pada urutan. Sebagai contoh pengkodean permutasi dapat dilihat pada Gambar 2.10.



Gambar 2.10. Contoh representasi kromosom dengan pengkodean permutasi

2.7.3. Pengkodean Nilai Langsung (*value)*

Pengkodean nilai langsung bisa digunakan di dalam permasalahan dengan nilai yang rumit seperti misalnya menggunakan angka riil (desimal). Penggunaan pengkodean biner untuk permasalahan seperti ini akan sangat sulit. Di dalam pengkodean nilai, setiap kromosom merupakan nilai tertentu. Nilai dapat berupa apa saja seperti misalnya angka biasa, angka riil atau karakter dari beberapa objek yang rumit. Contoh pengkodean nilai langsung dapat dilihat pada Gambar 2.11.



Gambar 2.11. Contoh representasi kromosom dengan pengkodean nilai langsung

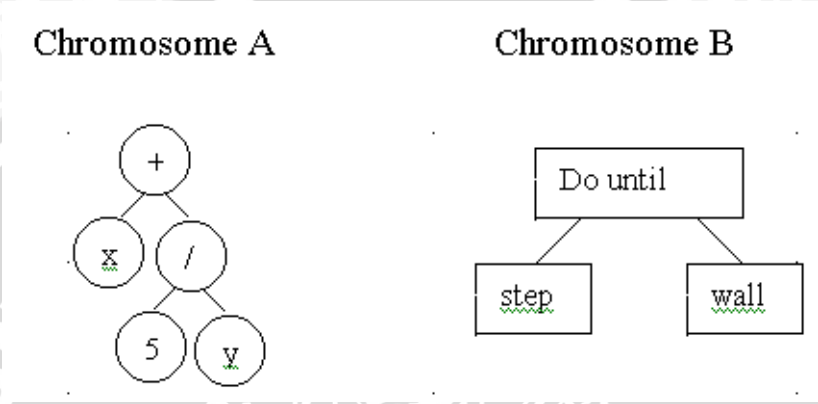
2.7.4. Pengkodean Pohon (*tree*)

Pengkodean pohon biasanya digunakan untuk menyusun program atau ekspresi, untuk pemrograman genetika. Dalam pengkodean pohon, setiap kromosom merupakan pohon dari beberapa objek, seperti fungsi atau perintah dalam bahasa pemrograman. Untuk contoh pengkodean pohon, dapat dilihat pada Gambar 2.12.

2.8. Seleksi

Untuk menghasilkan *offspring*, maka harus dilakukan perkawinan silang. Sebelum perkawinan silang dilakukan, maka terlebih dahulu dilakukan pemilihan induk yang disebut dengan seleksi. Seleksi bertujuan untuk memberikan kesempatan reproduksi yang lebih

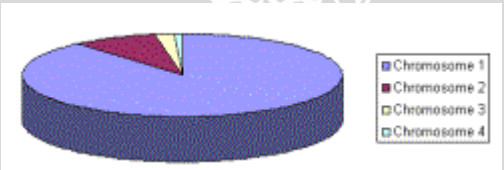
besar kepada invidu di dalam populasi yang paling kuat (Son Kuswadi, 2007). Menurut Obitko (1998), berikut ini adalah metode seleksi yang paling umum digunakan.



Gambar 2.12. Contoh representasi kromosom dengan pengkodean pohon

2.8.1. Seleksi *Roulette Wheel*

Tolak ukur yang digunakan dalam metode seleksi ini adalah nilai *fitness*, kromosom dengan nilai *fitness* tertinggi akan memiliki peluang yang paling tinggi untuk dipilih. Konsep ini menggunakan konsep roda *roulette*, di mana setiap kromosom akan menempati bagian tertentu yang mana besarnya bagian yang ditempati akan disesuaikan dengan besarnya nilai *fitness*. Kromosom dengan nilai *fitness* tertinggi akan menempati bagian yang paling luas, demikian juga sebaliknya. Kemudian sebuah kelereng dilemparkan ke roda *roulette* dan kromosom dipilih sesuai tempat jatuhnya kelereng. Contoh penempatan kromosom pada *roulette* ditunjukkan oleh Gambar 2.13



Gambar 2.13. Penempatan kromosom pada *roulette*

Dari gambar 2.13 terlihat bahwa kromosom 1 menempati area paling luas, hal itu menunjukkan bahwa kromosom 1 memiliki *fitness* yang paling besar, sedangkan kromosom 4 menempati area yang paling kecil karena *fitness*-nya paling rendah.

Menurut Obitko (1998), *pseudo code* berikut ini adalah salah satu cara untuk mensimulasikan seleksi *roulette wheel*

1. [Jumlah] : menghitung jumlah semua *fitness* kromosom dalam populasi (misalnya jumlahnya adalah S)
2. [Pilih] : Membangkitkan angka acak dari interval (0,S) (misalnya angka acak yang dihasilkan adalah r
3. [Ulangi] : Periksa kromosom dalam populasi dan jumlahkan *fitness* dari 0 (misalnya s), jika s lebih besar daripada r, maka hentikan dan hasilnya adalah kromosom yang dicek saat itu.

2.8.3. Seleksi Rangking

Pada seleksi rangking, populasi diurutkan berdasarkan objektifnya. Nilai *fitness* dari tiap-tiap individu hanya tergantung pada posisi individu tersebut dalam urutan, dan tidak dipengaruhi oleh nilai objektifnya (Sri Kusumadewi & Hari Purnomo, 2005). Perbandingan seleksi *roulette wheel* dengan seleksi rangking ditunjukkan oleh Gambar 2.14

Gambar 2.14. Perbandingan antara seleksi *roulette wheel* dengan seleksi rangking

2.9. Perkawinan Silang dan Mutasi

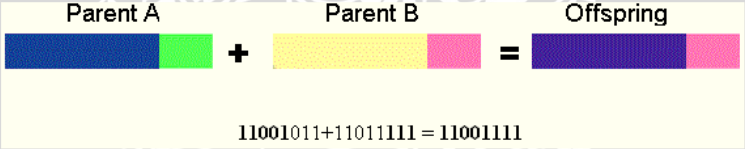
Perkawinan silang dan mutasi adalah operator dasar algoritma genetika. Kinerja algoritma genetika sangat tergantung pada kedua operator tersebut. Perkawinan silang akan menghasilkan *offspring* baru, sedangkan mutasi dilakukan untuk mnghindari terjadinya konvergensi dini. Baik perkawinan silang dan mutasi dilakukan dengan peluang tertentu yang masing-masing disebut dengan probabilitas perkawinan silang dan probabilitas mutasi. Mutasi dapat mencegah terjadinya konvergensi dini, karena dengan adanya mutasi

akan lebih banyak kemungkinan dievaluasi. Walaupun demikian probabilitas mutasi juga tidak boleh terlalu tinggi karena anak yang dihasilkan menjadi kehilangan kemiripan dengan induknya sehingga menghancurkan pencarian daerah solusi (Sri Kusumadewi & Hari Purnomo, 2005).Tipe dan implementasinya tergantung pada pengkodean kromosom dan juga permasalahannya. Menurut Obitko (1998), teknik perkawinan silang dan mutasi yang bisa dilakukan adalah sebagai berikut

2.9.1. Pengkodean Biner

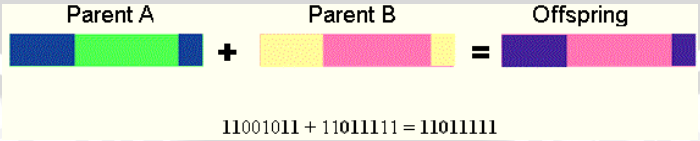
Ada banyak metode perkawinan silang untuk pengkodean biner, diantaranya adalah sebagai berikut :

a. Titik potong tunggal
Dengan metode ini, awal dari kromosom sampai titik potong akan disalin dari induk pertama, sisanya diambil dari induk kedua. Contoh perkawinan silang untuk pengkodean biner dengan titik potong tunggal ditunjukkan oleh Gambar 2.15.



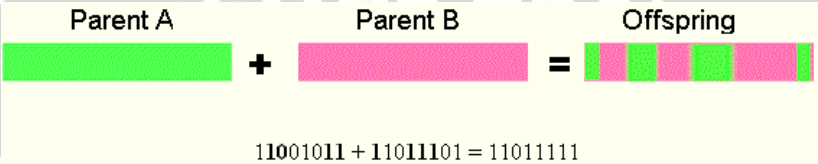
Gambar 2.15 Contoh perkawinan silang untuk pengkodean biner dengan satu titik potong

b. Titik potong ganda
Dengan metode ini, string biner dari awal kromosom sampai titik potong pertama disalin dari induk kemudian bagian pertama setelah titik potong pertama sampai titik potong kedua disalin dari induk kedua dan bagian akhir disalin dari induk pertama lagi. Contoh perkawinan silang untuk pengkodean biner dengan 2 titik potong ditunjukkan oleh Gambar 2.16.



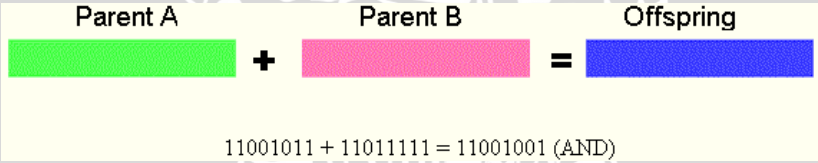
Gambar 2.16 Contoh perkawinan silang untuk pengkodean biner dengan dua titik potong

c. Perkawianan silang tidak seragam
Dengan metode ini, bit disalin secara acak dari induk pertama atau induk kedua. Contoh perkawinan silang tidak seragam untuk pengkodean biner ditunjukkan oleh Gambar 2.17.



Gambar 2.17 Contoh perkawinan silang tidak seragam untuk pengkodean biner

d. Perkawinan silang aritmatika
Dalam metode ini, beberapa operasi aritmatik dilibatkan untuk membuat *offspring* yang baru. Contoh perkawinan silang aritmatika untuk pengkodean biner ditunjukkan oleh Gambar 2.18.



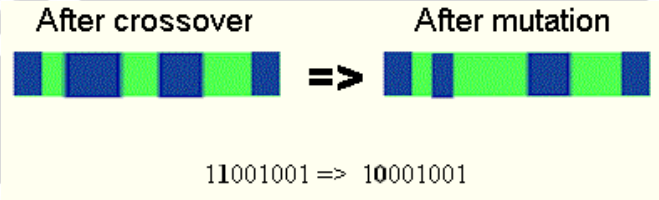
Gambar 2.18 Contoh perkawinan silang aritmatik untuk pengkodean biner

Sedangkan untuk metode mutasi pada pengkodean biner yang paling banyak digunakan adalah inversi bit (*bit inversion*), setiap bit yang terpilih akan dibalik. Contoh perkawinan mutasi inversi pada pengkodean biner ditunjukkan oleh Gambar 2.19.

2.9.2. Pengkodean Permutasi

Ada banyak metode perkawinan silang untuk pengkodean permutasi, namun yang paling umum digunakan adalah perkawinan

silang satu titik potong, di mana permutasi disalin dari induk pertama sampai titik potong, kemudian induk kedua ditinjau, jika ada angka yang belum ada di *offspring*, maka akan ditambahkan pada *offspring*. Contoh perkawinan silang pada pengkodean permutasi ditunjukkan oleh Gambar 2.20.



Gambar 2.19 Contoh mutasi inversi pada pengkodean biner

(**1 2 3 4 5** 6 7 8 9) + (**4 5 3 6 8** 9 7 2 1) = (**1 2 3 4 5** 6 8 9 7)

Gambar 2.20 Contoh perkawinan silang pada pengkodean permutasi

Sedangkan metode mutasi yang biasanya digunakan adalah penukaran urutan (*order changing*) yang mana dua angka dipilih dan kemudian dan ditukar. Contoh mutasi pada pengkodean permutasi ditunjukkan oleh Gambar 2.21.

(1 **2** 3 4 5 6 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)

Gambar 2.21 Contoh mutasi pada pengkodean permutasi

2.9.3. Pengkodean Nilai Langsung

Semua metode perkawinan silang untuk pengkodean biner dapat digunakan untuk pengkodean nilai langsung.

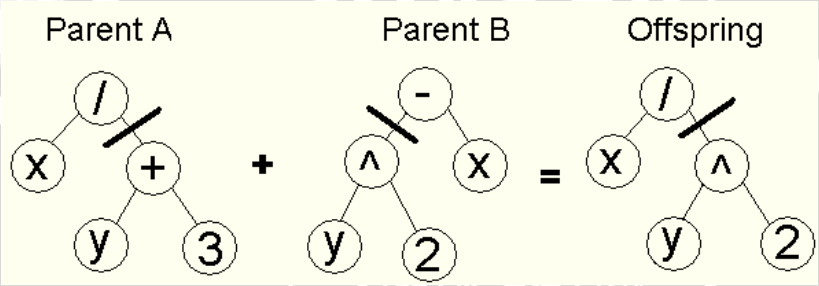
Mutasi pada pengkodean nilai langsung dilakukan dengan metode penambahan (*adding*), di mana sebuah angka kecil (untuk pengkodean bilangan riil) ditambahkan (atau dikurangi) dari angka yang terpilih. Contoh mutasi pada pengkodean nilai langsung ditunjukkan oleh Gambar 2.22.

(1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

Gambar 2.22 Contoh mutasi pada pengkodean nilai langsung

2.9.4. Pengkodean Pohon

Metode perkawinan silang pada pengkodean pohon yaitu satu titik dipilih dari kedua induk, induk dibagi dengan titik tersebut dan bagian di bawah titik potong ditukar untuk menghasilkan *offspring* baru. Contoh perkawinan silang pada pengkodean pohon ditunjukkan oleh Gambar 2.23.



Gambar 2.23 Contoh perkawinan silang pada pengkodean pohon

Mutasi pada pengkodean pohon dilakukan dengan memilih *node* kemudian diganti.

2.10. *Elitism*

Elitism dimaksudkan untuk menjaga kromosom terkuat (memiliki *fitness* tertinggi) di tiap generasi. *Elitism* dapat meningkatkan kinerja algoritma genetik secara tajam.

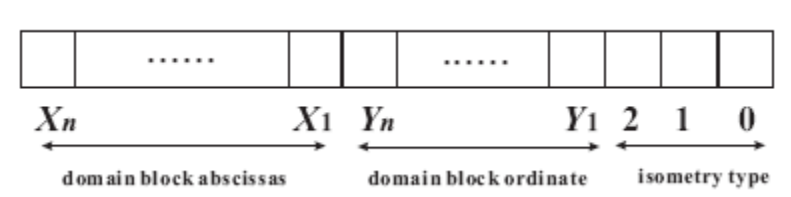
2.11. Kompresi Citra Fraktal dengan Algoritma Genetika

Diantara teknik kompresi citra, kompresi citra berdasarkan pada teori sistem fungsi iterasi lokal (LIFS) telah diterima dengan perhatian yang besar. Dasar teknik ini dikenal dengan permaslahan invers fraktal, yaitu bagaimana menemukan LIFS yang ada di dalam citra yang diberikan. Satu keterbatasan metode ini adalah banyaknya waktu yang dibutuhkan untuk mengompres citra (Lucis Vences dan Isaac Rudomin, 1997).

Untuk mengoptimasi waktu pencocokan blok ranah dengan blok jelajah dapat digunakan algoritma genetika untuk membangkitkan

populasi IFS lokal (LIFS) secara acak, populasi awal inilah yang akan dipakai untuk proses algoritma genetika selanjutnya.

Parameter LIFS tersusun atas $\{x,y,z,a,g\}$, di mana $\{x,y\}$ merupakan lokasi dari blok ranah yang paling cocok, $\{z\}$ merupakan tipe isometri dari domain blok yang paling cocok, $\{a,g\}$ merupakan kontras dan *offset brightness* (kecerahan). Kromosom direpresentasikan dengan pengkodean keabuan yaitu $\{x_1, x_2, ...x_n, y_1, y_2, ...y_n, z_1, z_2, z_3\}$ (Lifeng Xi & Liangbin Zhang, 2007). Representasi kromosom untuk pencarian domain blok ditunjukkan oleh Gambar 2.24.



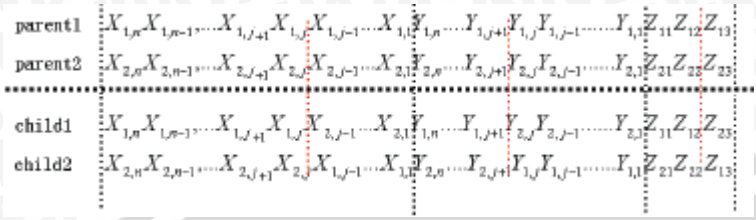
Gambar 2.24 Representasi kromosom pencarian domain blok

Fungsi *fitness* yang dipilih harus merefleksikan sifat dari anggota populasi yang diinginkan. Dalam kasus kompresi citra fraktal ini, fungsi *fitness* berdasarkan pada jarak antara blok ranah dengan blok jelajah yaitu dengan metrik *rms* (persamaan 2.7). Berikut ini adalah fungsi *fitness* yang digunakan :

$$fitness = 1/(1+\sum_{i=1}^{n} drms_i) \qquad (2.16)$$

di mana n adalah jumlah *pixel* dalam blok jelajah, $drms_i$ adalah matrik *rms* blok jelajah ke-i.

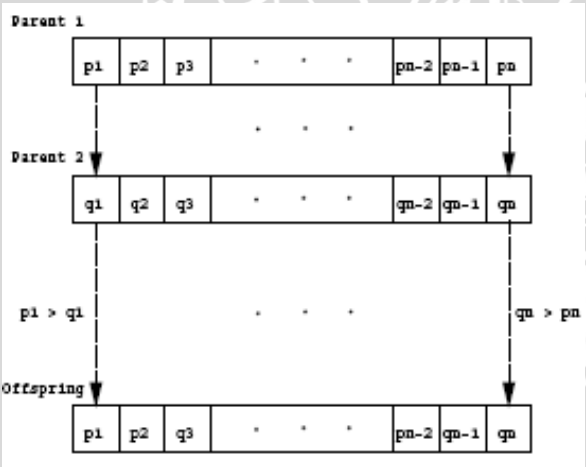
Proses seleksi menggunakan seleksi *roulette wheel* di mana kromosom dengan *fitness* tertinggi akan memiliki peluang terbesar. Setelah 2 kromosom terpilih sebagai induk, maka proses perkawinan silang bisa dilakukan. Karena kromosom terdiri dari tiga bagian, maka titik potong diletakkan secara acak di tiap bagian, kemudian perkawinan silang dilakukan (Lifeng Xi & Liangbin Zhang, 2007). Contoh perkawinan silang pada kompresi citra fraktal ditunjukkan oleh Gambar 2.25.



Gambar 2.25 Contoh perkawinan silang pada kompresi citra fraktal

Menurut Lucis Vences dan Isaac Rudomin (1997), operasi perkawinan silang bisa diperbaiki dengan membentuk anak dari dua induk, di mana tiap gen di dalam induk dibandingkan nilai *fitness*-nya. Gen dengan nilai *fitness* tertinggi dari kedua induk akan membentuk gen anak. Contoh metode perbaikan perkawinan silang ditunjukkan oleh Gambar 2.26

Mutasi melakukan pertukaran gen pada kromosom dengan probabilitas mutasi tertentu, biasanya antara 0 sampai 1. Mutasi bertujuan untuk memperbaiki materi genetik yang tidak pernah diperiksa sebelumnya dalam populasi agar tidak terjadi konvergensi dini.



Gambar 2.26 Perbaikan metode perkawinan silang pada kompresi citra fraktal

Menurut Lucis Vences dan Isaac Rudomin (1997), berikut ini adalah algoritma genetik untuk kompresi citra fraktal :

1. $P \leftarrow$ bangkitkan LIFS secara random

2. **for all** LIFS $p_i \varepsilon P$, evaluasi dan ukur *fitness*
3. **while**kriteria akhir belum terpenuhi
4. **do** reproduksi $p_i \varepsilon P$ berdasarkan pada evaluasi
5. terapkan mutasi untuk beberapa $p_i \varepsilon P$, kemudian buatlah LIFS yang baru
6. terapkan perkawinan untuk beberapa $p_i, p_j \varepsilon P$, kemudian buatlah LIFS yang baru
7. evaluasi LIFS yang baru
8. ganti string lama yang paling buruk dengan string baru terbaik.

BAB III
METODOLOGI DAN PERANCANGAN

3.1 Gambaran Umum Sistem

Metode kompresi citra fraktal memanfaatkan kemiripan lokal antara bagian-bagian citra dengan bagian yang lainnya. Metode ini merupakan metode kompresi *lossy* yang mampu menghasilkan perbandingan pemampatan yang tinggi. Kompresi fraktal membagi citra menjadi sejumlah blok berukuran sama dan tidak saling beririsan disebut blok jelajah. Kemudian membagi citra yang sama menjadi sejumlah blok yang saling beririsan dengan ukuran dua kali blok jelajah yang disebut blok ranah. Untuk setiap blok jelajah dihitung kemiripannya dengan setiap blok ranah. Kemudian diturunkan transformasi *affine*-nya. Transformasi yang meminimumkan jarak antara blok jelajah dengan blok ranah adalah transformasi terbaik. Parameter-parameter transformasi yang perlu disimpan adalah koordinat ujung kiri atas blok ranah, faktor kontras, dan *offset brightnes.*

3.1.1 Pengkodean (Representasi) Kromosom

Representasi kromosom untuk kompresi citra fraktal ini terdiri dari 2 bagian yaitu bagian absis (*e*) dan bagian ordinat (*f)* ujung kiri atas blok ranah yang dipetakan ke blok jelajah tertentu. Nilai *s* dan *o* tidak dimasukkan ke dalam kromosom karena nilai *s* dan *o* bisa dihitung dengan menggunakan nilai koordinat ujung kiri atas dan intensitas masing-masing blok ranah dan blok jelajah. Panjang masing-masing bagian dalam kromosom (gen) dalam suatu individu adalah sejumlah blok jelajah. Kromosom dipecah menjadi dua bagian yaitu bagian absis sepanjang jumlah blok jelajah, kemudian diikuti oleh bagian ordinat sepanjang blok jelajah dengan tujuan agar semakin banyak kemungkinan dievaluasi (nilai koordinat yang diperoleh lebih beragam), sehingga akan menghindarkan terjadinya konvergensi dini dan diharapkan nilai *fitness* yang diperoleh akan lebih baik.

Representasi kromosom menggunakan model pengkodean nilai, karena model ini dirasa paling cocok di mana nilai-nilai absis dan ordinat bisa langsung dimasukkan. Jika dilakukan dengan pengkodean biner maka nilai-nilai tersebut harus diubah ke dalam

bit-bit biner terlebih dahulu. Hal ini selain memerlukan lebih banyak waktu juga akan menambah panjangnya kromosom. Proses pembangkitan suatu kromosom adalah dengan membangkitkan secara berturut-turut absis dan ordinat secara acak sejumlah blok jelajah. Nilai absis adalah antara nol sampai dengan ukuran lebar citra dikurangi dengan ukuran blok ranah, sedangkan nilai ordinat yang dibangkitkan adalah antara nol sampai dengan ukuran tinggi citra dikurangi dengan ukuran blok ranah. Misalnya diberikan citra berukuran 6x6 seperti yang ditunjukkan oleh Gambar 3.1.

	0	1	2	3	4	5	6
1	25	30	55	34	88	151	
2	35	40	33	70	97	100	
3	24	15	53	48	65	56	
4	34	55	57	78	79	134	
5	56	72	49	80	56	110	
6	55	76	49	25	63	97	

Gambar 3.1 Nilai intensitas pada sebuah citra berukuran 6x6

Misalnya ukuran blok jelajah adalah 2x2, salah satu represntasi kromosom yang mungkin adalah : 1 2 0 0 2 2 1 0 1 0 1 1 1 0 2 2 1 0. Bagian kromosom yang berwarna merah menunjukkan bagian absis, sedangkan bagian kromosom yang berwarna biru menunjukkan bagian ordinat.

3.1.2 Inisialisasi Populasi

Algoritma genetik akan membangkitkan populasi awal secara acak. Ilustrasi inisialisasi kromosom ditunjukkan oleh Tabel 3.1

Tabel 3.1 Ilustrasi inisialisasi kromosom

Blok jelajah	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Individu 1	1	2	0	0	2	2	1	0	1	0	1	1	1	0	2	2	1	0
Individu 2	2	0	2	1	1	0	0	1	0	1	2	2	1	1	0	0	2	0
Individu 3	1	0	2	1	0	2	2	1	2	0	1	0	1	1	0	2	1	0

Dari Tabel 3.1 ditunjukkan bahwa jumlah blok jelajah yaitu sebanyak sembilan buah. Kemudian masing-masing individu berisi sejumlah absis (ditunjukkan oleh bagian berwarna merah) dan ordinat(ditunjukkan oleh bagian berwarna biru) ujung kiri atas blok ranah. Berdasarkan inisialisasi pada Tabel 3.1, blok ranah dengan blok jelajah yang bersesuaian ditunjukkan pada Tabel 3.2

Tabel 3.2 Blok ranah dengan blok jelajah yang bersesuaian

Individu / Blok jelajah	1	2	3
1	(1,0)	(2,1)	(1,0)
2	(2,1)	(0,2)	(0,1)
3	(0,1)	(2,2)	(2,0)
4	(0,1)	(1,1)	(1,1)
5	(2,0)	(1,1)	(0,1)
6	(2,2)	(0,0)	(2,0)
7	(1,2)	(0,0)	(2,2)
8	(0,1)	(1,2)	(1,1)
9	(1,0)	(0,0)	(2,0)

3.1.3 Perhitungan *fitness*

Setelah populasi awal terbentuk, maka dilakukan perhitungan nilai *fitness* masing-masing individu. Perhitungan nilai *fitness* ini didasarkan pada sebuah fungsi *fitness* yaitu ukuran kedekatan antara blok jelajah dengan blok ranah yang bersesuaian. Untuk menghitung ukuran kedekatan digunakan metrik *rms, drms*.

Suatu individu dikatakan terkuat jika memiliki nilai *fitness* paling tinggi. Sesuai dengan dengan fungsi di atas, maka individu yang dapat meminimalkan jarak antara masing-masing blok jelajah dan blok ranah akan memiliki nilai *fitness* tertinggi.

3.1.4 Seleksi

Seleksi merupakan proses yang sangat penting karena dalam seleksi inilah dipilih induk-induk untuk perkawinan silang. Metode seleksi yang digunakan adalah metode *roulette whell* di mana peluang setiap individu untuk terpilih sebagai induk akan sangat tergantung pada nilai *fitness*-nya. Semakin besar nilai *fitness,* maka peluang terpilihnya individu tersebut akan semakin besar.

Setiap individu diurutkan berdasarkan besarnya nilai *fitness*. Dalam metode seleksi *roulette wheel,* setiap individu harus ditempatkan pada *roulette* sesuai dengan besarnya *fitness.* Pada Tabel 2.1, ditampilkan urutan tiap individu dan penempatannya di dalam *roulette.*

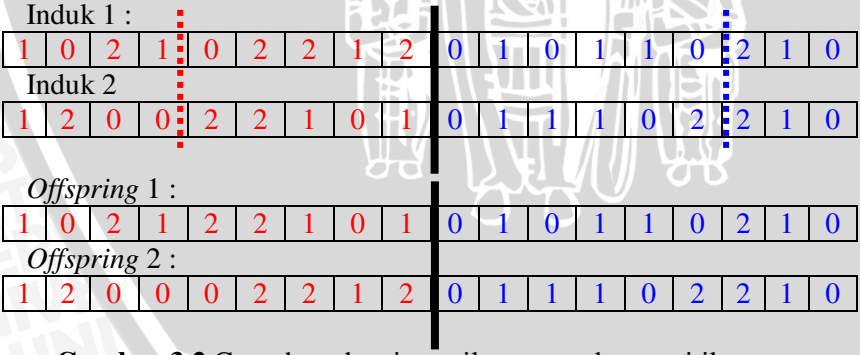
Tabel 3.3 Penempatan individu di dalam *roulette*

Individu	*Fitness*	Skala	*Roulette*
1 0 2 1 0 2 2 1 2 0 1 0 1 1 0 2 1 0	0.035227	39	1-39
2 0 2 1 1 0 0 1 0 1 2 2 1 1 0 0 2 0	0.027773	31	40 -70
1 2 0 0 2 2 1 0 1 0 1 1 1 0 2 2 1 0	0.0269149	30	71-100

Untuk memilih induk, maka diacak angka antara 1 sampai dengan 100. Misalnya angka yang muncul adalah 30, maka induk yang terpilih adalah individu ke satu.

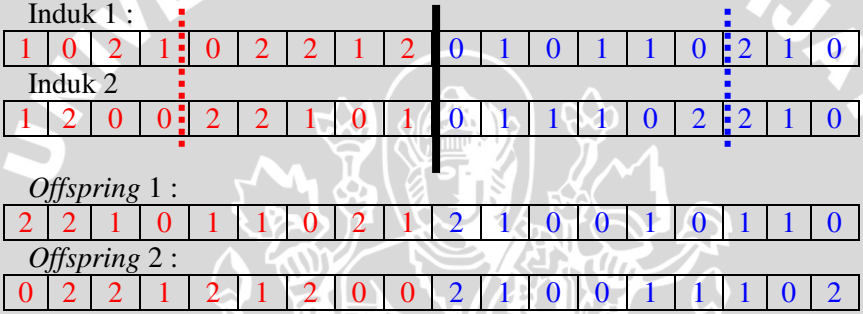
3.1.5 Perkawinan Silang

Setelah dilakukan penyeleksian terhadap beberapa induk, maka perkawinan silang antara pasangan-pasangan induk dilakukan. Metode perkawinan silang yang digunakan yaitu titik potong tunggal. Karena kromosom terdiri dari dua bagian, maka titik potong diletakkan di masing-masing bagian (absis dan ordinat), dan peletakkan titik potong tidak harus sama, agar lebih banyak kemungkinan yang bisa diperiksa. Misalnya setelah dilakukan seleksi, induk yang terpilih adalah kromosom 1 dan 3, proses perkawinan silangnya ditunjukkan oleh Gambar 3.2 :



Gambar 3.2 Contoh perkawinan silang metode satu titik potong biasa

Karena metode perkawinan silang pada Gambar 3.2 kemungkinan besar akan menghasilkan kromosom yang tidak bervariasi, maka dilakukan perbaikan dengan membalik hasil perkawinan silang. Titik potong yang digunakan tetap 1 titik potong, hanya saja absis anak pertama berasal dari absis induk ke dua setelah titik potong dan absis induk pertama sebelum titik potong. Sedangkan ordinat berasal dari ordinat induk kedua setelah titik potong dan ordinat induk pertama sebelum titik potong. Demikian sebaliknya dengan anak yang kedua. Untuk selanjutnya metode ini disebut dengan metode perkawinan silang satu titik potong pembalikan absis dan ordinat Ilustrasinya ditunjukkan oleh Gambar 3.3



Gambar 3.3 Contoh perkawinan silang metode satu titik potong pembalikan pengambilan absis dan ordinat

Selain kedua metode tersebut, juga digunakan metode ke tiga yaitu dengan memperhatikan *fitness* induk seperti yang sudah dijelaskan pada Bab II. Tiap-tiap gen (pasangan absis dan ordinat) ditinjau, pasangan mana yang memiliki nilai *fitness* tertinggi, pasangan tersebutlah yang akan terpilih membentuk *offspring*. Ilustrasinya ditunjukkan oleh Tabel 3.4, Tabel 3.5 dan Tabel 3.6

Tabel 3.6 menunjukkan bahwa anak yang dihasilkan adalah anak dengan nilai *fitness* yang lebih baik daripada induknya. Perkawinan seperti ini memang bisa menimbulkan konvergensi lebih cepat, tetapi masih ada peranan mutasi, sehingga konvergensi dini bisa dicegah. Untuk selanjutnya metode ini disebut dengan metode perkawinan memperhatikan *fitness* induk.

Offspring yang dihasilkan dari perkawinan silang adalah *offspring* yang legal karena semua nilai yang ada di dalam sebuah kromosom adalah nilai solusi yang mungkin sehingga tidak perlu dilakukan perbaikan (*repair).*

Tabel 3.4 Tabel gen dan *fitness* induk 1

Gen	Fitness
1,0	0.1
0,1	0.09
2,0	0.09
1,1	0.01
0,1	0.03
2,0	0.04
2,2	0.09
1,1	0.04
2,0	0.01

Tabel 3.5 Tabel gen dan *fitness* induk 2

Gen	Fitness
1,0	0.02
2,1	0.1
0,1	0.1
0,1	0.04
2,0	0.01
2,2	0.02
1,2	0.05
0,1	0.1
1,0	0.05

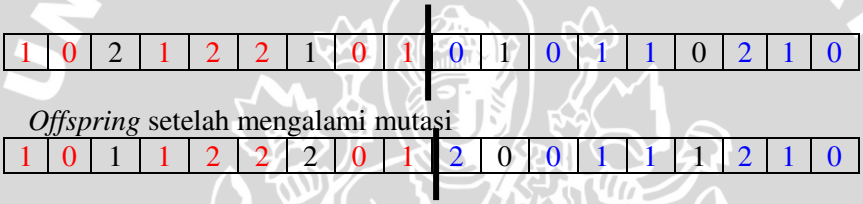
Tabel 3.6 Tabel gen dan *fitness* anak

Gen	Fitness
0,1	0.1
2,1	0.1
0,1	0.1
0,1	0.01
0,1	0.03
2,0	0.04
2,2	0.09
0,1	0.1
1,0	0.05

3.1.6 Mutasi

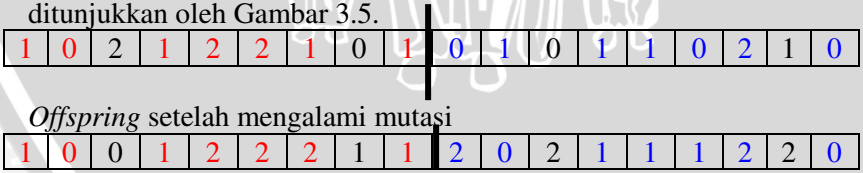
Mutasi merupakan salah satu operator genetik. Mutasi berperan agar semakin banyak kemungkinan dievaluasi sehingga mencegah terjadinya konvergensi dini. Metode yang dilakukan yaitu menukar dua buah nilai di dalam suatu kromosom.

Mutasi biasanya terjadi dengan peluang yang sangat kecil yang disebut probabilitas mutasi. Misalnya probabilitas mutasi adalah 0.05, maka akan dibangkitkan sebuah bilangan antara 0 sampai dengan 1, jika nilainya lebih kecil atau sama dengan 0.05, maka kromosom tersebut akan termutasi. Untuk itu perlu dibangkitkan dua buah bilangan acak sejumlah dua kali untuk menukar dua buah kromosom di bagian absis dan ordinat. Gambar 3.2 di bawah ini menunjukkan contoh mutasi



Gambar 3.4 Contoh mutasi metode pertukaran(*swap*)

Metode mutasi *swap* yang dilakukan dikhawatirkan tidak dapat mencegah terjadinya konvergensi dini, karena nilai-nilai yang muncul tidak berubah. Oleh sebab itu mutasi diperbaiki dengan metode penggantian beberapa persen gen di dalam kromosom. Di mana posisi gen yang diganti akan diacak dan diganti dengan bilangan acak pada interval tertentu (interval pada nilai absis dan ordinat yang mungkin) . Untuk selanjutnya metode ini disebut dengan metode mutasi penggantian. Ilustrasi metode mutasi dua ditunjukkan oleh Gambar 3.5.

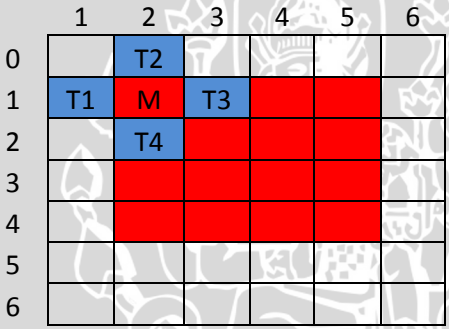


Gambar 3.5 Contoh mutasi metode penggantian

Pada Gambar 3.5 setelah diacak, posisi gen yang diganti adalah gen pada posisi ke tiga dan delapan, sehingga diacak nilai pengganti sebanyak dua kali untuk masing-masing bagian (absis dan ordinat).

Misalnya setelah diacak muncul angka nol dan satu untuk absis, maka posisi gen ke tiga untuk absis akan diganti oleh nol dan posisi ke delapan pada ordinat akan digantikan oleh satu. Demikian juga jika untuk ordinat nilai yang muncul adalah dua dan dua, maka posisi ke tiga pada gen bagian ordinat akan digantikan oleh nilai dua dan posisi ke delapan pada gen ordinat akan digantikan oleh dua.

Mutasi juga diperbaiki dengan metode pencarian optimum lokal di daerah tetangga gen yang terkena mutasi. Dalam metode ini akan diacak beberapa posisi gen yang akan termutasi, Kemudian daerah tetangga yang meliputi blok di sebelah kiri, kanan, atas dan bawah akan dicek *fitness*-nya, jika ada nilai *fitness* yang lebih tinggi, maka akan diambil untuk menggantikan gen tersebut. Selanjutnya metode ini disebut sebagai metode mutasi pencarian optimum lokal. Ilustrasinya ditunjukkan oleh Gambar 3.6.



Gambar 3.6 Ilustrasi mutasi pencarian optimum lokal

Pada Gambar 3.6 ditunjukkan bahwa gen yang termutasi adalah gen 1,1 (area berwarna merah). Kemudian diperiksa apakah tetangga dari gen tersebut yaitu 0.1 (T1), 1.0 (T2), 2.1 (T3) dan 1.2(T4). Jika *fitness* tertinggi dari keempat tetangga tersebut lebih tinggi daripada *fitness* gen yang termutasi, maka tetangga dengan *fitness* terbaik itulah yang akan menggantikan gen tersebut.

3.1.7 *Elitism*

Elitism berperan agar hanya individu-individu terkuat yang tetap bertahan di dalam suatu generasi. *Elitism* dilakukan dengan membandingkan nilai *fitness offspring* yang terbetuk dengan *fitness* semua individu di dalam suatu generasi. Jika nilai *fitness offspring* lebih besar dari salah satu atau beberapa individu di dalam suatu

generasi, maka *offpsring* tersebut akan masuk ke dalam generasi menggantikan individu dengan *fitness* terkecil.

Misalnya setelah dihitung diperoleh nilai *fitness* untuk *offspring* 1 0 2 0 2 2 1 0 0 0 1 0 1 1 2 2 1 0 adalah 0.032559, dan untuk *offspring* 1 0 2 1 0 2 2 1 2 0 1 0 1 1 0 2 1 0 adalah 0.032193. Generasi ke nol ditunjukkan oleh Tabel 3.7.

Tabel 3.7 Generasi ke nol

Individu	*Fitness*
1 0 2 1 0 2 2 1 2 0 1 0 1 1 0 2 1 0	0.035227
2 0 2 1 1 0 0 1 0 1 2 2 1 1 0 0 2 0	0.027773
1 2 0 0 2 2 1 0 1 0 1 1 1 0 2 2 1 0	0.026914939

Setelah membandingkan nilai *fitness offspring* dengan nilai *fitness* individu-individu di dalam populasi, maka didapatkan generasi ke satu seperti ditunjukkan oleh Tabel 3.8

Tabel 3.8 Generasi ke satu

Individu	*Fitness*
1 0 2 1 0 2 2 1 2 0 1 0 1 1 0 2 1 0	0.035227
1 0 2 0 2 2 1 0 0 0 1 0 1 1 2 2 1 0	0.032559
1 0 2 1 0 2 2 1 2 0 1 0 1 1 0 2 1 0	0.032193

3.1.8 Penyimpanan ke dalam Berkas Eksternal

Penyimpanan ke berkas eksternal diperlukan agar citra bisa direkonstruksi. *Header* berisi informasi jumlah blok jelajah ke arah horizontal dan vertikal serta ukuran blok jelajah. Kemudian isi berkas yaitu koordinat blok ranah, nilai *s,* dan nilai *o* untuk setiap blok jelajah dan disimpan di dalam berkas yang berupa *record* di mana *s* dan *o* bertipe *Single* sedangkan *x* dan *y* bertipe *SmallInt.*

3.2 Rekonstruksi Citra

Rekonstruksi citra dilakukan dengan menguraikan PIFS dari citra awal sembarang. Karena sesuai dengan konsep MRCM yang telah dijelaskan pada bab 2, citra apapun yang diinputkan ke mesin MRCM akan selalu menghasilkan sebuah citra yang konvergen. Selama proses rekonstruksi, IFS lokal mentransformasikan sekumpulan blok ranah menjadi sekumpulan blok jelajah. Proses rekonstruksi dilakukan dalam beberapa uraian yang berbeda-beda

untuk masing-masing citra. Pada umunya konvergensi terjadi pada uraian ke 8 sampai dengan uraian ke 10.

3.3 Mekanisme Kerja Sistem

Gambar 3.7 adalah langkah kerja proses kompresi fraktal dengan algoritma genetika yang telah dituangkan ke dalam *flowchart* (diagram alir).

Pada Gambar 3.7 ditunjukkan bahwa langkah pertama yang dilakukan sistem adalah membaca intensitas citra masukan, membaca ukuran blok jelajah dan kemudian membagi citra menjadi sejumlah blok yang berukuran sama dan tidak saling beririsan disebut dengan blok jelajah. Kemudian citra tersebut dibagi menjadi sejumlah blok yang saling beririsan dengan ukuran dua kali blok jelajah disebut blok ranah.

Langkah pertama algoritma genetika adalah membaca jumlah generasi, *pc, pm*. Selanjutnya membangkitkan populasi awal secara acak. Setiap individu terdiri dari dua bagian yaitu absis dan ordinat ujung kiri atas blok ranah. Kemudian dihitung *fitness* masing-masing individu. Langkah selanjutnya adalah melakukan tiga operator genetika yaitu seleksi yang bertujuan untuk memilih induk untuk perkawinan silang. Metode yang digunakan dalam proses seleksi adalah *roulette wheel*. Opeartor genetika kedua adalah perkawinan silang, yang dilakukan dengan metode titik potong tunggal. Setelah dilakukan perkawinan silang, maka akan dihasilkan sejumlah *offspring*. Operator genetika yang terakhir adalah mutasi. Mutasi dilakukan dengan melakukan pengecekan untuk setiap *offspring* apakah *offspring* tersebut terkena mutasi atau tidak, sesuai dengan probabilitas mutasi. Jika terkena, maka akan dilakukan mutasi. Setelah ketiga operator genetika dilakukan, kemudian *fitness* setiap *offspring* dihitung. Langkah terakhir adalah *elitism*, di mana *elitism* dilakukan untuk mempertahankan individu terkuat tetap bertahan di dalam suatu generasi. Caranya adalah dengan membandingkan *fitness offspring* yang terbentuk dengan individu-individu di dalam suatu populasi, jika *fitness ofsspring* lebih tinggi, maka *offspring* akan dimasukkan ke dalam generasi baru. Individu dengan *fitness* terendah di dalam populasi akan dikeluarkan. Selanjutnya adalah pengecekan apakah jumlah generasi telah terpenuhi, jika terpenuhi, maka proses berhenti dan keluaran berupa nilai absis (*x),* ordinat (*y),*

s,o. Sebaliknya jika tidak terpenuhi, maka kembali lagi melakukan ketiga operator genetika.

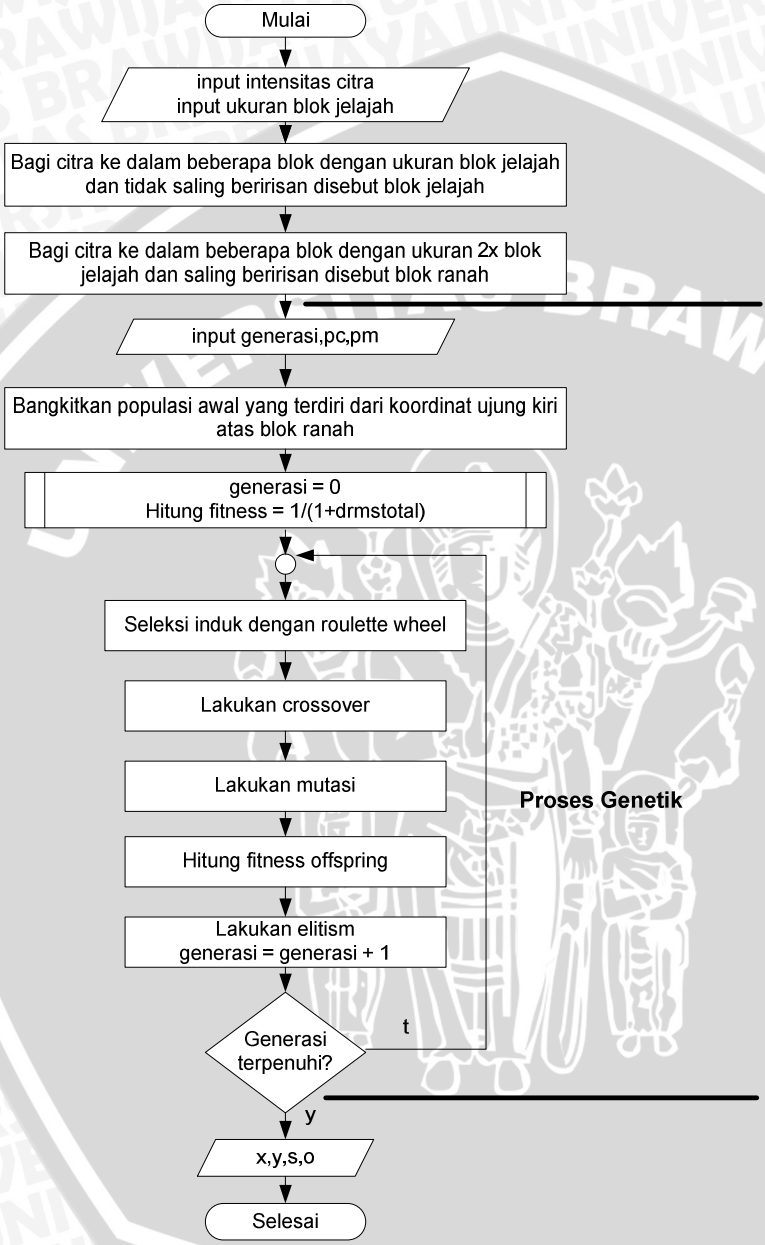
Gambar 3.8 adalah langkah kerja proses perhitungan $dtms_{total}$ yang telah dituangkan ke dalam *flowchart* (diagram alir).

Untuk menghitung *fitness* suatu individu, maka perlu dilakukan perhitungan $drms_{total}$, di mana $drms_{total}$ merupakan jarak seluruh blok jelajah dengan blok ranah yang bersesuaian di dalam suatu kromosom.

Langkah pertama adalah memasukkan ukuran blok jelajah. Kemudian nilai variabel *x* diberikan sejumlah seluruh blok jelajah, nilai variable *n* yaitu ukuran blok jelajah, kemudian *i* diset satu karena pengecekan akan dimulai dari blok jelajah yang pertama. Nilai awal untuk $drms_{total}$ adalah nol. Selanjutnya adalah membaca intensitas setiap *pixel* yang berada di dalam blok jelajah ke *i* dan blok ranah ke *i.* Kemudian nilai *i* dinaikkan sebanyak satu. Setelah semua nilai terbaca, dilakukan perhitungan nilai *s, o* dan *E* sesuai dengan rumus yang telah dijelaskan pada bab 2. Nilai *drms* adalah $\sqrt{E}/n$. Kemudian nilai *drms* yang didapatkan ditambahkan pada $drms_{total}$. Setelah itu dilakukan pengecekan apakah nilai *i* lebih besar daripada *x*. Jika lebih besar maka proses berhenti dan keluaran berupa nilai $drms_{total}$, *s,o.* Sedangkan jika tidak, maka proses kembali membaca intensitas blok ranah dan blok jelajah.

Gambar 3.9 adalah langkah kerja proses penyimpanan ke berkas yang telah dituangkan ke dalam *flowchart* (diagram alir).

Proses penyimpanan ke dalam berkas dimulai dengan membaca nilai jumlah blok jelajah, nilai absis (*x),* ordinat *(y), s, o* untuk masing-masing blok jelajah. Kemudian memberikan nilai *i* menjadi nol karena penyimpanan dimulai dari blok jelajah yang pertama, memberikan nilai *n* sama dengan jumlah blok jelajah. Kemudian membaca kromosom yang mempunyai nilai *fitness* tertinggi dan menyimpan ke variabel affine. Langkah selanjutnya yaitu memberikan nilai pada *x* yaitu affine yang ke *i*, dan *y* yaitu affine ke *i* ditambah *n*. Kemudian nilai *i* dinaikkan sebanyak satu, untuk mengecek blok jelajah berikutnya. Langkah selanjutnya yaitu menyimpan x,y,s dan o ke dalam berkas. Kemudian dilakukan pengecekan apakah *i* lebih besar daripada *n*. Jika tidak, maka proses dilakukan dengan membaca kembali nilai affine ke *i* dan ke *i*

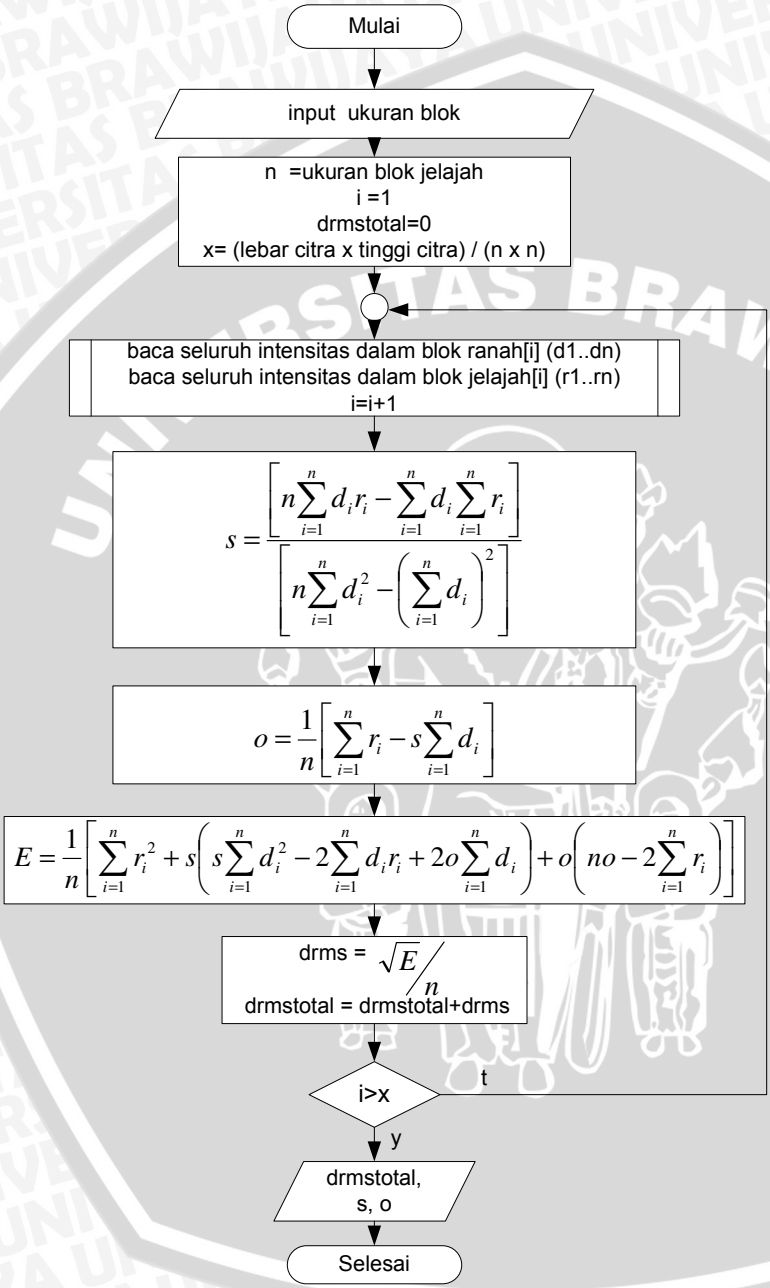


Gambar 3.7 *Flowchart* proses kompresi fraktal dengan algoritma genetika

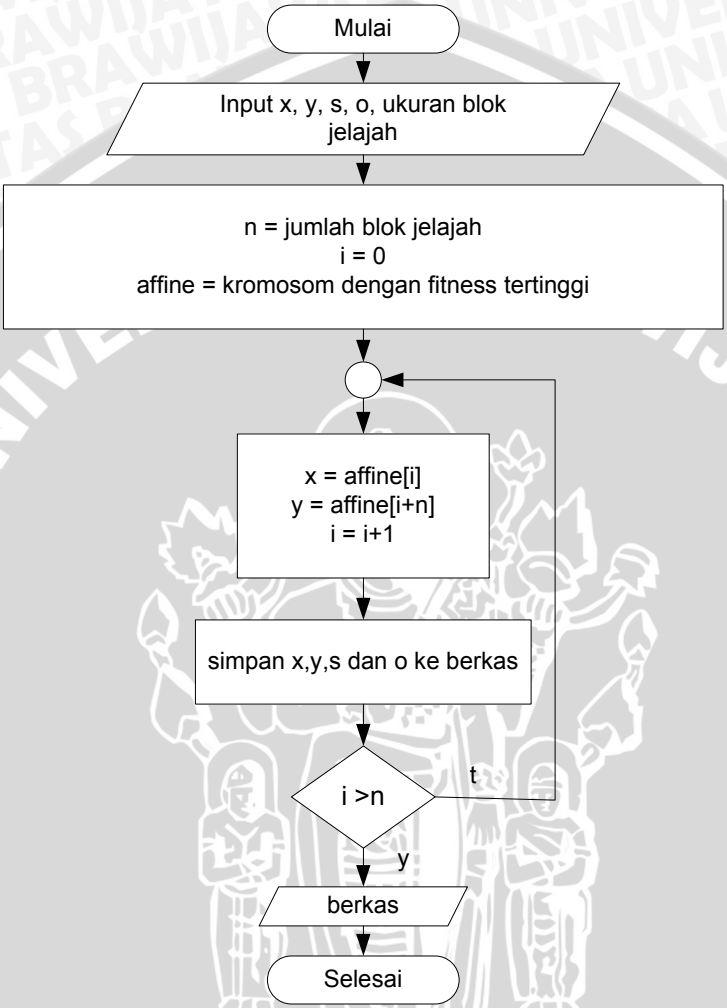
ditambah satu. Jika sebaliknya, maka proses dihentikan dan keluaran berupa berkas.

Gambar 3.10 adalah langkah kerja proses rekonstruksi citra yang telah dituangkan ke dalam *flowchart* (diagram alir).

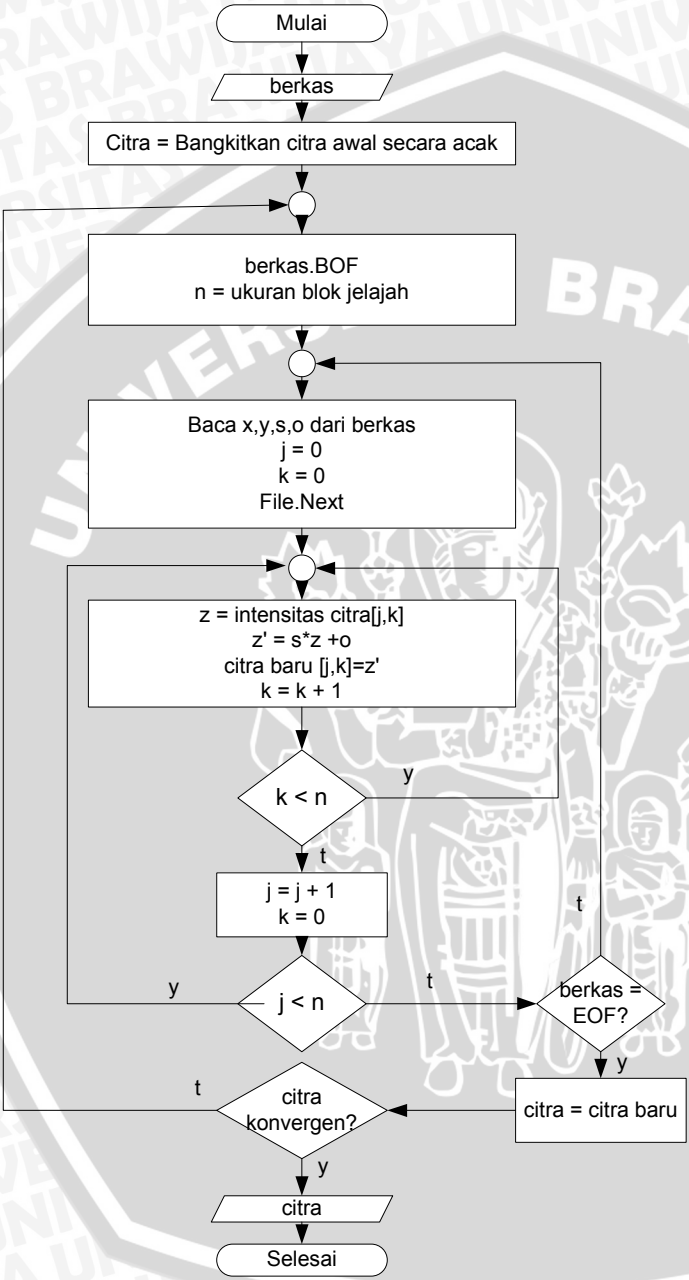
Langkah pertama dalam proses rekonstruksi citra adalah membaca masukan dari berkas hasil kompresi, kemudian membangkitkan suatu citra awal secara acak, karena citra awal yang dipakai adalah citra sembarang. Kemudian pointer file diarahkan untuk menuju ke *record* awal. Kemudian membaca ukuran blok jelajah dan disimpan ke dalam variabel *n*. Langkah selanjutnya yaitu membaca nilai *x, y, s* dan *o* dari berkas. Nilai *j* dan *k* diset dengan nol. Kemudian lanjutkan file ke *record* berikutnya. Setelah itu dilakukan pembacaan intensitas citra pada *pixel* ke *j,k* sebagai *z*. Nilai intensitas yang baru *z'* dihitung dengan rumus *s* dikali dengan *z* kemudian ditambahkan dengan *o*. Kemudian nilai *pixel* ke *j,k* diset dengan *z'*. Kemudian nilai *k* dinaikkan sebesar satu. Kemudian dilakukan pengecekan apakah nilai *k* lebih kecil daripada *n*, jika lebih kecil maka kembali ke proses membaca intensitas citra ke *j,k*. Jika tidak maka lanjutkan ke proses berikutnya yaitu nilai *j* dinaikkan sebesar satu dan nilai *k* diset nol. Langkah selajutnya yaitu dilakukan pengecekan apakah nilai *j* lebih kecil daripada *n*. Jika lebih kecil maka kembali ke proses membaca intensitas citra ke *j,k*. Jika tidak maka dilanjutkan ke proses berikutnya yaitu melakukan pengecekan apakah pointer sudah berada di akhir (*record* sudah habis). Jika tidak maka kembali ke proses membaca nilai *x, y, s* dan *o* dari berkas. Sebaliknya jika pointer sudah berada di akhir *record,* maka proses dilanjutkan yaitu menset citra dengan citra yang baru terbentuk dari hasil perhitungan intensitas baru dengan transformasi *affine*. Kemudian dilakukan pengecekan apakah citra konvergen. Jika tidak maka kembali ke proses mengarahkan pointer file untuk menuju ke *record* awal. Sebaliknya jika konvergen, maka proses dihentikan dan keluaran berupan citra yang sudah konvergen.



Gambar 3.8 *Flowchart* proses perhitungan $dtms_{total}$



Gambar 3.9 *Flowchart* proses penyimpanan ke berkas



Gambar 3.10 *Flowchart* proses rekonstruksi citra

3.4 Perancangan Uji Coba dan Evaluasi Hasil

Aplikasi kompresi citra fraktal dengan algoritma genetika dievaluasi untuk mengetahui tingkat efisiensi algoritma genetika dalam menyelesaikan kompresi citra fraktal. Adapun faktor yang dipakai sebagai tolak ukur adalah probabilitas perkawinan silang, probabilitas mutasi, nilai *fitness,* nisbah, dan waktu komputasi. Rancangan uji coba dan evaluasi hasil ditunjukkan oleh Tabel 3.9.

Tabel 3.9 Rancangan Tabel Uji Coba dan Evaluasi Hasil

No	pc	pm	Avg Min *drms*	Avg *dmrs*	Avg Max *Fitness*	Avg *Fitness*	Avg Rasio	Avg Waktu (s)	Avg MSE

Keterangan:

pc	:	Probabilitas *Crossover* merupakan peluang suatu perkawinan silang.
pm	:	Probabilitas Mutasi merupakan peluang terjadinya sutu mutasi pada *offspring* yang dihasilkan.
Avg Min drms		Nilai rata-rata *drms* atau matrik jarak terkecil yang dihasilkan dari 10 kali percobaan
Avg dmrs		Nilai rata-rata *drms* atau matrik jarak yang dihasilkan dalam satu generasi dari 10 kali percobaan
Max *Fitness*		Nilai rata-rata *fitness* tertinggi yang mampu dihasilkan dari 10 kali percobaan
Avg *Fitness*	:	Rata-rata *fitness* yang dihasilkan dalam satu generasi dari 10 kali percobaan
Rasio	:	Rata-rata perbandingan ukuran penyimpanan citra antara hasil pemampatan dengan citra asli dari 10 kali percobaan
Waktu (s)	:	Rata-rata waktu yang diperlukan

		untuk memampatkan sebuah citra dari 10 kali percobaan
MSE	:	Rata-rata kemiripan antara citra asli (sebelum dikompres) dengan citra hasil dekompresi dari 10 kali percobaan. MSE dihitung dengan matrik *rms* pada persamaan 2.7

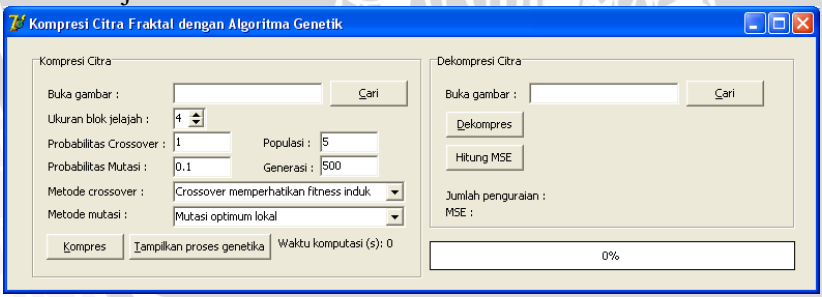
Ada tiga faktor yang dipakai untuk mengukur kinerja algoritma genetika dalam kompresi citra fraktal antara lain perbandingan ukuran citra sebelum dan sesudah dimampatkan, waktu komputasi dan MSE. Perbandingan dihitung dengan membandingkan ukuran penyimpanan citra digital dengan ekstensi .bmp dengan ukuran penyimpanan citra yang telah dimampatkan dengan kompresi fraktal dengan algoritma genetika, misalnya 20%. Rumus perhitungan rasio kompresi ditunjukkan oleh persamaan 3.1

Persen waktu = a*/b x 100%* (3.1)

di mama *a* adalah ukuran citra setelah dikompres, *b* adalah ukuran citra setelah dikompres. Faktor ketiga yaitu MSE adalah untuk mengukur kemiripan antara citra asli dengan citra hasil kompresi fraktal dengan algoritma genetika. MSE dihitung dengan matrik *rms* sesuai persamaan 2.7. Semakin kecil nilai MSE, maka semakin baik kualitas citra hasil pemampatan.

3.5 Perancangan Antar Muka

Rancangan antar muka untuk proses kompresi dan dekompresi citra ditunjukkan Gambar 3.11



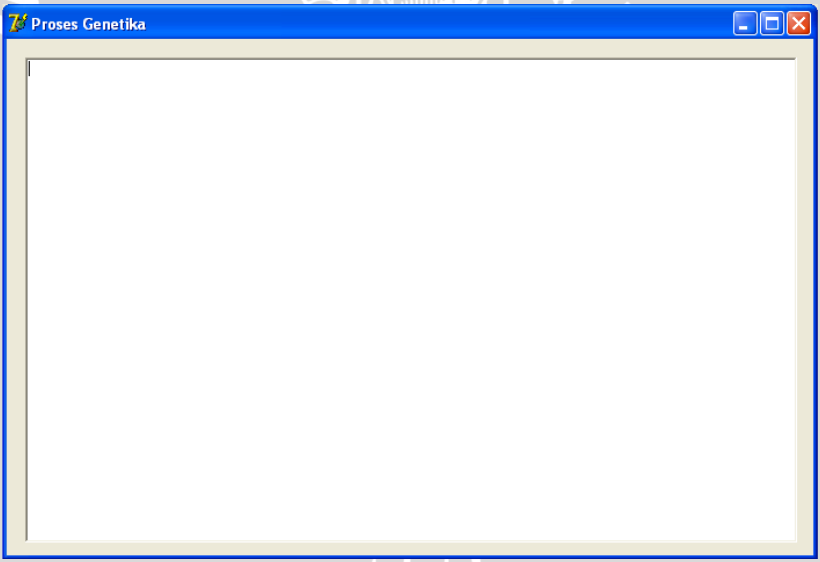
Gambar 3.11 Rancangan form kompresi dan dekompresi

Sedangkan rancangan form yang menunjukkan proses genetika pada saat kompresi citra ditunjukkan oleh Gambar 3.12

3.6 Studi Kasus Penerapan Algoritma Genetika Untuk Kompresi Citra Fraktal

Misalnya ada sebuah citra 6x6 dengan nilai intensitas seperti Gambar 3.1

Misalnya diberikan ukuran blok jelajah sebesar 2x2 dan ukuran blok ranah 4x4, sehingga ada 9 blok jelajah dan 9 blok ranah. Misalnya jumlah populasi awal ditentukan sebanyak 3, dengan probabilitas perkawinan silang p_c 0.6 dan probabilitas mutasi p_m 0.05. Dan perhitungan dilakuakan sampai generasi ke 1. Metode yang dipakai adalah metode *crossover* satu titik potong biasa dan mutasi pertukaran.



Gambar 3.12 Rancangan form proses genetika

3.6.1 Membangkitkan populasi awal

Misalnya populasi awal yang terbentuk adalah sebagai berikut :

1. 2 3 1 1 3 3 2 1 2 1 2 2 2 1 3 3 2 1
2. 3 1 3 2 2 1 1 2 1 2 3 3 2 2 1 1 3 1
3. 2 1 3 2 1 3 3 2 3 1 2 1 2 2 1 3 2 1

3.6.2 Seleksi

Sebelum proses seleksi dilakukan, maka harus dihitung terlebih dahulu nilai *fitness*-nya. Nilai fitness dihitung untuk setiap blok jelajah kemudian dijumlahkan untuk 1 gen. Jadi untuk setiap gen dilakukan perhitungan sebanyak sembilan kali, lalu dijumlahkan. Namun sebelum menghitung nilai *fitness*-nya, terlebih dahulu dilakukan penyekalaan untuk blok ranah dengan menjadikan 2x2 buah *pixel* menjadi satu buah *pixel*, di mana nilai satu buah *pixel* tersebut adalah nilai rata-rata dari keempat buah *pixel*. Kemudian dihitung nilai s dengan persamaan 2.12, o dengan persamaan 2.13, E dengan persamaan 2.14. Setelah itu baru dihitung nilai d_{rms} dengan persamaan 2.15. Nilai *fitness*-nya adalah $1/d_{rms}$.

Contoh :

Perhitungan untuk penyekalaan blok ranah pertama (0,0) yaitu

Pixel (1,1) = (25+30+35+40)/4 = 130/4 = 33
Pixel (3,1) = (55+34+33+70)/4 = 192/4 = 48
Pixel (1,3) = (24+15+34+55)/4 = 128/4 = 32
Pixel (3,3) = (53+48+57+78)/4 = 236/4 = 59

Perhitungan untuk penyekalaan blok ranah kedua (1,0) yaitu

Pixel (2,1) = (30+55+40+33)/4 = 158/4 = 40
Pixel (4,1) = (34+88+70+79)/4 = 289/4 = 72
Pixel (2,3) = (15+53+55+57)/4 = 180/4 = 45
Pixel (4,3) = (48+65+78+79)/4 = 270/4 = 68

Perhitungan untuk penyekalaan blok ranah ketiga (2,0) yaitu

Pixel (3,1) = (55+34+33+70)/4 = 192/4 = 48
Pixel (5,1) = (88+151+79+100)/4 = 436/4 = 109
Pixel (3,3) = (53+48+57+78)/4 = 236/4 = 59
Pixel (5,3) = (56+65+79+134)/4 = 334/4 = 84

Perhitungan untuk penyekalaan blok ranah keempat (0,1) yaitu

Pixel (1,2) = (35+40+24+15)/4 = 114/4 = 29
Pixel (3,2) = (33+70+53+48)/4 = 204/4 = 51
Pixel (1,4) = (34+55+56+72)/4 = 217/4 = 54
Pixel (3,4) = (57+78+49+80)/4 = 264/4 = 66

Perhitungan untuk penyekalaan blok ranah kelima (1,1) yaitu

Pixel (2,2) = (40+33+15+53)/4 = 141/4 = 35
Pixel (4,2) = (70+97+48+65)/4 = 280/4 = 70
Pixel (2,4) = (55+57+72+49)/4 = 233/4 = 58
Pixel (4,4) = (78+79+80+56)/4 = 293/4 = 73

Perhitungan untuk penyekalaan blok ranah keenam (2,1) yaitu

Pixel (3,2) = (33+70+53+48)/4 = 204/4 = 51
Pixel (5,2) = (79+100+65+56)/4 = 318/4 = 80
Pixel (3,4) = (57+78+49+80)/4 = 264/4 = 66
Pixel (5,4) = (79+134+56+110)/4 = 379/4 = 95

Perhitungan untuk penyekalaan blok ranah ketujuh (0,2) yaitu
Pixel (1,3) = (24+15+34+55)/4 = 128/4 = 32
Pixel (3,3) = (53+48+57+78)/4 = 236/4 = 59
Pixel (1,5) = (56+72+55+76)/4 = 259/4 = 65
Pixel (3,5) = (49+80+49+25)/4 = 203/4 = 51

Perhitungan untuk penyekalaan blok ranah kedelapan (1,2) yaitu
Pixel (2,3) = (15+53+55+57)/4 = 180/4 = 45
Pixel (4,3) = (48+65+78+79)/4 = 270/4 = 68
Pixel (2,5) = (72+49+76+49)/4 = 246/4 = 62
Pixel (4,5) = (80+56+25+63)/4 = 224/4 = 56

Perhitungan untuk penyekalaan blok ranah kesembilan (2,2) yaitu
Pixel (3,3) = (53+48+57+78)/4 = 236/4 = 59
Pixel (5,3) = (65+56+79+134)/4 = 334/4 = 84
Pixel (3,5) = (49+80+49+25)/4 = 203/4 = 51
Pixel (5,5) = (56+110+63+97)/4 = 326/4 = 82

Perhitungan s, o, E, drms, fitness pada masing-masing individu

a. Individu pertama

- Blok jelajah (0,0) dengan blok ranah (2,1)

  s = (4*(40*25+72*30+180*35+68*40)-(40+72+180+68) * (25 + 30+ 35 + 40))/(4 * (40^2 + 75^2 + 180^2 + 68^2)-(40+72+180+68)^2)
  = 0.04051

  o = ((25+30+35+40)-0.04051*(40+72+180+68))/4
  = 28.8541

  E = ((25^2+30^2+35^2+40^2)+0.04051 * (0.04051 * (40^2 +72^2+180^2+68^2) -2* (40*25+72*30+180*35+ 68*40) +2*28.8541*(40+72+180+68))+ 28.8541*(4* 28.8541 -2*(25+30+35+40)))/4
  = 26.2079

  drms = SQRT(26.2079)/4 = 1.279841

- Blok jelajah (2,0) dengan blok ranah (3,2)

s = (4*(51*55+80*34+66*33+95*70)-(51+80+66+95)* (55+34+33+70))/(4*(51^2+80^2+66^2+95^2)-(51+80+66+95)^2)
= 0.316135
o = ((55+34+33+70)-0.316135*(51+80+66+95))/4
= 24.92215
E = ((55^2+34^2+33^2+70^2)+0.316135*(0.316135* (51^2+80^2+66^2+95^2)-2*(51*55+80*34+66*33+95*70)+2*24.92215*(51+80+66+95))+24.92215*(4*24.92215-2*(55+34+33+70)))/4
= 211.8656
drms = SQRT(211.8656)/4 = 3.638901

- Blok jelajah (4,0) dengan blok ranah (1,2)
s = (4*(29*88+51*151+54*97+66*100)-(29+51+54+66)*(88+151+97+100))/(4*(29^2+51^2+54^2+66^2)-(29+51+54+66)^2)
= 0.407563
o = ((88+151+97+100)-0.407563*(29+51+54+66))/4
= 88.62185
E = ((88^2+151^2+97^2+100^2)+0.407563*(0.407563 * (29^2+51^2+54^2+66^2)-2*(29*88+51*151+54*97+66*100)+2*88.62185*(29+51+54+66))+88.62185*(4*88.62185-2*(88+151+97+100)))/4
= 577.8498
drms = SQRT (577.8498)/4 = 6.009627

- Blok jelajah (0,2) dengan blok ranah (1,2)
s = (4*(29*24+51*15+54*34+66*55)-(29+51+54+66) * (24+15+34+55))/(4*(29^2+51^2+54^2+66^2)-(29+51+54+66)^2)
= 0.738095
o = ((24+15+34+55)-0.738095*(29+51+54+66))/4
= -4.90475
E = ((24^2+15^2+34^2+55^2)+0.738095*(0.738095* (29^2+51^2+54^2+66^2)-2*(29*24+51*15+54*34+66*55)+2*-4.90475*(29+51+54+66))-4.90475*(4*-4.90475-2*(24+15+34+55)))/4
= 124.256

drms = SQRT (124.256)/4 = 2.786754

- Blok jelajah (2,2) dengan blok ranah (3,1)

  s = (4*(48*53+109*48+59*57+84*78)-(48+109+59+84) *(53+48+57+78))/(4*(48^2+109^2+59^2+84^2)-(48+ 109+59+84)^2)
  = -0.00405

  o = ((53+48+57+78)+0.00405*(48+109+59+84))/4
  = 59.30375

  E = ((53^2+48^2+57^2+78^2)-0.00405*(-0.00405* (48^2+ 109^2+59^2+84^2)-2*(48*53+109*48+59*57+84*78)+ 2*59.30375*(48+109+59+84))+59.30375*(4*59.30375- 2*(53+48+57+78)))/4
  = 130.4909

  drms = SQRT (130.4909)/4 = 2.855815

- Blok jelajah (4,2) dengan blok ranah (3,3)

  s = (4*(59*65+84*56+51*79+82*134)-(59+84+51+82) * (65+56+79+134))/(4*(59^2+84^2+51^2+82^2)-(59+84 +51+82)^2)
  = 0.623472

  o = ((65+56+79+134)-0.623472*(59+84+51+82))/4
  = 40.48043

  E = ((65^2+56^2+79^2+134^2)+0.623472*(0.623472* (59^2+84^2+51^2+82^2)-2*(59*65+84*56+51*79+ 82*134)+2*40.48043*(59+84+51+82))+40.48043*(4* 40.48043-2*(65+56+79+134)))/4
  = 837.7573

  drms = SQRT (837.7573)/4 = 7.236009

- Blok jelajah (0,4) dengan blok ranah (2,3)

  s = (4*(45*56+68*72+62*55+56*76)-(45+68+62+56) * (56+72+55+76))/(4*(45^2+72^2+55^2+76^2)- (45+72+55+76)^2)
  = 0.196767

  o = ((56+72+55+76)-0.196767*(45+68+62+56))/4
  = 53.38671

  E = ((56^2+72^2+55^2+76^2)+0.196767*(0.196767* (45^2 +68^2+62^2+56^2)-2*(45*56+68*72+62*55+56*76)

+2*53.38671*(45+68+62+56))+53.38671*(4*53.38671-
2*(56+72+55+76)))/4
= 78.20906
drms = SQRT (78.20906)/4 = 2.210897

- Blok jelajah (2,4) dengan blok ranah (1,2)
s = (4*(29*49+51*80+54*49+66*25)-(29+51+54+66)
(49+80+49+25))/(4(29^2+51^2+54^2+66^2)-
(29+51+54+66)^2)
= -0.4944
o = ((49+80+49+25)+0.4944*(29+51+54+66))/4
= 75.47
E = ((49^2+80^2+49^2+25^2)-0.4944*(-0.4944*(29^2
+51^2+54^2+66^2)-2*(29*49+51*80+54*49+66*25)+
2*75.47*(29+51+54+66))+75.47*(4*75.47-
2*(49+80+49+25)))/4
= 337.5569
drms = SQRT (337.5569)/4 = 4.59318

- Blok jelajah (4,4) dengan blok ranah (2,1)
s = (4*(40*56+72*110+180*63+68*97)-(40+72+180+68)*
(56+110+63+97))/(4*(40^2+75^2+180^2+68^2)-(40+72
+180+68)^2)
= -0.10499
o = ((56+110+63+97)+0.10499*(40+72+180+68))/4
= 90.9491
E = ((56^2+110^2+63^2+97^2)-0.10499*(-0.10499* (40^2+
72^2+180^2+68^2)-2*(40*56+72*110+180*63+68*97)
+2*90.9491*(40+72+180+68))+90.9491*(4*90.9491-
2*(56+110+63+97)))/4
= 477.3835
drms = SQRT (477.3835)/4 = 5.462277

Fitness = 1/(1+1.279841+3.638901+6.009627+2.86754 +
2.855815+7.236009+2.210897+4.59318+5.462277)
= 0.026914939

b. Individu kedua
- Blok jelajah (0,0) dengan blok ranah (2,1)

s = (4*(51*25+80*30+66*35+95*40)-(51+80+66+95)*(25+30+35+40))/(4*(51^2+80^2+66^2+95^2)-(51+80+66+95)^2)
= 0.276735

o = ((25+30+35+40)-0.276735*(51+80+66+95))/4
= 12.29835

E = ((25^2+30^2+35^2+40^2)+0.276735*(0.276735* (51^2+80^2+66^2+95^2)-2*(51*25+80*30+66*35+95*40)+2*12.29835*(51+80+66+95))+12.29835*(4*12.29835-2*(25+30+35+40)))/4
= 10.84076

drms = SQRT(10.84076)/4 = 0.823133

- Blok jelajah (2,0) dengan blok ranah (3,2)

s = (4*(32*55+59*34+65*33+51*70)-(32+59+65+51)*(55+34+33+70))/(4*(32^2+59^2+65^2+51^2)-(32+59+59+51)^2)
= -0.36969

o = ((55+34+33+70)-0.36959*(32+59+65+51))/4
= 28.87372

E = ((55^2+34^2+33^2+70^2)-0.36959*(-0.36969* (32^2+59^2+65^2+51^2)-2*(32*55+59*34+65*33+51*70)+2*28.87372*(32+59+65+51))+28.87372*(4*28.87372-2*(55+34+33+70)))/4
= 1638.911

drms = SQRT(1638.911)/4 = 10.12087

- Blok jelajah (4,0) dengan blok ranah (1,2)

s = (4*(59*88+84*151+51*97+82*100)-(59+84+51+82)*(88+151+97+100))/(4*(59^2+84^2+51^2+82^2)-(59+84+51+82)^2)
= 1.147922

o = ((88+151+97+100)-1.147922*(59+84+51+82))/4
= 29.79338

E = ((88^2+151^2+97^2+100^2)+1.147922*(1.147922*(59^2+84^2+51^2+82^2)-2*(59*88+84*151+51*97+82*100)+2*29.79338*(59+84+51+82))+29.79338*(4*29.79338-2*(88+151+97+100)))/4
= 338.0254

drms = SQRT (338.0254)/4 = 4.596367

- Blok jelajah (0,2) dengan blok ranah (1,2)

  s = (4*(35*24+70*15+58*34+73*55)-(35+70+58+73) *(24+15+34+55))/(4*(35^2+70^2+58^2+73^2)-(35+70+58+73)^2)
  = 0.363535

  o = ((24+15+34+55)-0.363535*(35+70+58+73))/4
  = 10.55144

  E = ((24^2+15^2+34^2+55^2)+0.363535*(0.363535*(35^2+70^2+58^2+73^2)-2*(35*24+70*15+58*34+73*55)+2*10.55144*(35+70+58+73))+10.55144*(4*10.55144-2*(24+15+34+55)))/4
  = 191.9628

  drms = SQRT (191.9628)/4 = 3.463766

- Blok jelajah (2,2) dengan blok ranah (3,1)

  s = (4*(35*53+70*48+58*57+73*78)-(35+70+58+73) *(53+48+57+78))/(4*(35^2+70^2+58^2+73^2)-(35+70+58+73)^2)
  = 0.325503

  o = ((53+48+57+78)-0.325503*(35+70+58+73))/4
  = 39.79532

  E = ((53^2+48^2+57^2+78^2)+0.325503*(0.325503*(35^2+70^2+58^2+73^2)-2*(35*53+70*48+58*57+73*78)+2*39.79532*(35+70+58+73))+39.79532*(4*39.79532-2*(53+48+57+78)))/4
  = 106.8196

  drms = SQRT (106.8196)/4 = 2.583839

- Blok jelajah (4,2) dengan blok ranah (3,3)

  s = (4*(33*65+48*56+32*79+59*134)-(33+48+32+59) *(65+56+79+134))/(4*(33^2+48^2+32^2+59^2)-(33+48+32+59)^2)
  = 1.802789

  o = ((65+56+79+134)-1.802789*(33+48+32+59))/4
  = 5.980073

  E = ((65^2+56^2+79^2+134^2)+1.802789* (1.802789*(33^2+48^2+32^2+59^2)-2*(33*65+48*56+32*79+

59*134)+2*5.980073*(33+48+32+59))+5.980073*(4*
5.980073-2*(65+56+79+134)))/4
= 509.369
drms = SQRT (509.369)/4 = 5.642301

- Blok jelajah (0,4) dengan blok ranah (2,3)
s = (4*(33*56+48*72+32*55+59*76)-(33+48+32+59)*
(56+72+55+76))/(4*(33^2+48^2+32^2+59^2)-
(33+48+32+59)^2)
= 0.818725
o = ((56+72+55+76)-0.818725*(33+48+32+59))/4
= 29.54483
E = ((56^2+72^2+55^2+76^2)+0.818725*(0.818725*
(33^2+48^2+32^2+59^2)-2*(33*56+48*72+32*55+
59*76)+2*29.54483*(33+48+32+59))+29.54483*(4*
29.54483-2*(56+72+55+76)))/4
= 3.563496
drms = SQRT (3.563496)/4 = 1.359308

- Blok jelajah (2,4) dengan blok ranah (1,2)
s = (4*(45*49+68*80+62*49+56*25)-(45+68+62+56)*
(49+80+49+25))/(4*(45^2+68^2+62^2+56^2)-
(45+68+62+56)^2)
= 1.245887
o = ((49+80+49+25)-1.245887*(45+68+62+56))/4
= -21.2
E = ((49^2+80^2+49^2+25^2)+1.245887* (1.245887*
(45^2+68^2+62^2+56^2)-2*(45*49+68*80+65*49
+56*25)+2*-21.2*(45+68+62+56))-21.2*(4*-21.2-
2*(49+80+49+25)))/4
= 177.5628
drms = SQRT (177.5628)/4 = 3.331317

- Blok jelajah (4,4) dengan blok ranah (2,1)
s = (4*(33*56+48*110+32*63+59*97)-(33+48+32+59)*
(56+110+63+97))/(4*(33^2+48^2+32^2+59^2)-
(33+48+32+59)^2)
= 1.691235
o = ((56+110+63+97)-1.691235*(33+48+32+59))/4

= 8.776895

E = ((56^2+110^2+63^2+97^2)+1.691235* (1.691235* (33^2+48^2+32^2+59^2)-2*(33*56+48*110+32*63+ 59*97)+2*8.776895*(33+48+32+59))+8.776895*(4* 8.776895-2*(56+110+63+97)))/4

= 152.2854

drms = SQRT (152.2854)/4 = 3.085099

Fitness = 1/(1+0.823133+10.12087+4.596367+3.463766 + 2.583839+5.642301+1.359308+3.331317+3.085099)

= 0.027773

c. Individu ketiga

- Blok jelajah (0,0) dengan blok ranah (2,1)

s = (4*(40*25+72*30+180*35+68*40)-(40+72+180+68) *(25+30+35+40))/(4*(40^2+72^2+180^2+68^2)- (40+72+180+68)^2)

= 0.042076

o = ((25+30+35+40)-0.042076*(40+72+180+68))/4

= 28.71316

E = ((25^2+30^2+35^2+40^2)+0.042076* (0.042076* (40^2+72^2+180^2+68^2)-2*(40*25+72*30+180*35+ 68*40)+2*28.71316*(40+72+180+68))+28.71316*(4*2 8.71316-2*(25+30+35+40)))/4

= 26.20091

drms = SQRT(26.20091)/4 = 1.279671

- Blok jelajah (2,0) dengan blok ranah (3,2)

s = (4*(29*55+51*34+54*33+66*70)-(29+51+54+66) *(55+34+33+70))/(4*(29^2+51^2+54^2+66^2)- (29+51+54+66)^2)

= 0.183473

o = ((55+34+33+70)-0.183473*(29+51+54+66))/4

= 38.82635

E = ((55^2+34^2+33^2+70^2)+0.183473* (0.183473* (29^2+51^2+54^2+66^2)-2*(29*55+51*34 + 54*33+ 66*70)+2*38.82635*(29+51+54+66))+38.82635*(4* 38.82635-2*(55+34+33+70)))/4

= 232.4912

drms = SQRT(232.4912)/4 = 3.811916

- Blok jelajah (4,0) dengan blok ranah (1,2)

  s = (4*(48*88+109*151+59*97+84*100)-(48+109+59+84)*(88+151+97+100))/(4*(48^2+109^2+59^2+84^2)-(48+109+59+84)^2)

  = 0.947795

  o = ((88+151+97+100)-0.947795*(48+109+59+84))/4

  = 37.91538

  E = ((88^2+151^2+97^2+100^2)+0.947795* (0.947795*(48^2+109^2+59^2+84^2)-2*(48*88+109*151+59*97+84*100)+2*37.91538*(48+109+59+84))+37.91538*(4*37.91538-2*(88+151+97+100)))/4

  = 108.486

  drms = SQRT (108.486)/4 = 2.603915

- Blok jelajah (0,2) dengan blok ranah (1,2)

  s = (4*(35*24+70*15+58*34+73*55)-(35+70+58+73)*(24+15+34+55))/(4*(35^2+70^2+58^2+73^2)-(35+70+58+73)^2)

  = 0.363535

  o = ((24+15+34+55)-0.363535*(35+70+58+73))/4

  = 10.55144

  E = ((24^2+15^2+34^2+55^2)+0.363535* (0.363535*(35^2+70^2+58^2+73^2)-2*(35*24+70*15+58*34+73*55)+2*10.55144*(35+70+58+73))+10.55144*(4*10.55144-2*(24+15+34+55)))/4

  = 191.9628

  drms = SQRT (191.9628)/4 = 3.463766

- Blok jelajah (2,2) dengan blok ranah (3,1)

  s = (4*(29*53+51*48+54*57+66*78)-(29+51+54+66)*(53+48+57+78))/(4*(29^2+51^2+54^2+66^2)-(29+51+54+66)^2)

  = 0.57563

  o = ((53+48+57+78)-0.57563*(29+51+54+66))/4

  = 30.2185

E = ((53^2+48^2+57^2+78^2)+0.57563* (0.57563* (29^2+51^2+54^2+66^2)-2*(29*53+51*48+54*57+ 66*78)+2*30.2185*(29+51+54+66))+30.2185*(4* 30.2185-2*(53+48+57+78)))/4
= 71.35399
drms = SQRT (71.35399)/4 = 2.111782

- Blok jelajah (4,2) dengan blok ranah (3,3)
  s = (4*(48*65+109*56+59*79+84*134)-(48+109+59+84)* (65+56+79+134))/(4*(48^2+109^2+59^2+84^2)- (48+109+59+84)^2)
  = 0.040954
  o = ((65+56+79+134)-0.040954*(48+109+59+84))/4
  = 80.42845
  E = ((65^2+56^2+79^2+134^2)+0.040954* (0.040954* (48^2+109^2+59^2+84^2)-2*(48*65+109*56+59*79+ 84*134)+2*80.42845*(48+109+59+84))+80.42845*(4* 80.42845-2*(65+56+79+134)))/4
  = 916.3183
  drms = SQRT (916.3183)/4 = 7.567687

- Blok jelajah (0,4) dengan blok ranah (2,3)
  s = (4*(59*56+84*72+51*55+82*76)-(59+84+51+82)* (56+72+55+76))/(4*(59^2+84^2+51^2+82^2)- (59+84+51+82)^2)
  = 0.633252
  o = ((56+72+55+76)-0.633252*(59+84+51+82))/4
  = 21.05561
  E = ((56^2+72^2+55^2+76^2)+0.633252 *(0.633252 * (59^2+84^2+51^2+82^2)-2*(59*56+84*72+51*55+ 82*76)+2*21.05561*(59+84+51+82))+21.05561*(4* 21.05561-2*(56+72+55+76)))/4
  = 5.681388
  drms = SQRT (5.681388)/4 = 0.595892

- Blok jelajah (2,4) dengan blok ranah (1,2)
  s = (4*(35*49+70*80+58*49+73*25)-(35+70+58+73)* (49+80+49+25))/(4*(35^2+70^2+58^2+73^2)- (35+70+58+73)^2)

= 0.005593

o = ((49+80+49+25)-0.005593*(35+70+58+73))/4

= 50.42001

E = ((49^2+80^2+49^2+25^2)+0.005593*(0.005593 * (35^2+70^2+58^2+73^2)-2*(35*49+70*80+58*49+ 73*25)+2*50.42001*(35+70+58+73))+50.42001*(4* 50.42001-2*(49+80+49+25)))/4

= 381.1805

drms = SQRT (381.1805)/4 = 4.880961

- Blok jelajah (4,4) dengan blok ranah (2,1)

s = (4*(48*56+109*110+59*63+84*97)-(48+109+59+84) *(56+110+63+97))/(4*(48^2+109^2+59^2+84^2)- (48+109+59+84)^2)

= 0.941944

o = ((56+110+63+97)-0.941944*(48+109+59+84))/4

= 10.8542

E = ((56^2+110^2+63^2+97^2)+0.941944*(0.941944* (48^2+109^2+59^2+84^2)-2*(48*56+109*110+59*63+ 84*97)+2*10.8542*(48+109+59+84))+10.8542*(4* 10.8542-2*(56+110+63+97)))/4

= 18.3777

drms = SQRT (18.3777)/4 = 1.07173

Fitness = 1/(1+1.279671+3.811916+2.603915+3.463766+ 2.111782+7.567687+0.595892+4.880961+1.07173)

= 0.035227

Untuk proses seleksi induk, setiap individu diurutkan berdasarkan besarnya nilai *fitness*. Metode seleksi yang digunakan adalah metode *roulette wheel*. Sehingga setiap individu harus ditempatkan pada *roulette* sesuai dengan besarnya *fitness*. Urutan tiap individu dan penempatannya di dalam *roulette* ditunjukkan oleh Tabel 3.10.

Tabel 3.10 Penempatan individu di dalam *roulette*

Individu	*Fitness*	Skala	*Roulette*
2 1 3 2 1 3 3 2 3 1 2 1 2 2 1 3 2 1	0.035227	39	1-39
3 1 3 2 2 1 1 2 1 2 3 3 2 2 1 1 3 1	0.027773	31	40-70
2 3 1 1 3 3 2 1 2 1 2 2 2 1 3 3 2 1	0.0269149	30	71-100

Untuk memilih dua buah induk, maka diacak angka antara 1 sampai dengan 100 sebanyak dua kali. Misalnya angka yang muncul adalah 30 dan 75, maka induk yang terpilih adalah individu ke satu dan individu ke tiga.

3.6.3 Perkawinan Silang dan Mutasi

Setelah diseleksi, didapatkan individu ke tiga dan individu ke satu sebagai induk. Proses perkawinan silang yang dilakukan adalah sebagai berikut :

P1 : 2 1 3 2 1 3 3 2 3 | 1 2 1 2 2 1 3 2 1
P2 : 2 3 1 1 3 3 2 1 2 | 1 2 2 2 1 3 3 2 1

Karena kromosom terdiri dari dua bagian, maka titik potong untuk proses perkawinan silang diletakkan pada masing-masing bagian, dan titik potong pada masing-masing bagian tidak harus sama, agar lebih banyak kemungkinan yang bisa diperiksa.

Misalnya setelah dibangkitkan secara acak diperoleh titik potong yaitu 3 dan 5, maka berikut ini adalah proses perkawinan silang yang terjadi :

P1 : 2 1 3 | 2 1 3 3 2 3 | 1 2 1 2 2 | 1 3 2 1
P2 : 2 3 1 | 1 3 3 2 1 2 | 1 2 2 2 1 | 3 3 2 1

C1 (*Offspring* 1) : 2 1 3 1 3 3 2 1 1 1 2 1 2 2 3 3 2 1
C2 (*Offspring* 2) : 2 3 1 2 1 3 3 2 3 1 2 2 2 1 1 3 2 1

Setelah didapatkan dua buah *offspring*, maka dilakukan pengecekan mutasi untuk kedua *offspring* sesuai dengan probabilitas mutasi.. Metode yang digunakan yaitu dengan membangkitkan bilangan antara 0 sampai dengan 1, untuk setiap *offspring*, jika bilangan yang muncul antara 0 sampai 0.05, maka *offspring* tersebut terkena mutasi. Maka dibangkitkan dua nilai secara acak dalam interval 1 sampai panjang kromosom. Kemudian ditukarkan kedua gen pada ururtan yang sesuai dengan nilai yang telah dibangkitkan sebelumnya. Karena kromosom terdiri dari dua bagian, maka pertukaran dilakukan masing-masing pada bagian absis dan ordinat secara terpisah. Misalnya setelah diacak muncul angka 0.85 untuk *offspring* pertama, maka *offspring* pertama tidak terkena mutasi. Kemudian untuk *offspring* kedua juga diacak bilangan antara 0 sampai 1, misalnya muncul 0.24, maka *offspring* kedua tidak terkena mutasi.

3.6.4 *Elitism*

Setelah proses perkawinan silang dan mutasi dilakukan, maka dilakukan proses elitism untuk memastikan individu-individu terkuatlah yang masuk ke dalam generasi yang baru. Untuk itu, masing-masing *offspring* yang dihasilkan harus dihitung dahulu nilai *fitness*-nya. Langkah perhitungan *fitness* adalah sebagai berikut :

1. *Offspring* 1

→ Blok jelajah (0,0) dengan blok ranah (2,1)

s = (4*(40*25+72*30+180*35+68*40)-(40+72+180+68)*(25+30+35+40))/(4*(40^2+72^2+180^2+68^2)-(40+72+180+68)^2)
= 0.042076

o = ((25+30+35+40)-0.042076*(40+72+180+68))/4
= 28.71316

E = ((25^2+30^2+35^2+40^2)+0.042076* (0.042076*(40^2+72^2+180^2+68^2)-2*(40*25+72*30+180*35+68*40)+2*28.71316*(40+72+180+68))+28.71316*(4*28.71316-2*(25+30+35+40)))/4
= 26.20091

drms = SQRT(26.20091)/4 = 1.279671

→ Blok jelajah (2,0) dengan blok ranah (3,2)

s = (4*(29*55+51*34+54*33+66*70)-(29+51+54+66) *(55+34+33+70))/(4*(29^2+51^2+54^2+66^2)-(29+51+54+66)^2)
= 0.183473

o = ((55+34+33+70)-0.183473*(29+51+54+66))/4
= 38.82635

E = ((55^2+34^2+33^2+70^2)+0.183473* (0.183473*(29^2+51^2+54^2+66^2)-2*(29*55+51*34+54*33+66*70)+2*38.82635*(29+51+54+66))+38.82635*(4*38.82635-2*(55+34+33+70)))/4
= 232.4912

drms = SQRT(232.4912)/4 = 3.811916

→ Blok jelajah (4,0) dengan blok ranah (1,2)

s = (4*(48*88+109*151+59*97+84*100)-(48+109+59+84)*(88+151+97+100))/(4*(48^2+109^2+59^2+84^2)-(48+109+59+84)^2)

= 0.947795

o = ((88+151+97+100)-0.947795*(48+109+59+84))/4
= 37.91538

E = ((88^2+151^2+97^2+100^2)+0.947795* (0.947795*
(48^2+109^2+59^2+84^2)-2*(48*88+109*151+59*97+
84*100)+2*37.91538*(48+109+59+84))+37.91538*(4*
37.91538-2*(88+151+97+100)))/4
= 108.486

drms = SQRT (108.486)/4 = 2.603915

→ Blok jelajah (0,2) dengan blok ranah (1,2)

s = (4*(29*24+51*15+54*34+66*55)-(29+51+54+66)*
(24+15+34+55))/(4*(29^2+51^2+54^2+66^2)-
(29+51+54+66)^2)
= 0.738095

o = ((24+15+34+55)-0.738095*(29+51+54+66))/4
= -4.90475

E = ((24^2+15^2+34^2+55^2)+0.738095* (0.738095*
(29^2+51^2+54^2+66^2)-2*(29*24+51*15+54*34+
66*55)+2*-4.90475*(29+51+54+66))-4.90475*(4*-
4.90475-2*(24+15+34+55)))/4
= 124.256

drms = SQRT (124.256)/4 = 2.786754

→ Blok jelajah (2,2) dengan blok ranah (3,1)

s = (4*(51*53+80*48+66*57+95*78)-(51+80+66+95)*
(53+48+57+78))/(4*(51^2+80^2+66^2+95^2)-
(51+80+66+95)^2)
= 0.456848

o = ((53+48+57+78)-0.456848*(51+80+66+95))/4
= 25.6501

E = ((53^2+48^2+57^2+78^2)+0.456848* (0.456848*
(51^2+80^2+66^2+95^2)-2*(51*53+80*48+66*57+
95*78)+2*25.6501*(51+80+66+95))+25.6501*(4*
25.6501-2*(53+48+57+78)))/4
= 74.87875

drms = SQRT (74.87875)/4 = 2.163313

→ Blok jelajah (4,2) dengan blok ranah (3,3)

s = (4*(59*65+84*56+51*79+82*134)-(59+84+51+82)*(65+56+79+134))/(4*(59^2+84^2+51^2+82^2)-(59+84+51+82)^2)
= 0.623472
o = ((65+56+79+134)-0.623472*(59+84+51+82))/4
= 40.48043
E = ((65^2+56^2+79^2+134^2)+0.623472* (0.623472*(59^2+84^2+51^2+82^2)-2*(59*65+84*56+51*79+82*134)+2*40.48043*(59+84+51+82))+40.48043*(4*40.48043-2*(65+56+79+134)))/4
= 837.7573
drms = SQRT (837.7573)/4 = 7.236009

→ Blok jelajah (0,4) dengan blok ranah (2,3)
s = (4*(45*56+68*72+62*55+56*76)-(45+68+62+56)*(56+72+55+76))/(4*(45^2+68^2+62^2+56^2)-(45+68+62+56)^2)
= 0.432035
o = ((56+72+55+76)-0.432035*(45+68+62+56))/4
= 39.79998
E = ((56^2+72^2+55^2+76^2)+0.432035* (0.432035*(45^2+68^2+62^2+56^2)-2*(45*56+68*72+62*55+56*76)+2*39.79998*(45+68+62+56))+39.79998*(4*39.79998-2*(56+72+55+76)))/4
= 74.21342
drms = SQRT (74.21342)/4 = 2.15368

→ Blok jelajah (2,4) dengan blok ranah (1,2)
s = (4*(29*49+51*80+54*49+66*25)-(29+51+54+66)*(49+80+49+25))/(4*(29^2+51^2+54^2+66^2)-(29+51+54+66)^2)
= -0.4944
o = ((49+80+49+25)+0.4944*(29+51+54+66))/4
= 75.47
E = ((49^2+80^2+49^2+25^2)-0.4944* (-0.4944*(29^2+51^2+54^2+66^2)-2*(29*49+51*80+54*49+66*25)+2*75.47*(29+51+54+66))+75.47*(4*75.47-2*(49+80+49+25)))/4
= 337.5569

drms = SQRT (337.5569)/4 = 4.59318

→ Blok jelajah (4,4) dengan blok ranah (2,1)

s = (4*(33*56+48*110+32*63+59*97)-(33+48+32+59)*(56+110+63+97))/(4*(33^2+48^2+32^2+59^2)-(33+48+32+59)^2)
= 1.691235

o = ((56+110+63+97)-1.691235*(33+48+32+59))/4
= 8.776895

E = ((56^2+110^2+63^2+97^2)+1.691235* (1.691235*(33^2+48^2+32^2+59^2)-2*(33*56+48*110+32*63+59*97)+2*8.776895*(33+48+32+59))+8.776895*(4*8.776895-2*(56+110+63+97)))/4
= 152.2854

drms = SQRT (152.2854)/4 = 3.085099

Fitness = 1/(1+1.279671+3.811916+2.603915+ 2.786754+2.163313+7.236009+2.15368+4.59318+3.085099)
= 0.032559

2. *Offspring* 2

→ Blok jelajah (0,0) dengan blok ranah (2,1)

s = (4*(40*25+72*30+180*35+68*40)-(40+72+180+68)*(25+30+35+40))/(4*(40^2+72^2+180^2+68^2)-(40+72+180+68)^2)
= 0.042076

o = ((25+30+35+40)-0.042076*(40+72+180+68))/4
= 28.71316

E = ((25^2+30^2+35^2+40^2)+0.042076* (0.042076*(40^2+72^2+180^2+68^2)-2*(40*25+72*30+180*35+68*40)+2*28.71316*(40+72+180+68))+28.71316*(4*28.71316-2*(25+30+35+40)))/4
= 26.20091

drms = SQRT(26.20091)/4 = 1.279671

→ Blok jelajah (2,0) dengan blok ranah (3,2)

s = (4*(51*55+80*34+66*33+95*70)-(51+80+66+95)*(55+34+33+70))/(4*(51^2+80^2+66^2+95^2)-(51+80+66+95)^2)

= 0.316135

o = ((55+34+33+70)-0.313135*(51+80+66+95))/4
= 25.14115

E = ((55^2+34^2+33^2+70^2)+0.316135* (0.316135*(51^2+80^2+66^2+95^2)-2*(51*55+80*34+66*33+95*70)+2*25.14115*(51+80+66+95))+25.14115*(4*25.14115-2*(55+34+33+70)))/4
= 211.9136

drms = SQRT(211.9136)/4 = 3.639313

→ Blok jelajah (4,0) dengan blok ranah (1,2)

s = (4*(29*88+51*151+54*97+66*100)-(29+51+54+66)*(88+151+97+100))/(4*(29^2+51^2+54^2+66^2)-(29+51+54+66)^2)
= 0.407563

o = ((88+151+97+100)-0.407563*(29+51+54+66))/4
= 88.62185

E = ((88^2+151^2+97^2+100^2)+0.407563* (0.407563*(29^2+51^2+54^2+66^2)-2*(29*88+51*151+54*97+66*100)+2*88.62185*(29+51+54+66))+88.62185*(4*88.62185-2*(88+151+97+100)))/4
= 577.8498

drms = SQRT (577.8498)/4 = 6.009627

→ Blok jelajah (0,2) dengan blok ranah (1,2)

s = (4*(35*24+70*15+58*34+73*55)-(35+70+58+73)*(24+15+34+55))/(4*(35^2+70^2+58^2+73^2)-(35+70+58+73)^2)
= 0.363535

o = ((24+15+34+55)-0.363535*(35+70+58+73))/4
= 10.55144

E = ((24^2+15^2+34^2+55^2)+0.363535* (0.363535*(35^2+70^2+58^2+73^2)-2*(35*24+70*15+58*34+73*55)+2*10.55144*(29+51+54+66))+10.55144*(4*10.55144-2*(24+15+34+55)))/4
= 122.9181

drms = SQRT (122.9181)/4 = 2.771711

→ Blok jelajah (2,2) dengan blok ranah (3,1)

s = (4*(33*53+46*48+32*57+59*78)-(33+48+32+59)*(53+48+57+78))/(4*(33^2+48^2+32^2+59^2)-(33+48+32+59)^2)
= 0.456848
o = ((53+48+57+78)-0.468127*(33+48+32+59))/4
= 38.87054
E = ((53^2+48^2+57^2+78^2)+0.468127* (0.468127*(33^2+48^2+32^2+59^2)-2*(33*53+48*48+32*57+59*78)+2*38.87054*(33+48+32+59))+38.87054*(4*38.87054-2*(53+48+57+78)))/4
= 80.52741
$drms$ = SQRT (80.52741)/4 = 2.243427

→ Blok jelajah (4,2) dengan blok ranah (3,3)
s = (4*(48*65+109*56+59*79+84*134)-(48+109+59+84)*(65+56+79+134))/(4*(48^2+109^2+59^2+84^2)-(48+109+59+84)^2)
= 0.040954
o = ((65+56+79+134)-0.040954*(48+109+59+84))/4
= 80.42845
E = ((65^2+56^2+79^2+134^2)+0.040954* (0.040954*(48^2+109^2+59^2+84^2)-2*(48*65+109*56+59*79+84*134)+2*80.42845*(48+109+59+84))+80.42845*(4*80.42845-2*(65+56+79+134)))/4
= 916.3183
$drms$ = SQRT (916.3183)/4 = 7.567687

→ Blok jelajah (0,4) dengan blok ranah (2,3)
s = (4*(59*56+84*72+51*55+82*76)-(59+84+51+82)*(56+72+55+76))/(4*(59^2+84^2+51^2+82^2)-(59+84+51+82)^2)
= 0.633252
o = ((56+72+55+76)-0.633252*(59+84+51+82))/4
= 21.05561
E = ((56^2+72^2+55^2+76^2)+0.633252* (0.633252*(59^2+84^2+51^2+82^2)-2*(59*56+84*72+51*55+82*76)+2*21.05561*(59+84+51+82))+21.05561*(4*21.05561-2*(56+72+55+76)))/4
= 5.681388

drms = SQRT (5.681388)/4 = 0.595892

→ Blok jelajah (2,4) dengan blok ranah (1,2)

s = (4*(35*49+70*80+58*49+73*25)-(35+70+58+73)*(49+80+49+25))/(4*(35^2+70^2+58^2+73^2)-(35+70+58+73)^2)
= 0.005593

o = ((49+80+49+25)+0.005593*(35+70+58+73))/4
= 51.07999

E = ((49^2+80^2+49^2+25^2)+0.005593* (0.005593*(35^2+70^2+58^2+73^2)-2*(35*49+70*80+58*49+73*25)+2*51.07999*(35+70+58+73))+51.07999*(4*51.07999-2*(49+80+49+25)))/4
= 381.6161

drms = SQRT (381.6161)/4 = 4.883749

→ Blok jelajah (4,4) dengan blok ranah (2,1)

s = (4*(48*56+109*110+59*63+84*97)-(48+109+59+84)*(56+110+63+97))/(4*(48^2+109^2+59^2+84^2)-(48+109+59+84)^2)
= 0.941944

o = ((56+110+63+97)-0.941944*(48+109+59+84))/4
= 10.8542

E = ((56^2+110^2+63^2+97^2)+0.941944*(0.941944*(48^2+109^2+59^2+84^2)-2*(48*56+109*110+59*63+84*97)+2*10.8542*(48+109+59+84))+10.8542*(4*10.8542-2*(56+110+63+97)))/4
= 18.3777

drms = SQRT (18.3777)/4 = 1.07173

Fitness = 1/(1+1.279671+3.639313+6.009627+2.771711+2.243427+7.567687+0.595892+4.883749+1.07173)
= 0.032193

Setelah nilai *fitness* masing-masing *offspring* dihitung, maka dilakukan perbandingan nilai *fitness offspring* dengan generasi sebelumnya. Jika nilai *fitness offspring* lebih besar daripada individu pada generasi sebelumnya, maka *offspring* tersebut akan masuk ke dalam generasi yang baru, dan individu pada generasi sebelumnya, yang memiliki *fitness* terkecil akan dikeluarkan. Setelah

dibandingkan, maka akan terbentuk generasi yang baru seperti Tabel 3.11

Tabel 3.11 Generasi baru (Generasi ke 1)

Individu	*Fitness*
2 1 3 2 1 3 3 2 3 1 2 1 2 2 1 3 2 1	0.035227
2 3 1 2 1 3 3 2 3 1 2 2 2 1 1 3 2 1	0.032193
2 1 3 1 3 3 2 1 1 1 2 1 2 2 3 3 2 1	0.032559

Setelah didapatkan sampai generasi ke satu, maka individu yang terbaik akan diambil sebagai PIFS terbaik, yaitu individu pertama 2 1 3 2 1 3 3 2 3 1 2 1 2 2 1 3 2 1

3.6.5 Penyimpanan ke dalam berkas eksternal

Setelah didapatkan kumpulan transformasi *affine* (PIFS) terbaik melalui pendekatan algoritma genetik, maka dilakukan penyimpanan ke berkas eksternal. Nilai yang disimpan yaitu absis, ordinat, *s*, dan *o*. Kromosom yang menghasilkan transformasi terbaik yaitu 2 1 3 2 1 3 3 2 3 1 2 1 2 2 1 3 2 1 dengan masing-masing nilai *s* dan *o* untuk setiap koordinat yaitu 0.042076, 28.71316, 0.183473, 38.82635, 0.947795, 37.91538, 0.363535, 10.55144, 0.57563, 30.2185, 0.040954, 80.42845, 0.633252, 21.05561, 0.005593, 50.42001, 0.941944, 10.8542.

3.6.6 Rekonstruksi Citra

Rekonstruksi citra dilakukan dengan menguraikan PIFS dari citra awal sembarang. Berikut ini adalah contoh proses penguraian yang dimulai dengan suatu citra sembarang, di mana intensitas tiap *pixel*-nya dibangkitkan secara acak. Citra yang telah dibangkitkan secara acak ditunjukkan oleh Gambar 3.13

Sebelum diuraikan, maka data dibaca terlebih dahulu dari berkas eksternal, lalu setiap blok ranah ditransformasikan menjadi sekumpulan blok jelajah.

0	25	84	14	16	78
45	55	200	100	54	115
35	65	185	33	54	66
25	87	154	24	68	154
58	65	44	89	25	88
152	77	56	99	35	74

Gambar 3.13 Nilai intensitas citra awal

Berikut adalah perhitungan penguraian pertama :

Uraian pertama :

- Data ke 1 (3 1 0.042076 28.71316)
  (0,0) = ROUND(0.042076*100+28.71316,0) = 33
  (1,0) = ROUND(0.042076*54+28.71316,0) = 31
  (0,1) = ROUND(0.042076*33+28.71316,0) = 30
  (1,1) = ROUND(0.042076*54+28.71316,0) = 31
- Data ke 2
  (2,0) = ROUND(0.183473*65+38.82635,0) = 51
  (3,0) = ROUND(0.183473*185+38.82635,0) = 73
  (2,1) = ROUND(0.183473*87+38.82635,0) = 55
  (3,1) = ROUND(0.183473*154+38.82635,0) = 67
- Data ke 3
  (4,0) = ROUND(0.947795*100+37.91538,0) = 133
  (5,0) = ROUND(0.947795*54+37.91538,0) = 89
  (4,1) = ROUND(0.947795*33+37.91538,0) = 69
  (5,1) = ROUND(0.947795*54+37.91538,0) = 89
- Data ke 4
  (0,2) = ROUND(0.363535*185+10.55144,0) = 78
  (1,2) = ROUND(0.363535*33+10.55144,0) = 23
  (0,3) = ROUND(0.363535*154+10.55144,0) = 67
  (1,3) = ROUND(0.363535*24+10.55144,0) = 19
- Data ke 5
  (2,2) = ROUND(0.57563*65+30.2185,0) = 68
  (3,2) = ROUND(0.57563*185+30.2185,0) = 137
  (2,3) = ROUND(0.57563*87+30.2185,0) = 80

(3,3) = ROUND(0.57563*154+30.2185,0) =119

- Data ke 6
  (4,2) = ROUND(0.040954*100+80.42845,0) = 85
  (5,2) = ROUND(0.040954*54+80.42845,0) = 83
  (4,3) = ROUND(0.040954*33+80.42845,0) = 82
  (5,3) = ROUND(0.040954*54+80.42845,0) = 83
- Data ke 7
  (0,4) = ROUND(0.633252*24+21.05561,0) = 36
  (1,4) = ROUND(0.633252*68+21.05561,0) = 64
  (0,5) = ROUND(0.633252*89+21.05561,0) = 77
  (1,5) = ROUND(0.633252*25+21.05561,0) = 37
- Data ke 8
  (2,4) = ROUND(0.005593*185+50.42001,0) = 51
  (3,4) = ROUND(0.005593*33+50.42001,0) = 51
  (2,5) = ROUND(0.005593*154+50.42001,0) = 51
  (3,5) = ROUND(0.005593*24+50.42001,0) = 51
- Data ke 9
  (4,4) = ROUND(0.941944*100+10.8542,0) = 105
  (5,4) = ROUND(0.941944*54+10.8542,0) = 62
  (4,5) = ROUND(0.941944*33+10.8542,0) = 42
  (5,5) = ROUND(0.941944*54+10.8542,0) =62

Dari hasil perhitungan, maka diperoleh sebuah citra dengan intensitas seperti yang ditunjukkan oleh Gambar 3.14

Citra hasil uraian pertama pada umumnya belum menyerupai citra asli, oleh sebab itu citra hasil uraian pertama harus diuraikan kembali sampai mencapai titik konvergen. Proses penguraian citra akan menghasilkan titik konvergen pada uraian tertentu. Biasanya konvergensi diperoleh setelah uraian ke 8 sampai 10.

33	31	51	73	133	89
30	31	55	67	69	89
78	23	68	137	85	83
67	19	80	119	82	83
36	64	51	51	105	62.
77	37	51	51	42	62

Gambar 3.14 Nilai intensitas citra uraian pertama

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Implementasi program dibuat dengan menggunakan Borland Delphi 7.0. Dalam implementasi dibuat beberapa fungsi dan prosedur untuk menyelesaikan permaslahan.

4.1.1 Struktur Data

Struktur data yang digunakan untuk sebuah individu adalah *record.* Struktur data tersebut adalah sebagai berikut :

```
type
      TKromosom = array of Smallint;

      TIndividual = record
          KromosomX: TKromosom;
          KromosomY: TKromosom;
          s :array of Single;
          o :array of Single;
          E :array of Single;
          drms : array of Single;
          drmsTotal : Single;
          Fitness: Single;
          bAtas,
          bBawah  : Smallint;
  end;
```

Dari struktur data tersebut dapat dijelaskan bahwa sebuah individu terdiri dari kromosomX yang merupakan *array* dari tipe data *SmallInt* dan menyimpan nilai absis blok ranah, kromosomY yang merupakan *array* dari tipe data *SmallInt* dan menyimpan nilai ordinat blok ranah, *s* merupakan *array* dari tipe data *Single* dan menyimpan nilai kontras, *o* merupakan *array* dari tipe data *Single* dan menyimpan *offset* kecerahan, *E* merupakan *array* dari tipe data *Single* dan menyimpan nilai kuadrat galat antara blok jelajah dan blok ranah, *drms* merupakan *array* dari tipe data *Single* dan menyimpan jarak antara blok ranah dengan blok jelajah, *drmsTotal* merupakan tipe data *Single* dan menyimpan jarak total blok ranah dengan blok jelajah, *fitness* merupakan tipe data *Single* dan

menyimpan nilai *fitness* dari suatu individu, *bAtas* merupakan tipe data *SmallInt* dan menyimpan batas tertinggi dari area yang dikuasai suatu individu (untuk seleksi *roulette wheel)*, *bBawah* merupakan tipe data *SmallInt* dan menyimpan batas terendah dari area yang dikuasai suatu individu (untuk seleksi *roulette wheel).*

Sedangkan untuk penyimpanan hasil kompresi digunakan *file* dari *record* sebagai berikut

```
type
    TTransforms = record
       s :Single;
       o:Single;
       x : Smallint;
       y : Smallint;
    end;
 TFileTransforms = file of TTransforms;
```

Dari struktur data tersebut dapat dijelaskan bahwa *record* TTransforms terdiri dari empat elemen yaitu *s* merupakan tipe data *Single* yang menyimpan nilai kontras, *o* merupakan tipe data *Single* yang menyimpan *offset* kecerahan, *x* merupakan tipe data *SmallInt* yang menyimpan nilai absis blok ranah, *y* merupakan tipe data *SmallInt* yang menyimpan ordinat blok ranah. Keempat elemen tersebut disimpan ke dalam berkas yang bertipe TFileTransforms, di mana TFileTransforms merupakan file bertipe TTransform.

4.1.2 Perhitungan Nilai *s*

Setiap gen di dalam individu akan memiliki nilai kontras, fungsi untuk menghitung nilai kontras ditunjukkan oleh Source Code 4.1

```
s:=0;
 for y:=0 to n-1 do
  for x:=0 to n-1
     s:=s+grayDomain[(xD+(x*2))+(xD+((x+1)*2))
        div 2,(yD+(y*2))+(yD+((y+1)*2)) div 2]
        *grayRange[xR+x,yR+y];
     s := power(n,2)*s;
     temp1:=0;
     for y:=0 to n-1 do
      for x:=0 to n-1 do
       temp1:=temp1 + grayDomain[(xD+(x*2))+
              (xD+((x+1)*2)) div
              2,(yD+(y*2))+(yD+((y+1)*2)) div
              2];
```

```
temp2:=0;
for y:=0 to n-1 do
  for x:=0 to n-1 do
    temp2:=temp2 + grayRange[xR+x,yR+y];

s:=s-temp1*temp2;

temp1 := 0;
for y:=0 to n-1 do
  for x:=0 to n-1 do
   temp1:=temp1 + Power(
           grayDomain[(xD+(x*2))+(xD+((x+1)*2
           )) div 2,(yD+(y*2))+(yD+((y+1)*2))
           div 2],2);
temp1 := power(n,2)* temp1;

temp2 := 0;
for y:=0 to n-1 do
  for x:=0 to n-1 do
    temp2:=temp2 +
           grayDomain[(xD+(x*2))+(xD+((x+1)*2
           )) div 2,(yD+(y*2))+(yD+((y+1)*2))
           div 2];
temp2 := Power(temp2,2);

if (temp1-temp2)=0 then
  Result :=0
else
  Result:=s/(temp1 - temp2);
```

Source Code 4.1 Source code fungsi perhitungan nilai *s*

Graydomain adalah suatu variabel bertipe *array* dari *byte* yang menyimpan nilai keabuan dari blok ranah yang sudah diskalakan. Sedangkan grayRange adalah suatu variabel bertipe *array* dari *byte* yang menyimpan nilai keabuan dari blok jelajah. Alasan penyimpanan nilai keabuan ke dalam *array* adalah untuk mempercepat pengaksesan nilai keabuan, karena nilai-nilai tersebut diakses berkali-kali, sehingga jika dilakukan pengaksesan secara langsung pada citra akan memerlukan waktu yang lama. Dengan

menyimpannya ke dalam *array,* maka pengaksesan akan dilakukan pada *array* dan waktu yang diperlukan juga akan lebih singkat.

4.1.3 Perhitungan Nilai *o*

Perhitungan *offset* kecerhan dilakukan dengan fungsi yang ditunjukkan oleh Source Code 4.2

```
o:=0;
  for y:=0 to n-1 do
    for x:=0 to n-1 do
      o:=o+ grayRange[xR+x,yR+y];

temp := 0;
  for y:=0 to n-1 do
    for x:=0 to n-1 do
      temp:=temp+
            grayDomain[(xD+(x*2))+(xD+((x+1)*2
            )) div 2,(yD+(y*2))+(yD+((y+1)*2))
            div 2];
temp := s*temp;

Result := (o - temp)/power(n,2);
```

Source Code 4.2 Source code perhitungan *o*

Perhitungan nilai *offset* kecerahan menggunakan nilai kontras (*s)* yang telah didapatkan melalui perhitungan sebelumnya.

4.1.4 Perhitungan Nilai *E*

Perhitungan nilai E(kuadrat galat antara blok jelajah dengan blok ranah) dilakukan dengan fungsi yang ditunjukkan oleh Source Code 4.3

```
temp:=0;
for y:=0 to n-1 do
  for x:=0 to n-1 do
    temp:=temp+ Power(grayRange[xR+x,yR+y],2);

temp1 := 0;
for y:=0 to n-1 do
  for x:=0 to n-1 do
    temp1:=temp1+
          Power(grayDomain[(xD+(x*2))+(xD+((x
```

```
                +1)*2)) div
                2,(yD+(y*2))+(yD+((y+1)*2)) div
                2],2);

    temp2 := 0;
    for y:=0 to n-1 do
      for x:=0 to n-1 do
        temp2:=temp2+
               grayDomain[(xD+(x*2))+(xD+((x+1)*2)
               )  div  2,(yD+(y*2))+(yD+((y+1)*2))
               div 2] *grayRange[xR+x,yR+y];

    temp3:=0;
    for y:=0 to n-1 do
      for x:=0 to n-1 do
        temp3:=temp3+
               grayDomain[(xD+(x*2))+(xD+((x+1)*2)
               )  div  2,(yD+(y*2))+(yD+((y+1)*2))
               div 2];

    temp4:=0;
    for y:=0 to n-1 do
      for x:=0 to n-1 do
        temp4:=temp4+ grayRange[xR+x,yR+y];

    Result := (temp + s*(s*temp1-
              2*temp2+2*o*temp3)+o*(power(n,2)*o-
              2*temp4))/Power(n,2);
```

Source Code 4.3 Source code perhitungan nilai *E*

Perhitungan nilai *offset* kecerahan menggunkan nilai kontras (*s)* dan nilai *offset* kecerahan(*o)* yang telah didapatkan melalui perhitungan sebelumnya.

4.1.5 Perhitungan Nilai *Fitness*

Untuk menghitung nilai *fitness* harus dilakukan perhitungan nilai *drms* dan *drmsTotal* terlebih dahulu. Fungsi untuk menghitung nilai *drms* adalah sebagai berikut :

```
Result := Power(E,0.5)/power(n,2);
```

Perhitungan *drms* menggunakan nilai E yang telah didapatkan melalui perhitungan sebelumnya.

Sedangkan fungsi untuk menghitung *drmsTotal* ditunjukkan oleh Source Code 4.4

```
drms := Ind.drms[0];
for i:=1 to panjangKromosom –1 do
  drms := drms + Ind.drms[i];

Result := drms;
```

Source Code 4.4 Source code perhitungan *drmsTotal*

Perhitungan *drmsTotal* dilakukan untuk menghitung nilai *drmsTotal* sebuah individu. Perhitungan dilakukan dengan menjumlahkan masing-masing *drms* gen di dalam suatu individu. Karena dalam struktur data yang digunakan *drms* disimpan sebagai *array* dengan indeks dimulai dari nol, maka penjumlahan dilakukan mulai dari *drms* ke nol sampai habis.

Setelah nilai *drmsTotal* diperoleh, maka nilai *fitness* bisa dihitung dengan prosedur sebagai berikut :

```
Ind.Fitness := 1/(1+drmsTotal);
```

4.1.6 Inisialisasi Populasi

Langkah pertama dari algoritma genetika adalah membangkitkan populasi awal secara random. Prosedur untuk membangkitkan populasi secara random ditunjukkan oleh Source Code 4.5

```
for i:=0 to panjangKromosom-1 do
begin
  Ind.KromosomX[i] :=
                   RandomRange(xAwal,xAkhir+1);
  Ind.KromosomY[i] :=
                   RandomRange(yAwal,yAkhir+1);
end;
```

Source Code 4.5 Source code inisialisasi populasi

Suatu individu terdiri dari kromosomX dan kromosomY yang dibangkitkan secara acak. Di mana interval angkan yang bisa dibangkitkan yaitu dari mulai dari xAwal sampai dengan xAkhir untuk kromosomX, sedangkan untuk kromosomY dimulai dari yAwal sampai dengan yAkhir. Setelah invidu dibangkitkan secara

acak, dilakukan perhitungan nilai *s, o, E, drms, drmTotal* dan *fitness* dengan prosedur yang ditunjukkan oleh Source Code 4.6

```
for y:=0 to tinggi div size -1 do
begin
 for x:=0 to lebar div size -1 do
 begin
    Ind.s[x+lebar div size*y] :=
          GetS(size,Ind.KromosomX[x+lebar div
          size*y],Ind.KromosomY[x+lebar div
          size*y],x*size,y*size);
    Ind.o[x+lebar div size*y] :=
          GetO(size,Ind.KromosomX[x+lebar div
          size*y],Ind.KromosomY[x+lebar div
          size*y],x*size,y*size,Ind.s[x+lebar div
          size*y]);
    Ind.E[x+lebar div size*y] :=
         GetE(size,Ind.KromosomX[x+lebar div
         size*y],Ind.KromosomY[x+lebar div
         size*y],x*size,y*size,Ind.s[x+lebar div
         size*y],Ind.o[x+lebar div size*y]);
    Ind.drms[x+lebar div size*y] :=
         GetDrms(Ind.E[x+lebar div size*y],size);

 end;
end;
drmsTotal := GetDrmsTotal(Ind,panjangKromosom);
Ind.drmsTotal := drmsTotal;
GetFitness(drmsTotal,Ind);
```

Source Code 4.6 Source code perhitungan *s, o, drms, drmsTotal* dan *fitness* sebuah individu.

Setelah dilakukan perhitungan *drmsTotal* dan *fitness,* selanjutnya adalah mengurutkan populasi berdasarkan *drmsTotal* secara *ascending*. Pengurutan dilakukan berdasarkan nilai *drmsTotal* agar lebih valid, karena pada beberapa kasus nilai *drmsTotal* sedikit berbeda, namun *fitness*-nya sama. Hal tersebut dikarenakan dalam perhitungan *fitness* telah terjadi pembulatan, karena nilai pembagi (*drmsTotal)* terlalu besar. Algoritma yang dipakai dalam proses pengurutan adalah *selection sort,* walaupun bukan algoritma pengurutan terbaik, namun algoritma ini mempunyai kinerja yang cukup baik.

Langkah selanjutnya yaitu melakukan perhitungan nilai bAtas dan bBawah untuk proses seleksi. Prosedurnya ditunjukkan oleh Source Code 4.7

```
komulatif := 0;
for i:=0 to nPop-2 do
begin
  //menggunakan skala fitness 0-100
  Populasi[i].bBawah := komulatif+1;
  skala :=
        Round(Populasi[i].Fitness/fitnessTotal*
        100);
  komulatif := skala;
  Populasi[i].bAtas := komulatif;
end;
Populasi[nPop-1].bBawah := komulatif+1;
Populasi[nPop-1].bAtas := 100;
```

Source Code 4.7 Source code perhitungan bAtas dan bBawah

4.1.7 Seleksi

Seleksi dengan *roulette wheel* dilakukan dengan fungsi yang ditunjukkan oleh Source Code 4.8

```
random := RandomRange (1, 101);
for i:=0 to nPop-1 do
begin
  if (random>=Populasi[i].bBawah) and
     (random<=Populasi[i].bAtas) then
  begin
      Result := i;
      break;
  end;
end;
```

Source Code 4.8 Source code untuk proses seleksi *roulette wheel*

Langkah seleksi dimulai dengan membangkitkan angka secara acak mulai dari 1 sampai 100, jika angka yang muncul ada diantara bAtas dan bBawah dari suatu individu, maka individu tersebut yang terpilih.

4.1.8 Perkawinan Silang

Setelah melakukan seleksi, maka dilakukan perkawinan silang untuk menghasilkan anak. Metode perkawinan silang yang digunakan ada tiga. Hal ini dilakukan untuk memaksimalkan hasil algoritma genetika. Prosedur yang digunakan untuk metode perkawinan silang dengan satu titik potong biasa ditunjukkan oleh Source Code 4.9

```
cut1 := RandomRange (1, panjangKromosom - 2);
cut2 := RandomRange (1, panjangKromosom - 2);

for i:=0 to cut1 do
begin
  Offspring1.KromosomX[i] := P1.KromosomX[i];
  Offspring2.KromosomX[i] := P2.KromosomX[i];
end;

for i:=0 to cut2 do
begin
  Offspring1.KromosomY[i] := P1.KromosomY[i];
  Offspring2.KromosomY[i] := P2.KromosomY[i];
end;

for i:=cut1+1 to panjangKromosom -1 do
begin
  Offspring1.KromosomX[i] := P2.KromosomX[i];
  Offspring2.KromosomX[i] := P1.KromosomX[i];
end;

for i:=cut2+1 to panjangKromosom -1 do
begin
  Offspring1.KromosomY[i] := P2.KromosomY[i];
  Offspring2.KromosomY[i] := P1.KromosomY[i];
end;
```

Source Code 4.9 Source code perkawinan silang dengan satu titik potong biasa

Langkah pertama dalam perkawinan silang ini adalah membangkitkan 2 titik potong secara acak, di mana satu titik potong (cut1) akan diletakkan di bagian kromosomX dan cut2 akan diletakkan di kromsomY. Selanjutnya perkawinan silang akan menghasilkan dua anak. Di mana kromosomX dari anak pertama

berasal dari kromosomX induk pertama sebelum cut1 dan kromosoX dari induk kedua setelah cut1, kromosomY dari anak pertama berasal dari kromosomY induk pertama sebelum cut2 dan kromosomY dari induk kedua setelah cut2. Demikian sebaliknya dengan anak yang kedua.

Sedangkan untuk metode perkawinan silang yang kedua hampir mirip dengan metode perkawinan silang yang pertama, hanya saja kromosomX dari anak pertama berasal dari kromosomX induk 2 setelah cut1 dan kromosomX dari induk pertama sebelum cut1, sedangkan kromosomY berasal dari kromosomY induk kedua setelah cut2 dan kromosomY dari induk pertama sebelum cut2. Hal ini dimaksudkan agar hasil yang diperoleh lebih bervariasi, sehingga lebih banyak kemungkinan yang dievaluasi. Demikian sebaliknya dengan anak yang ke dua. Prosedur perkawinan silang yang satu titik potong dengan pembalikan pengambilan absis dan ordinat ditunjukkan oleh Source Code 4.10

```
cut1 := RandomRange (1, panjangKromosom - 2);
cut2 := RandomRange (1, panjangKromosom - 2);

j:=0;
for i:=cut1+1 to panjangKromosom -1 do
begin
  Offspring1.KromosomX[j] := P2.KromosomX[i];
  Offspring2.KromosomX[j] := P1.KromosomX[i];
  inc(j);
end;
for i:=0 to cut1 do
begin
  Offspring1.KromosomX[j] := P1.KromosomX[i];
  Offspring2.KromosomX[j] := P2.KromosomX[i];
  inc(j);
end;
j:=0;
for i:=cut2+1 to panjangKromosom -1 do
begin
  Offspring1.KromosomY[j] := P2.KromosomY[i];
  Offspring2.KromosomY[j] := P1.KromosomY[i];
  inc(j);
end;
for i:=0 to cut2 do
begin
  Offspring1.KromosomY[j] := P1.KromosomY[i];
```

```
    Offspring2.KromosomY[j] := P2.KromosomY[i];
    inc(j);
  end;
```

Source Code 4.10 Source code perkawinan silang yang satu titik potong dengan pembalikan pengambilan absis dan ordinat

Metode perkawinan silang yang ketiga mencoba melakukan perbaikan terhadap metode pertama dan kedua dengan memilih pasangan gen yang mempunyai nilai *drms* terkecil dari masing-masing induk. Anak yang dihasilkan hanya satu, yaitu anak dengan nilai *fitness* yang lebih baik daripada kedua induknya. Dengan metode ini diharapkan nilai *fitness* yang dihasilkan akan lebih baik, selain itu konvergen bisa lebih cepat, sehingga waktu yang diperlukan lebih sedikit. Prosedur untuk perkawinan silang yang memperhatikan *fitness* induk ditunjukkan oleh Source Code 4.11

```
for i:=0 to panjangKromosom -1 do
begin
   if P1.drms[i]<=P2.drms[i] then
   begin
      Offspring.KromosomX[i]:=P1.KromosomX[i];
      Offspring.KromosomY[i]:=P1.KromosomY[i];
      Offspring.s[i]:=P1.s[i];
      Offspring.o[i]:=P1.o[i];
      Offspring.drms[i]:=P1.drms[i];
      Offspring.E[i]:=P1.E[i];
   end
   else
   begin
      Offspring.KromosomX[i]:=P2.KromosomX[i];
      Offspring.KromosomY[i]:=P2.KromosomY[i];
      Offspring.s[i]:=P2.s[i];
      Offspring.o[i]:=P2.o[i];
      Offspring.drms[i]:=P2.drms[i];
      Offspring.E[i]:=P2.E[i];
   end;
end;
GetFitness(panjangKromosom,Offspring);
```

Source Code 4.11 Source code perkawinan silang yang memperhatikan *fitness* induk

4.1.9 Mutasi

Mutasi perlu dilakukan untuk menghindari terjadinya konvergensi dini. Konvergensi dini memang akan mempercepat waktu, tapi tidak baik karena belum semua kemungkinan ditelusuri, sehingga dikhawatirkan nilai *fitness* kurang optimal. Metode mutasi yang digunakan yaitu mutasi *swap,* penggantian lima persen dari jumlah gen di dalam kromosom dengan nilai acak, penggantian lima persen gen di dalam kromosom dengan nilai tetangga gen yang terkena mutasi(pencarian optimum lokal)

Metode mutasi pertukaran dilakukan dengan prosedur yang ditunjukkan oleh Source Code 4.12

```
      //kromosom x
      i := RandomRange(1,panjangKromosom-2);
      repeat
        j := RandomRange(1,panjangKromosom -2);
      until i<>j;
      SwapInt (Ind.KromosomX[i], Ind.KromosomX[j]);

      //kromosom y
      i := RandomRange(1,panjangKromosom-2);
      repeat
        j := RandomRange(1,panjangKromosom -2);
      until i<>j;
      SwapInt (Ind.KromosomY[i], Ind.KromosomY[j]);
```

Source Code 4.12 Source code mutasi pertukaran

Sedangkan metode mutasi pergantian lima persen gen dilakukan dengan prosedur yang ditunjukkan oleh Source Code 4.13

```
    {5% gen terkena mutasi}
    i:= panjangKromosom div 20;
    for k:= 1 to i do
    begin
      //kromosom x
      j := RandomRange(0,panjangKromosom);
      Ind.KromosomX[j] :=
RandomRange(xAwal,xAkhir+1);

      //kromosom y
      j := RandomRange(0,panjangKromosom);
      Ind.KromosomY[j] :=
```

```
RandomRange(yAwal,yAkhir+1);
    end;
```

Source Code 4.13 Source code mutasi pergantian lima persen gen

Metode mutasi dengan pencarian optimum lokal dilakukan dengan prosedur yang ditunjukkan oleh Source Code 4.14

```
i:= panjangKromosom div 20;
for k:= 1 to i do
begin
  j := RandomRange(0,panjangKromosom);
  Ind.s[j] :=
      GetS(size,Ind.KromosomX[j],Ind.KromosomY[j
      ],(j mod (lebar div size))*size,(j div
      (lebar div size))*size);
  Ind.o[j] :=
      GetO(size,Ind.KromosomX[j],Ind.KromosomY[j
      ],(j mod (lebar div size))*size,(j div
      (lebar div size))*size,Ind.s[j]);
  Ind.E[j] :=
      GetE(size,Ind.KromosomX[j],Ind.KromosomY[j
      ],(j mod (lebar div size))*size,(j div
      (lebar div size))*size,Ind.s[j],Ind.o[j]);
  Ind.drms[j] :=GetDrms(Ind.E[j],size);

  //menghitung drms dari kromosom[j]
  drmsMin := Ind.drms[j];
  xGanti := Ind.KromosomX[j];
  yGanti := Ind.KromosomY[j];
  sGanti := Ind.s[j];
  oGanti := Ind.o[j];
  EGanti := Ind.E[j];
  drmsGanti := Ind.drms[j];

  //mengecek drms tetangga blok ranah terpilih
  //x-1,y  (samping kiri)
  Ganti(j,-1,0,xGanti,yGanti,sGanti,oGanti,
       EGanti,drmsGanti,drmsMin);
  //x+1,y  (samping kanan)
  Ganti(j,1,0,xGanti,yGanti,sGanti,oGanti,EGanti,
       drmsGanti,drmsMin);
  //x,y-1  (atas)
  Ganti(j,0,-1,xGanti,yGanti,sGanti,oGanti,
       EGanti,drmsGanti,drmsMin);
```

```
//x,y+1  (bawah)
Ganti(j,0,1,xGanti,yGanti,sGanti,oGanti,
      EGanti,drmsGanti,drmsMin);

//mengganti gen dg koordinat yg memiliki drms
terendah
    Ind.KromosomX[j] := xGanti;
    Ind.KromosomY[j] := yGanti;
    Ind.s[j] := sGanti;
    Ind.o[j] := oGanti;
    Ind.E[j] := EGanti;
    Ind.drms[j] := drmsGanti;
  end;
```

Source Code 4.14 Source code mutasi pencarian optimum lokal

Prinsip kerja metode yang ketiga ini adalah mengecek tetangga samping kiri, kanan, atas dan bawah dari suatu gen yang terkena mutasi. Nilai gen akan digantikan dengan pasangan absis dan ordinat yang memiliki nilai *drms* terkecil. Untuk itu perlu dilakukan perhitungan nilai *s, o, E,* dan *drms* dari suatu gen yang terkena mutasi agar bisa dibandingkan dengan tetangganya. Setelah diketahui pasangan absis dan ordinat yang memiliki *drms* terkecil, maka dilakukan pergantian gen oleh pasangan absis dan ordinat yang memiliki *drms* terkecil tersebut. Pengecekan tetangga dilakukan oleh prosedur Ganti ditunjukkan oleh Source Code 4.15

```
s := GetS(size,Ind.KromosomX[index]+geserX,
     Ind.KromosomY[index]+geserY,(index mod (lebar
     div size))*size,(index div (lebar div
     size))*size);
o := GetO(size,Ind.KromosomX[index]+geserX,
      Ind.KromosomY[index]+geserY,(index mod (lebar
      div size))*size,(index div (lebar div
      size))*size,s);
E := GetE(size,Ind.KromosomX[index]+geserX,
      Ind.KromosomY[index]+geserY,(index mod (lebar
      div size))*size,(index div (lebar div
      size))*size,s,o);
drms := GetDrms(E,size);
if drmsMin>drms then
begin
  xGanti := Ind.KromosomX[index]-1;
```

```
    yGanti := Ind.KromosomY[index];
    sGanti := s;
    oGanti := o;
    EGanti := E;
    drmsGanti := drms;
end;
```

Source Code 4.15 Source code prosedur Ganti

Setelah melakukan perhitungan *s, o, E* dan *drms*, dilakukan perbandingan dengan *drmsMin*, jika nilai *drms* yang didapatkan lebih kecil daripada nilai *drmsMin* maka pasangan yang sebelumnya memiliki *drms* terkcil akan digantikan oleh pasangan yang sedang ditinjau.

4.1.10 Penyimpanan ke Dalam Berkas

Nilai-nilai yang telah didapatkan dari proses genetika berupa nilai *s, o, x* dan *y* merupakan nilai transformasi dari citra yang dimampatkan. Sehingga nilai-nilai tersebut akan disimpan ke dalam berkas. Sebelum disimpan ke dalam berkas, nilai-nilai tersebut dimasukkan terlebih dahulu ke dalam *record*. Adapun prosedur yang dilakukan untuk penyimpanan ke dalam berkas ditunjukkan oleh Source Code 4.16

```
      AssignFile(fl,SaveDialog1.filename);
      Rewrite(fl);

     //menyimpan header
     rec.s :=0;
     rec.o :=0;
     rec.x :=lebar div size;
     rec.y :=tinggi div size;
     write(fl,rec);
     rec.s :=0;
     rec.o :=0;
     rec.x :=size;
     rec.y :=0;
     write(fl,rec);


      for i:= 0 to panjangKromosom-1 do
      begin
           rec.s :=populasi[0].s[i];
```

```
            rec.o :=populasi[0].o[i];
            rec.x :=populasi[0].kromosomX[i];
            rec.y :=populasi[0].kromosomY[i];
            write(fl,rec);
        end;
        CloseFile(fl);
```

Source Code 4.16 Source code penyimpanan ke dalam berkas

Sebelum menyimpan informsi yang sebenarnya, terlebih dahulu dilakukan penyimpanan *header* yang terdiri dari jumlah blok jelajah ke arah horizontal, jumlah blok jelajah ke arah vertikal dan ukuran blok jelajah. Informasi ini sangat penting untuk proses dekompresi. Selanjutnya dilakukan penyimpanan *s, o, x* dan *y* ke dalam *record*, kemudian record tersebut akan disimpan ke dalam berkas tertentu.

4.1.11 Dekompresi citra

Dekompresi citra dilakukan dengan prosedur yang ditunjukkan oleh Source Code 4.17

```
    AssignFile(fl,fileName);
    reset(fl);
    n:=filesize(fl);

    ListS := TStringList.Create;
    ListO := TStringList.Create;
    ListX := TStringList.Create;
    ListY := TStringList.Create;

    for i := 0 to n-1 do
    begin
       read(fl,rec);
       ListS.Add(floattostr(rec.s));
       ListO.Add(floattostr(rec.o));
       ListX.Add(inttostr(rec.x));
       ListY.Add(inttostr(rec.y));

    end;

    //membaca informasi header
    jumlahH := strtoint(ListX[0]);
    jumlahV := strtoint(ListY[0]);
    sizeAsli := strtoint(ListX[1]);
    sizeDekompres := sizeAsli;
```

```
lebar := jumlahH * sizeDekompres;
tinggi := jumlahV * sizeDekompres;
Img1.Width := lebar;
Img1.Height := tinggi;

//membangkitkan Img1 secara acak
for y := 0 to tinggi -1 do
  for x := 0 to lebar -1 do
  begin
    Img1.Canvas.Pixels[x,y] := RGB(0,0,0);
    grayBefore[x,y] := 0;
  end;

SaveGrayDomain(Img1);

j:= (jumlahH * jumlahV);
iterasi :=0;
while (j>(jumlahH * jumlahV) div 20)  do
begin
  j:=0;
  i:=2;
  xR :=0;
  yR:=0;
  while i<= n -1 do
  begin
    //membaca xD
    xD := StrToInt(listX[i]);
    //Inc(i);
    //membaca yD
    yD := StrToInt(listY[i]);
    //Inc(i);
    //membaca s
    s := StrToFloat(listS[i]);
    //Inc(i);
    //membaca o
    o := StrToFloat(listO[i]);
    Inc(i);

    for y:=0 to sizeDekompres - 1 do
      for x :=0 to sizeDekompres -1 do
      begin
          z := s*
```

```
            GrayDomain[(xD+(x*2))+(xD+((x+1)
            *2)) div 2,(yD+(y*2))+(yD+((y+1)*2))
            div 2]+o;
          bulat :=Round(z);
          if (bulat<=255) and (bulat>=0) then
            grayRange[xR+x,yR+y]:= bulat
          else if bulat<0 then
         grayRange[xR+x,yR+y]:= 0
        else
         grayRange[xR+x,yR+y]:= 255;

      if abs(grayRange[xR+x,yR+y]-
      grayBefore[xR+x,yR+y])>10 then
          Inc(j);

    GrayBefore[xR+x,yR+y]:=grayRange[xR+x,yR+y];

    end;

    if xR >= (jumlahH-1)*sizeDekompres then
    begin
        xR :=0;
        Inc(yR,sizeDekompres);
    end
    else
    begin
       Inc(xR,sizeDekompres);
    end;
  end; //end i

  SaveGrayDomainDekompres(lebar,tinggi);
  Inc(iterasi);
end; //end j

for y := 0 to tinggi -1 do
  for x := 0 to lebar -1 do
  begin
    Img1.Canvas.Pixels[x,y] :=
      RGB(grayRange[x,y],grayRange[x,y],grayRan
      ge[x,y]);
  end;
CloseFile(fl);
```

Source Code 4.17 Source code dekompresi citra

Langkah pertama yang dilakukan adalah membaca berkas yang kemudian dimasukkan ke dalam *list*, hal ini dimaksudkan untuk mempercepat waktu karena pengaksesan nilai *s, o, x* dan *y* dilakukan berkali-kali sampai citra hasil dekompresi konvergen. Setelah semua *record* dimasukkan ke dalam *list*, maka dilakukan pembacaan *header* untuk menentukan ukuran citra hasil dekompresi. Kemudian dilakukan pembangkitan citra awal secara acak, di mana citra awal dibangkitkan sebagai citra dengan nilai keabuan seratus dua puluh tujuh. Nilai diambil secara acak, karena berdasarkan teori citra awal dekompresi adalah citra sembarang. Setelah citra dibangkitkan dilakukan penyekalaan blok ranah dan hasil penyekalaan tersebut akan disimpan ke dalam *array grayDomain*. Kemudian dilakukan pembacaan nilai *s, o, x* dan *y* dan dilakukan perhitungan nilai intensitas blok jelajah dengan persamaan 2.9. Setelah nilai intensitas didapatkan, maka dibandingkan dengan nilai intensitas sebelumnya yang telah disimpan ke dalam *grayBefore*. Jika perbedaan warna lebih dari sepuluh maka dianggap tidak sama. Apabila jumlah *pixel* yang berbeda lebih dari lima persen, maka citra dianggap belum konvergen dan proses dekompresi diulangi mulai dari pembacaan nilai *s,o,x* dan *y* hingga citra dinyatakan konvergen.

4.1.11 Perhitungan MSE

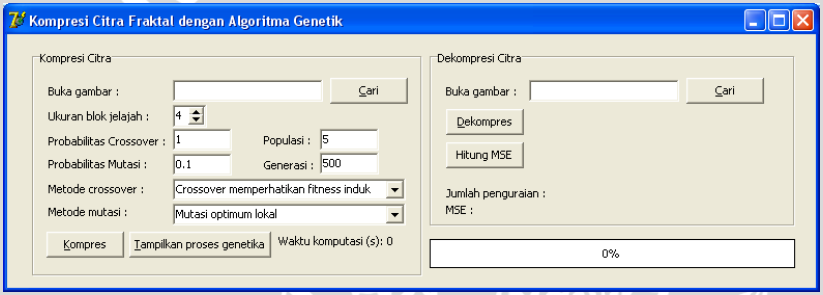
Untuk memeriksa tingkat kemiripan antara citra asli (sebelum dikompres) dengan citra hasil dekompres dilakukan dengan menghitung nilai akar dari kuadrat selisih jarak seluruh *pixel.* Prosedur yang digunakan ditunjukkan oleh Source Code 4.18

```
MSE :=0;
for y:= 0 to frPict.Image1.Picture.Height -1 do
begin
  for x:=0 to frPict.Image1.Picture.Width -1 do
  begin
    MSE := MSE +
         Power(getIntensitas(frPict.Image1,x,y)-
         getIntensitas(frDecomp.Image1,x,y),2);
  end;
end;
MSE :=
   Power(MSE,0.5)/(frPict.Image1.Picture.Height*fr
   Pict.Image1.Picture.Width);
```

Source Code 4.18 Source code perhitungan MSE

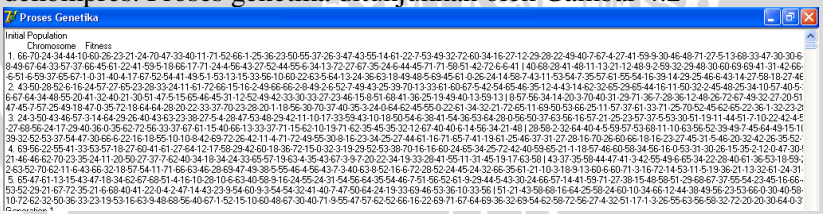
4.2 Penerapan Aplikasi

Tampilan awal dari program ditunjukkan oleh Gambar 4.1. Pada Gambar 4.1 ditunjukkan *form* untuk melakukan proses genetika. Di mana sebelum melakukan kompresi citra dengan algoritma genetika harus dimasukkan dulu citra yang akan dikompres dan ukuran blok jelajah, kemudian pengguna bisa melakukan pengaturan parameter-parameter yang terlibat dalam proses genetika, diantaranya *pc, pm,* jumlah populasi, jumlah generasi maksimal, metode perkawinan silang dan metode mutasi.



Gambar 4.1 Tampilan awal aplikasi

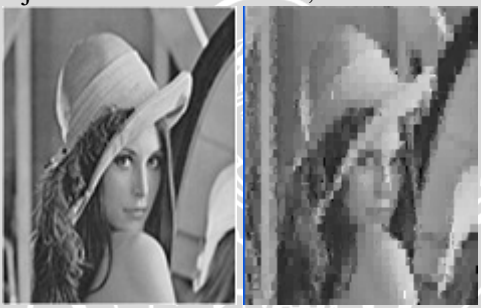
Setelah proses genetika dilakukan, akan terbentuk kumpulan transformasi yang bisa disimpan ke dalam berkas sebagai citra hasil dekompres. Proses genetika ditunjukkan oleh Gambar 4.2



Gambar 4.2 Proses Genetika

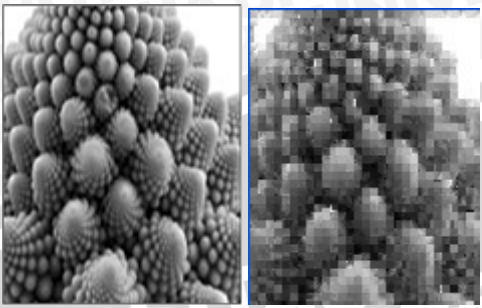
Transformasi terbaik diperoleh dari individu dengan *fitness* terbaik pada generasi terakhir. Karena nilai *fitness* yang diperoleh sangat tergantung pada banyaknya generasi, maka percobaan dilakukan sampai konvergen atau nilai *fitness* terbaik tidak mengalami perubahan sampai seratus kali. Namun jika nilai *fitness*

belum konvergen sampai jumlah generasi mencapai 500, maka iterasi akan dihentikan, karena waktu yang diperlukan akan semakin lama. Dari hasil percobaan terhadap tiga citra yaitu lena1.bmp, NES1.bmp dan flower1.bmp diperoleh nilai *fitness* terbaik untuk masing-masing citra yaitu 0.001023, 0.000818, 0.000611 dan nilai *drms* untuk masing-masing citra yaitu 976.4298, 1220.886, 1634.989. Ketiga citra tersebut diuji coba dengan tiga metode *crossover* dan tiga metode mutasi. Hasil *fitness* terbaik dari ketiga citra tersebut diperoleh dengan melakukan metode *crossover* yang memperhatikan nilai *fitness* masing-masing induk dan metode mutasi dengan pencarian nilai optimum lokal dari tetangga gen yang terkena mutasi. Hasil dekompres citra Lena1, NES1 dan flower1 setelah dikompres ditunjukkan oleh Gambar 4.3, 4.4 dan 4.5



Gambar 4.3 Citra Lena1 sebelum dan setelah dikompresi

Gambar 4.4 Citra NES1 sebelum dan setelah dikompresi



Gambar 4.5 Citra flower1 sebelum dan setelah dikompresi

Citra Lena1.bmp setelah dikompresi memiliki ukuran 13 KB, lebih kecil daripada ukuran semula yaitu 52 KB. Sedangkan nilai MSE yaitu 0.161584. Citra NES1.bmp setelah dikompresi memiliki ukuran 14 KB dan lebih kecil daripada ukuran awal yaitu 53 KB. Sedangkan MSE yaitu 0.303563. Citra flower1.bmp dengan ukuran 53 KB, setelah dikompresi memiliki ukuran 14 KB dengan MSE sebesar 0.245303. Karena citra yang digunakan adalah citra grayscale maka untuk mencari rasio hanya dibandingkan dengan satu komponen saja dari citra asli(ukuran citra asli dibagi tiga). Sehingga rasio kompresi untuk citra Lena1 adalah 75.01%, sedangkan untuk citra NES1 dan flower1 sebesar 79.24%. Jika ukuran blok jelajah diperbesar, maka rasio akan menjadi lebih kecil, namun kualitas citra akan menjadi lebih buruk.

4.3 Analisa Hasil

Untuk memaksimalkan hasil kompresi, maka digunakan berbagai metode *crossover* dan mutasi. Nilai *fitness* yang diperoleh sangat tergantung pada metode yang dipakai. Selain tergantung pada metode, nilai *fitness* yang diperoleh juga tergantung pada probabilitas *crossover* dan probabilitas mutasi. Probabilitas *crossover* akan menentukan jumlah anak yang dihasilkan, sedangkan probabilitas mutasi menentukan peluang suatu individu termutasi. Mutasi sangat penting karena dapat mencegah terjadinya konvergensi dini. Dalam hal ini jika probabilitas mutasi terlalu rendah dapat menimbulkan gen-gen yang berpotensi tidak ditinjau. Sedangkan apabila probabilitas mutasi terlalu tinggi, maka anak hasil mutasi yang dihasilkan bisa menjadi sangat berbeda dengan induk.

Dalam uji coba nilai probabilitas mutasi yang digunakan adalah 0.1 sedangkan nilai probabilitas *crossover* yang digunakan yaitu 0.5, 0.6, 0.7, 0.8 0.9 dan 1.0. Jumlah generasi maksimal adalah 500 generasi, jumlah individu di dalam setiap tahap adalah 5, sedangkan ukuran blok jelajah 4. Citra uji yang dianalisa adalah citra Lena1.bmp yang berukuran 117 x 149 *pixel* dan ukuran *file* sebesar 52 KB. Data hasil uji coba yang telah diurutkan berdasarkan nilai *drms* dapat dilihat pada Tabel 4.1.

Dari Tabel 4.1 dapat dilihat bahwa nilai rata-rata *drms* terbaik diperoleh dengan menggunakan metode *crossover* yang memperhatikan *fitness* tiap gen induk dan mutasi dengan pencarian optimum lokal yaitu sebesar 951.9091. Dari uji coba juga dapat diketahui bahwa perpaduan antara metode *crossover* yang memperhatikan *fitness* induk dan mutasi pencarian optimum lokal menghasilkan *fitness* terbaik untuk semua probabilitas *crossover* jika dibandingkan dengan perpaduan metode yang lainnya. Selain itu rata-rata nilai *drms* yang dihasilkan untuk perpaduan metode *crossover* yang memperhatikan *fitness* tiap gen induk dan mutasi dengan pencarian optimum lokal hampir sama dengan nilai rata-rata *drms* minimum yang dihasilkan, hal ini berarti individu yang dihasilkan dari perpaduan kedua metode tersebut memiliki nilai *fitness* yang hampir sama baik. Hal ini dikarenakan individu yang dihasilkan dari perkawinan silang merupakan individu yang terbaik yaitu individu yang dibentuk dari pasangan gen induk yang mempunyai *fitness* lebih tinggi. Sedangkan mutasi dilakukan dengan mencari optimum lokal daerah tetangga gen-gen yang termutasi, sehingga *fitness* dapat dinaikkan.

Dari sembilan macam perpaduan metode tersebut, yang memerlukan rata-rata waktu paling lama untuk proses eksekusinya adalah perpaduan metode *crossover* satu titik potong biasa dan mutasi dengan pencarian optimum lokal dengan probabilitas *crossover* 1.0 yaitu 25.6 detik. Jika ditelusuri, rata-rata waktu yang besar yaitu di atas 20 detik selalu ditemui jika menggunakan metode *crossover* satu titik potong biasa. Hal ini disebabkan karena nilai *fitness* konvergen pada saat nilai generasi besar. Sedangkan waktu tercepat diperoleh dengan menggunakan metode *crossover* yang memperhatikan nilai *fitness* tiap gen induk dan mutasi penggantian pada probabilitas *crossover* 0.5 yaitu selama 3 detik. Sedangkan

waktu komputasi yang cepat secara umum diperoleh jika menggunakan metode *crossover* satu titik potong pembalikan pengambilan absis dan ordinat. Hal ini menunjukkan bahwa metode *crossover* satu titik potong pembalikan pengambilan absis dan ordinat tidak menghasilkan kromosom yang lebih bervariasi jika dibandingkan dengan metode *crossover* satu titik potong biasa.

Tabel 4.1. Tabel uji coba citra Lena1.bmp dengan berbagai metode *crossover* dan mutasi serta probabilitas *crossover*

Metode	Avg Min drms	Avg Drms	Avg Max Fitness	Avg Fitness	Avg Rasio (%)	Avg Waktu (s)	Avg MSE
c3m3p5	951.9091	952.0604	0.00105	0.00105	75	10.7	0.158839
c3m3p8	954.8529	954.9946	0.001047	0.001047	75	21.6	0.160592
c3m3p10	963.3896	963.5688	0.001038	0.001037	75	23.9	0.161622
c3m3p7	965.0301	965.2381	0.001036	0.001035	75	18.7	0.160963
c3m3p6	974.7385	974.8204	0.001025	0.001025	75	18.3	0.162707
c3m3p9	976.1725	976.325	0.001024	0.001024	75	18.3	0.162439
c3m1p5	990.8282	991.0066	0.001009	0.001008	75	12.7	0.162918
c3m1p10	992.5147	992.6931	0.001007	0.001007	75	33.3	0.163147
c3m2p5	993.5107	993.8018	0.001006	0.001006	75	3	0.163729
c3m1p9	994.3634	994.5712	0.001005	0.001005	75	26.1	0.177234
c3m1p6	997.9504	998.0531	0.001001	0.001001	75	19.3	0.164009
c3m1p7	1001.467	1001.626	0.000998	0.000998	75	26.2	0.178071
c3m2p9	1002.749	1003.108	0.000997	98.66142	75	10.1	0.16347
c3m1p8	1003.737	1003.797	0.000996	0.000995	75	26	0.177044
c3m2p6	1005.449	1005.681	0.000994	0.000994	75	6	0.164302
c3m2p10	1010.35	1010.745	0.000989	0.000989	75	12.5	0.163956
c3m2p8	1010.87	1011.284	0.000988	0.000988	75	7.6	0.164522
c3m2p7	1014.779	1015.037	0.000985	0.000985	75	7.9	0.165067
c1m2p10	1074.381	1074.466	0.00093	0.00093	75	31.2	0.170381
c1m2p8	1078.366	1078.461	0.000927	0.000926	75	23.5	0.169374
c1m2p7	1081.693	1081.764	0.000924	0.000924	75	22.4	0.169531
c1m2p9	1082.307	1082.383	0.000923	0.000923	75	22.5	0.169858

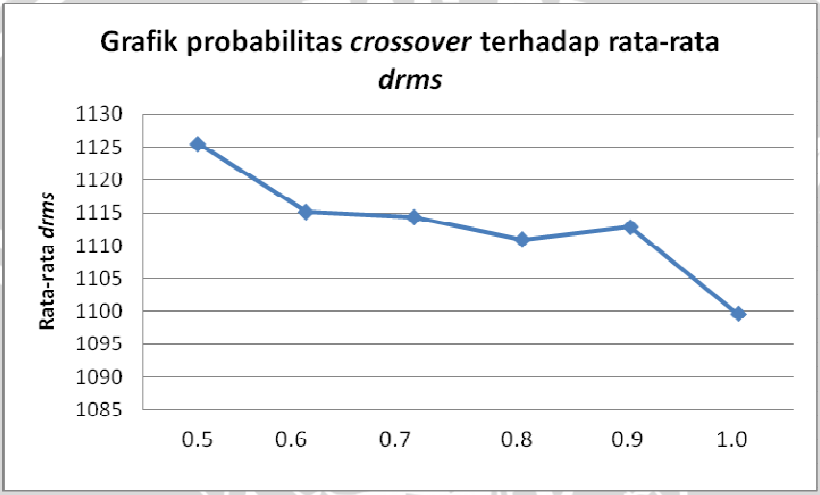
c1m2p6	1083.024	1083.065	0.000923	0.000922	75	21.9	0.171501
c1m3p10	1096.609	1096.972	0.000911	0.000911	75	35.6	0.176917
c1m3p7	1097.49	1098.031	0.00091	0.00091	75	24.1	0.176889
c1m3p9	1098.323	1098.774	0.00091	0.000909	75	24.5	0.176755
c1m2p5	1098.88	1099.089	0.000909	0.000909	75	11.5	0.17363
c1m1p10	1099.565	1099.89	0.000909	0.000908	75	34.8	0.174461
c1m3p6	1100.504	1101.045	0.000908	0.000907	75	23.8	0.176643
c1m3p8	1101.073	1101.617	0.000907	0.000907	75	24.7	0.176925
c1m3p5	1102.982	1104.017	0.000906	0.000905	75	11.9	0.17664
c1m1p8	1110.926	1110.982	0.000899	0.000899	75	23.9	0.176758
c1m1p9	1112.831	1112.912	0.000898	0.000898	75	23.6	0.176711
c1m1p7	1114.332	1114.436	0.000897	0.000897	75	24.1	0.17662
c1m1p6	1115.145	1115.29	0.000896	0.000896	75	24.9	0.176239
c2m3p10	1117.939	1120.931	0.000894	0.000891	75	13.6	0.178517
c2m3p9	1117.945	1122.984	0.000894	0.00089	75	8	0.178672
c2m3p8	1120.159	1123.856	0.000892	0.000889	75	11.9	0.178582
c2m3p6	1120.356	1123.862	0.000892	0.000889	75	10.7	0.179426
c2m3p7	1121.978	1125.52	0.00089	0.000888	75	9.1	0.179478
c2m3p5	1122.543	1126.44	0.00089	0.000887	75	4.5	0.178927
c2m2p10	1123.982	1126.755	0.000889	0.000887	75	12.2	0.177291
c1m1p5	1125.505	1125.63	0.000888	0.000888	75	12.4	0.178957
c2m1p10	1126.767	1129.495	0.000887	0.000885	75	16	0.17835
c2m1p9	1128.975	1131.684	0.000885	0.000883	75	10.4	0.176666
c2m1p8	1129.232	1131.975	0.000885	0.000883	75	8.9	0.176211
c2m2p6	1129.421	1132.318	0.000885	0.000882	75	10.9	0.178071
c2m2p8	1129.853	1131.848	0.000884	0.000883	75	10.1	0.174139
c2m2p5	1129.98	1133.503	0.000884	0.000881	75	5.3	0.178294
c2m1p6	1130.093	1133.826	0.000884	0.000881	75	9.3	0.178176
c2m2p9	1130.457	1133.455	0.000884	0.000881	75	9.5	0.174385
c2m1p5	1131.198	1134.989	0.000883	0.00088	75	5.1	0.178015
c2m2p7	1131.531	1133.585	0.000883	0.000881	75	9.4	0.174447
c2m1p7	1132.154	1135.164	0.000883	0.00088	75	8.7	0.176954

Keterangan Metode pada Tabel 4.1 :
c1 : Metode *crossover* satu titik potong biasa
c2 : Metode *crossover* satu titik potong dengan pembalikan pengambilan absis dan ordinat dari induk
c3 : Metode *crossover* dengan memperhatikan *fitness* tiap gen induk
m1 : Metode mutasi pertukaran
m2 : Metode mutasi penggantian
m3 : Metode mutasi pencarian optimum lokal
p5 : Probabilitas *crossover* 0.5
p6 : Probabilitas *crossover* 0.6
p7 : Probabilitas *crossover* 0.7
p8 : Probabilitas *crossover* 0.8
p9 : Probabilitas *crossover* 0.9
p10 : Probabilitas *crossover* 1.0

Pengaruh probabilitas *crossover* berbeda-beda untuk setiap perpaduan metode yang dipakai. Pada metode *crossover* satu titik potong biasa dan mutasi pertukaran, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar 1125.505, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami penurunan yang cukup besar menjadi 1115.145, sedangkan pada saat probabilitas *crossover* 0.7 terjadi penurunan nilai *drms* menjadi 1114.332. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan menjadi 1110.926, dan pada saat probabilitas *crossover* dinaikkan menjadi 0.9, nilai *drms* mengalami kenaikan menjadi 1112.831. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami penurunan yang cukup besar menjadi 1099.565. Hal tersebut menunjukkan bahwa peningkatan nilai probabilitas *crossover* mulai dari 0.5 sampai 0.8 berpengaruh pada penurunan nilai *drms*. Grafik pengaruh probabilitas *crossover* ditunjukkan oleh Gambar 4.6.

Pada metode *crossover* satu titik potong biasa dan mutasi penggantian, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar 1098.88, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami penurunan yang besar menjadi 1083.024, sedangkan pada saat probabilitas *crossover* 0.7 terjadi penurunan nilai *drms* menjadi 1081.693. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan menjadi 1078.366, dan pada saat probabilitas *crossover* dinaikkan

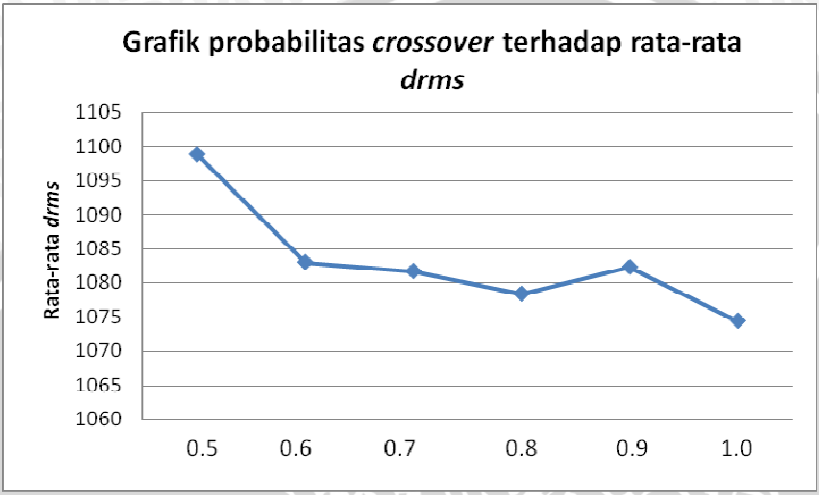
menjadi 0.9, nilai *drms* mengalami kenaikan menjadi 1082.307. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami penurunan yang sangat kecil menjadi 1074.381. Hal tersebut menunjukkan bahwa peningkatan nilai probabilitas *crossover* mulai dari 0.5 sampai 0.8 berpengaruh pada penurunan nilai *drms*. Grafik pengaruh probabilitas *crossover* ditunjukkan oleh Gambar 4.7.



Gambar 4.6 Grafik pengaruh probabilitas *crossover* terhadap rata-rata *drms* pada metode *crossover* satu titik potong biasa dan mutasi pertukaran

Pada metode *crossover* satu titik potong biasa dan mutasi pencarian optimum lokal, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar 1102.982, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami penurunan kecil menjadi 1100.504, sedangkan pada saat probabilitas *crossover* 0.7 terjadi penurunan nilai *drms* menjadi 1097.49. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami kenaikan menjadi 1101.073, dan pada saat probabilitas *crossover* dinaikkan menjadi 0.9, nilai *drms* mengalami penurunan menjadi 1098.323. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami penurunan menjadi 1096.609. Hal tersebut menunjukkan bahwa peningkatan ataupun penurunan

probabilitas *crossover* tidak berpengaruh pada nilai *drms* untuk metode ini.

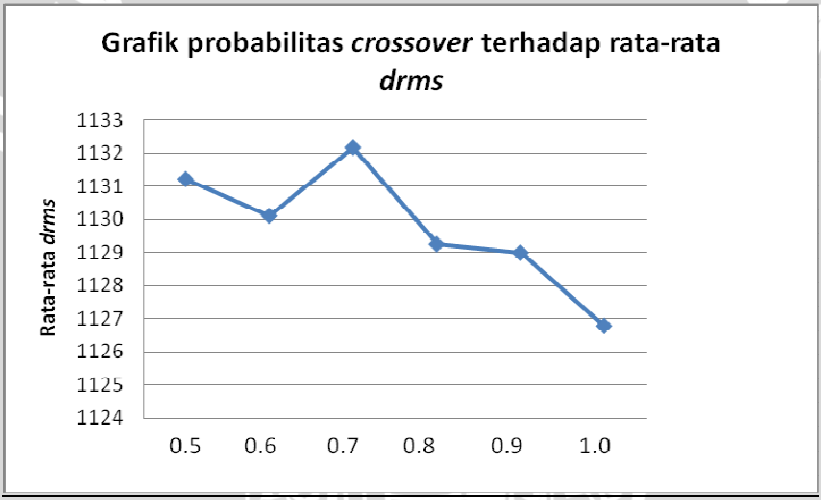


Gambar 4.7 Grafik pengaruh probabilitas *crossover* terhadap rata-rata *drms* pada metode *crossover* satu titik potong biasa dan mutasi penggantian

Pada metode *crossover* satu titik potong dengan pembalikan pengambilan absis dan ordinat dari induk dan mutasi pertukaran, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar 1131.198, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami penurunan menjadi 1130.093, sedangkan pada saat probabilitas *crossover* 0.7 terjadi kenaikan nilai *drms* menjadi 1132.154. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan menjadi 1129.232, dan pada saat probabilitas *crossover* dinaikkan menjadi 0.9, nilai *drms* mengalami penurunan menjadi 1128.975. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami kenaikan menjadi 1126.767. Hal tersebut menunjukkan bahwa nilai *drms* yang lebih baik diperoleh dengan menaikkan probabilitas *crossover* di atas 0.7 untuk metode ini. Grafik pengaruh probabilitas *crossover* ditunjukkan oleh Gambar 4.8.

Pada metode *crossover* satu titik potong dengan pembalikan pengambilan absis dan ordinat dari induk dan mutasi penggantian, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar

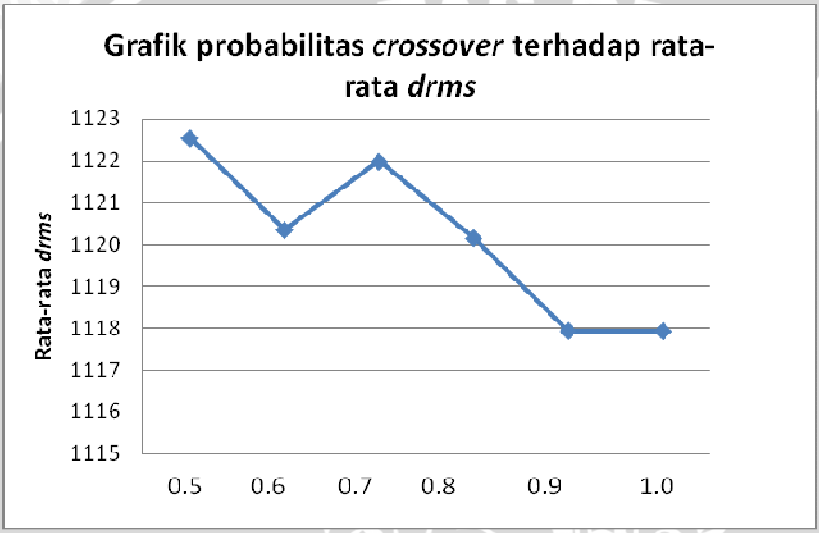
1129.98, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami penurunan kecil menjadi 1129.421, sedangkan pada saat probabilitas *crossover* 0.7 terjadi kenaikan nilai *drms* menjadi 1131.531. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan menjadi 1129.853, dan pada saat probabilitas *crossover* dinaikkan menjadi 0.9, nilai *drms* mengalami kenaikan menjadi 1130.457. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami penurunan yang cukup besar menjadi 1123.982. Hal tersebut menunjukkan bahwa peningkatan ataupun penurunan probabilitas *crossover* tidak berpengaruh pada nilai *drms* untuk metode ini.



Gambar 4.8 Grafik pengaruh probabilitas *crossover* terhadap rata-rata *drms* pada metode *crossover* satu titik potong pembalikan pengambilan absis dan ordinat dan mutasi pertukaran

Pada metode *crossover* satu titik potong dengan pembalikan pengambilan absis dan ordinat dari induk dan mutasi pencarian optimum lokal, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar 1122.543, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami penurunan kecil menjadi 1120.356, sedangkan pada saat probabilitas *crossover* 0.7 terjadi kenaikan nilai *drms* menjadi 1121.978. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan menjadi 1120.159, dan pada saat probabilitas *crossover*

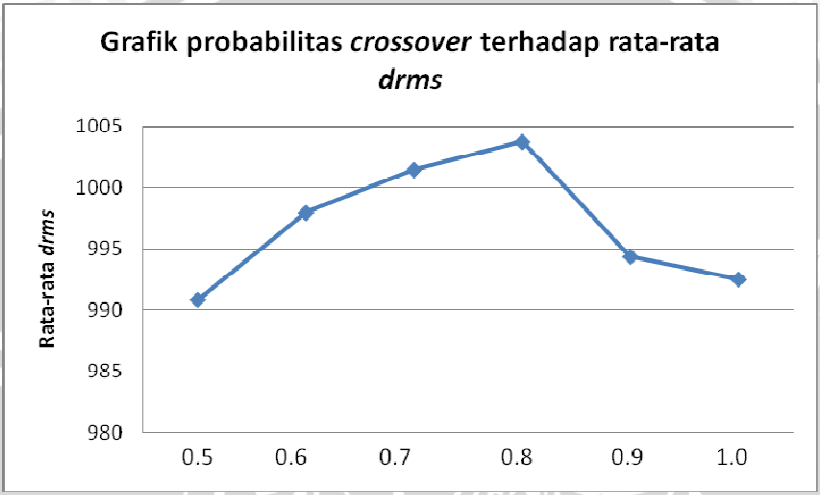
dinaikkan menjadi 0.9, nilai *drms* mengalami penurunan menjadi 1117.945. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami penurunan yang sangat kecil menjadi 1117.939. Hal tersebut menunjukkan bahwa nilai *drms* yang lebih baik diperoleh dengan menaikkan probabilitas *crossover* di atas 0.7 untuk metode ini. Grafik pengaruh probabilitas *crossover* ditunjukkan oleh Gambar 4.9.



Gambar 4.9 Grafik pengaruh probabilitas *crossover* terhadap rata-rata *drms* pada metode *crossover* satu titik potong pembalikan pengambilan absis dan ordinat dan mutasi pencarian optimum lokal

Pada metode *crossover* dengan memperhatikan *fitness* gen induk dan mutasi pertukaran, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar 990.8282, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami kenaikan yang cukup besar menjadi 997.9504, sedangkan pada saat probabilitas *crossover* 0.7 terjadi kenaikan nilai *drms* menjadi 1001.467. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan menjadi 1003.737, dan pada saat probabilitas *crossover* dinaikkan menjadi 0.9, nilai *drms* mengalami penurunan menjadi 994.3634. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami penurunan yang cukup besar menjadi 992.5147. Hal tersebut menunjukkan bahwa peningkatan

nilai probabilitas *crossover* mulai dari 0.5 sampai 0.8 berpengaruh pada kenaikan nilai *drms*, sedangkan probabilitas *crossover* di atas 0.8 menghasilkan nilai *drms* yang semakin kecil. Grafik pengaruh probabilitas *crossover* ditunjukkan oleh Gambar 4.10.



Gambar 4.10 Grafik pengaruh probabilitas *crossover* terhadap rata-rata *drms* pada metode *crossover* dengan memperhatikan *fitness* induk dan mutasi pertukaran

Pada metode *crossover* dengan memperhatikan *fitness* gen induk dan mutasi penggantian, pada saat probabilitas *crossover* sebesar 0.5, nilai *drms* sebesar 993.5107, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami kenaikan menjadi 1005.449, sedangkan pada saat probabilitas *crossover* 0.7 terjadi kenaikan yang cukup besar nilai *drms* menjadi 1014.779. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan menjadi 1010.87, dan pada saat probabilitas *crossover* dinaikkan menjadi 0.9, nilai *drms* mengalami penurunan yang cukup besar menjadi 1002.749. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami kenaikan yang cukup besar menjadi 1010.35. Hal tersebut menunjukkan bahwa peningkatan ataupun penurunan probabilitas *crossover* tidak berpengaruh pada nilai *drms* untuk metode ini.

Pada metode *crossover* dengan memperhatikan *fitness* gen induk dan mutasi pencarian optimum lokal, pada saat probabilitas

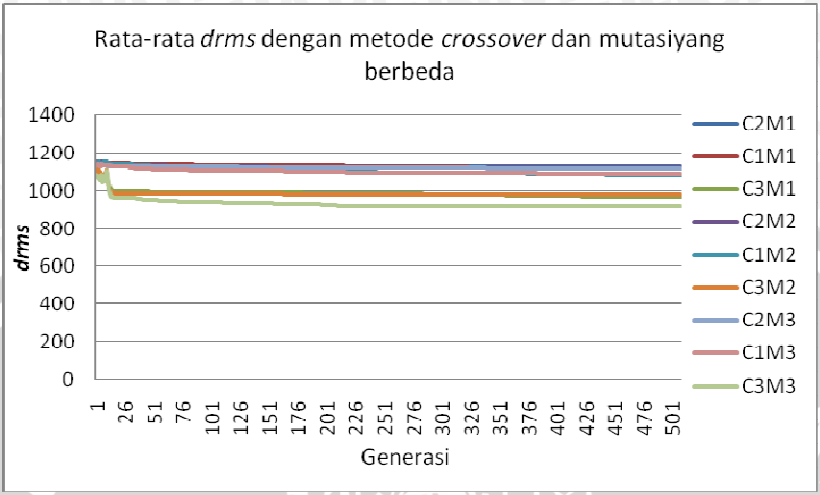
crossover sebesar 0.5, nilai *drms* sebesar 951.9091, kemudian saat probabilitas *crossover* dinaikkan menjadi 0.6 nilai *drms* mengalami kenaikan yang besar menjadi 974.7385, sedangkan pada saat probabilitas *crossover* 0.7 terjadi penurunan yang cukup besar nilai *drms* menjadi 965.0301. Ketika nilai probabilitas *crossover* dinaikkan menjadi 0.8, nilai *drms* mengalami penurunan yang cukup menjadi 954.8529, dan pada saat probabilitas *crossover* dinaikkan menjadi 0.9, nilai *drms* mengalami kenaikan yang cukup besar menjadi 976.1725. Kemudian pada saat probabilitas *crossover* dinaikkan menjadi 1.0, nilai *drms* mengalami penurunan yang cukup besar menjadi 963.3896. Hal tersebut menunjukkan bahwa peningkatan ataupun penurunan probabilitas *crossover* tidak berpengaruh pada nilai *drms* untuk metode ini.

Hasil uji coba juga menunjukkan bahwa penggunaan probabilitas yang mampu menghasilkan nilai *drms* yang baik untuk perpaduan metode yang menggunakan metode *crossover* dengan memperhatikan *fitness* tiap gen induk adalah 0.5. Sedangkan untuk metode lainnya yaitu 1.0.

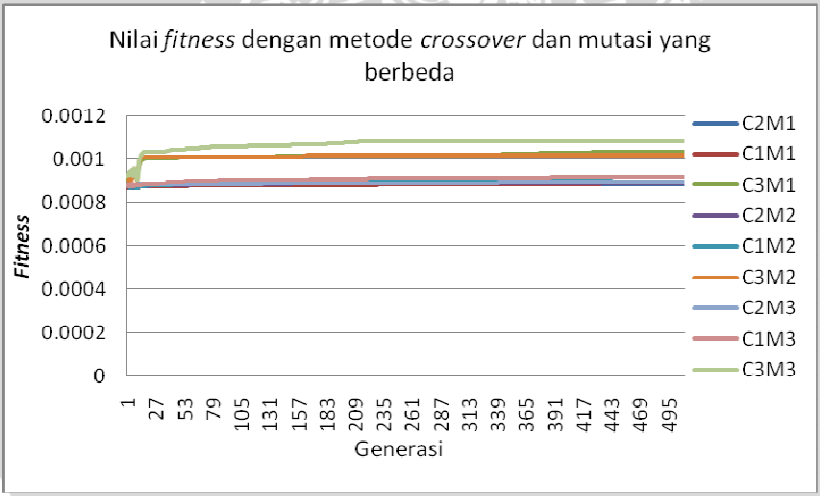
Untuk mengetahui kinerja masing-masing perpaduan metode secara lebih detail, bisa dilihat pada Gambar 4.11 dan 4.12.

Grafik pada Gambar 4.11 diperoleh dengan menjalankan sembilan perpaduan metode *crossover* dan mutasi pada probabilitas *crossover* 0.8. Dari grafik diperlihatkan bahwa metode yang mampu menghasilkan nilai *drms* terbaik adalah metode *crossover* dengan memperhatikan *fitness* tiap gen induk dan mutasi pencarian optimum lokal.

Grafik pada Gambar 4.12 diperoleh dengan menjalankan sembilan perpaduan metode *crossover* dan mutasi pada probabilitas *crossover* 0.8. Dari grafik diperlihatkan bahwa metode yang mampu menghasilkan nilai *fitness* terbaik adalah metode *crossover* dengan memperhatikan *fitness* tiap gen induk dan mutasi pencarian optimum lokal.



Gambar 4.11 Grafik perbandingan rata-rata *drms* terhadap generasi dengan metode *crossover* dan mutasi yang berbeda.



Gambar 4.12 Grafik perbandingan rata-rata *fitness* terhadap generasi dengan metode *crossover* dan mutasi yang berbeda.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil penyusunan Skripsi ini adalah

1. Algoritma genetika merupakan salah satu teknik pencarian heuristik yang mampu megefisienkan waktu pencocokkan blok jelajah dengan blok ranah pada kompresi citra fraktal, namun menghasilkan kualitas citra yang kurang baik.
2. Dari tiga metode *crossover* dan tiga metode mutasi yang diterapkan, metode yang mampu menghasilkan nilai *fitness* terbaik adalah metode *crossover* dengan memperhatikan *fitness* tiap gen induk dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.5, rasio yang mapu dihasilkan adalah 75.01%, waktu kompresi rata-rata10.7 detik dengan MSE sebesar 0.158839.
3. Nilai probabilitas *crossover* yang diterapkan berbeda untuk setiap metode. Dari sembilan kombinasi metode yang digunakan, nilai probabilitas *crossover* yang baik untuk kombinasi dengan menggunakan metode *crossover* dengan memperhatikan *fitness* tiap gen induk adalah 0.5, sedangkan untuk metode lainnya adalah 1.0.
4. Waktu terbaik diperoleh dengan kombinasi metode *crossover* dengan memperhatikan *fitness* induk dan mutasi penggantian dengan probabilitas *crossover* 0.5 yaitu selama 3 detik, sedangkan waktu yang paling lama dibutuhkan oleh metode *crossover* satu titik potong biasa dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 1.0

5.2 Saran

Untuk pengembangan aplikasi selanjutnya diharapkan :

1. Kualitas citra hasil kompresi bias lebih ditungkatkan, bisa dengan mencoba metode perkawinan silang dan mutasi yang lain.
2. Penyimpanan parameter-parameter transformasi *affine* diharapkan bisa menggunakan metode yang lebih baik

sehingga ukuran penyimpanan bisa lebih baik (rasio lebih kecil).

3. Jika terdapat bagian tepi citra yang masih tersisa (karena ukuran lebih kecil daripada blok jelajah) tetap dihitung, sehingga tidak ada *pixel* yang hilang setelah proses dekompresi.
4. Citra tidak hanya bisa didekompres pada ukuran asli, namun juga bisa pada ukuran lain, sesuai inputan *user*.

DAFTAR PUSTAKA

Anonymous. 1997. *An Introduction to Fractal Image Compression*. Texas Instruments Europe

Hermawanto, Denny. 2006. *Representasi Kromosom Algoritma Genetika dalam Bentuk Biner*, http://dennyhermawanto.webhop.org. Tanggal akses : 24 Mei 2008.

Kuswadi, Son. 2007. *Kendali Cerdas, Teori dan Praktisnya.* Andi : Yogyjakarta

Kusumadewi, Sri & Purnomo, Hari. 2005. *Penyelesaian Masalah Optimasi dengan Teknik-teknik Heuristik.* Graha Ilmu : Yogyakarta

Munir, Rinaldi. 2004. *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Informatika: Bandung.

Nurjaya, Wahyu. 2006. *Analisis Proses Word Matching Problem Menggunakan Algoritma Genetika*. Universitas Komputer Indonesia.

Obitko, M. 1998. *Genetic Algorithm*, http://cs.felk.cvut.cz/~xobitko/ga/. Tanggal akses: 24 Mei 2008.

Obitko, M. 1998. *Introduction to Genetic Algorithm*, http://cs.felk.cvut.cz/~xobitko/ga/. Tanggal akses: 24 Mei 2008.

Vences, Lucia, Rudomin, Isaac. 1997. *Genetic Algorithms for Fractal Image and Image Sequence Compression*. Instituto Tecnologico de Estudios Superiores de Monterrey, Campus Estado de Mexico: Mexico.

Xi, Lifeng, Zhang, Liangbin. 2007. *A Study of Fractal Image Compression Based on an Improved Genetic Algorithm.* Institute of Mathematics, Zhejiang Wanli University.

LAMPIRAN

Hasil uji coba kompresi citra fraktal dengan algoritma genetika terhadap citra Lena1.bmp

Tabel 1 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pertukaran dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1131.803	1131.873	0.000883	0.000883	75	12	0.177944
2	1113.641	1114.097	0.000897	0.000897	75	11	0.176655
3	1132.694	1132.735	0.000882	0.000882	75	12	0.179392
4	1120.309	1120.663	0.000892	0.000892	75	12	0.17948
5	1125.391	1125.417	0.000888	0.000888	75	12	0.177607
6	1138.46	1138.502	0.000878	0.000878	75	12	0.183416
7	1113.964	1114.217	0.000897	0.000897	75	13	0.179972
8	1121.619	1121.619	0.000891	0.000891	75	13	0.178239
9	1134.202	1134.205	0.000881	0.000881	75	13	0.178401
10	1122.966	1122.975	0.00089	0.00089	75	14	0.17846
Rata-rata	**1125.505**	**1125.63**	**0.000888**	**0.000888**	**25**	**12.4**	**0.178957**

Tabel 2 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pertukaran dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1114.957	1115.289	0.000896	0.000896	75	26	0.175556
2	1119.129	1119.132	0.000893	0.000893	75	27	0.177634
3	1118.973	1118.977	0.000893	0.000893	75	26	0.175923
4	1120.353	1120.402	0.000892	0.000892	75	26	0.182353
5	1105.33	1105.358	0.000904	0.000904	75	27	0.172468
6	1110.7	1110.705	0.0009	0.0009	75	23	0.174314
7	1107.024	1107.809	0.000903	0.000902	75	24	0.172638
8	1116.243	1116.279	0.000895	0.000895	75	23	0.178853
9	1121.517	1121.62	0.000891	0.000891	75	24	0.177784
10	1117.223	1117.331	0.000894	0.000894	75	23	0.17487
Rata-rata	**1115.145**	**1115.29**	**0.000896**	**0.000896**	**75**	**24.9**	**0.176239**

Tabel 3 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pertukaran dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1118.733	1118.991	0.000893	0.000893	75	23	0.177283
2	1115.517	1115.523	0.000896	0.000896	75	24	0.175323
3	1118.945	1118.948	0.000893	0.000893	75	24	0.177318
4	1115.386	1115.388	0.000896	0.000896	75	24	0.177371
5	1126.486	1126.553	0.000887	0.000887	75	25	0.176601
6	1107.08	1107.113	0.000902	0.000902	75	26	0.177178
7	1108.113	1108.343	0.000902	0.000901	75	24	0.176037
8	1107.027	1107.102	0.000903	0.000902	75	23	0.177487
9	1121.5	1121.647	0.000891	0.000891	75	24	0.176385
10	1104.533	1104.755	0.000905	0.000904	75	24	0.175215
Rata-rata	**1114.332**	**1114.436**	**0.000897**	**0.000897**	**75**	**24.1**	**0.17662**

Tabel 4 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pertukaran dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1121.743	1121.813	0.000891	0.000891	75	24	0.176048
2	1110.009	1110.016	0.0009	0.0009	75	25	0.175543
3	1107.08	1107.08	0.000902	0.000902	75	24	0.178234
4	1113.904	1113.904	0.000897	0.000897	75	23	0.177695
5	1106.167	1106.197	0.000903	0.000903	75	24	0.175574
6	1116.833	1116.917	0.000895	0.000895	75	24	0.177334
7	1109.629	1109.636	0.0009	0.0009	75	23	0.178957
8	1114.752	1114.958	0.000896	0.000896	75	24	0.174217
9	1104.327	1104.387	0.000905	0.000905	75	23	0.176647
10	1104.818	1104.915	0.000904	0.000904	75	25	0.17733
Rata-rata	**1110.926**	**1110.982**	**0.000899**	**0.000899**	**75**	**23.9**	**0.176758**

Tabel 5 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pertukaran dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1116.631	1116.763	0.000895	0.000895	75	24	0.177557
2	1107.494	1107.52	0.000902	0.000902	75	24	0.176311
3	1125.977	1125.99	0.000887	0.000887	75	23	0.176311
4	1108.711	1108.856	0.000901	0.000901	75	23	0.175622
5	1118.647	1118.795	0.000893	0.000893	75	23	0.175622
6	1110.313	1110.468	0.0009	0.0009	75	24	0.178808
7	1110.954	1111.019	0.000899	0.000899	75	23	0.176462
8	1107.66	1107.737	0.000902	0.000902	75	24	0.176758
9	1099.759	1099.803	0.000908	0.000908	75	24	0.177098
10	1122.167	1122.17	0.00089	0.00089	75	24	0.176559
Rata-rata	**1112.831**	**1112.912**	**0.000898**	**0.000898**	**75**	**23.6**	**0.176711**

Tabel 6 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pertukaran dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1091.194	1091.31	0.000916	0.000915	75	35	0.175076
2	1099.576	1099.87	0.000909	0.000908	75	35	0.174902
3	1107.188	1107.195	0.000902	0.000902	75	34	0.176532
4	1102.907	1102.907	0.000906	0.000906	75	35	0.175259
5	1105.799	1108.305	0.000904	0.000901	75	35	0.175294
6	1100.679	1100.698	0.000908	0.000908	75	35	0.172948
7	1110.01	1110.01	0.0009	0.0009	75	34	0.173221
8	1083.689	1083.832	0.000922	0.000922	75	35	0.174731
9	1101.849	1101.922	0.000907	0.000907	75	35	0.174119
10	1092.763	1092.853	0.000914	0.000914	75	35	0.172525
Rata-rata	**1099.565**	**1099.89**	**0.000909**	**0.000908**	**75**	**34.8**	**0.174461**

Tabel 7 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Avg Rasio (%)	Avg Waktu (s)	MSE
1	1129.917	1135.463	0.000884	0.00088	75	5	0.178037
2	1127.316	1131.919	0.000886	0.000883	75	5	0.178666
3	1136.465	1140.46	0.000879	0.000876	75	6	0.178615
4	1130.057	1134.505	0.000884	0.000881	75	5	0.177547
5	1127.905	1132.587	0.000886	0.000882	75	3	0.18149
6	1132.431	1137.97	0.000882	0.000878	75	4	0.178066
7	1135.687	1136.444	0.00088	0.000879	75	3	0.177308
8	1133.093	1134.647	0.000882	0.000881	75	5	0.177979
9	1127.066	1130.146	0.000886	0.000884	75	9	0.176578
10	1132.045	1135.748	0.000883	0.00088	75	6	0.175867
Rata-rata	**1131.198**	**1134.989**	**0.000883**	**0.00088**	**75**	**5.1**	**0.178015**

Tabel 8 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1133.111	1135.065	0.000882	0.00088	75	9	0.181199
2	1129.542	1133.391	0.000885	0.000882	75	7	0.177014
3	1128.536	1131.487	0.000885	0.000883	75	18	0.17632
4	1123.472	1131.234	0.000889	0.000883	75	9	0.175833
5	1128.75	1131.97	0.000885	0.000883	75	10	0.17666
6	1126.355	1134.143	0.000887	0.000881	75	7	0.175253
7	1130.983	1133.026	0.000883	0.000882	75	11	0.178876
8	1134.395	1137.537	0.000881	0.000878	75	8	0.182404
9	1132.254	1136.113	0.000882	0.000879	75	6	0.177618
10	1133.534	1134.296	0.000881	0.000881	75	8	0.180584
Rata-rata	**1130.093**	**1133.826**	**0.000884**	**0.000881**	**75**	**9.3**	**0.178176**

Tabel 9 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1131.227	1133.89	0.000883	0.000881	75	12	0.176099
2	1133.961	1135.823	0.000881	0.00088	75	8	0.176791
3	1132.118	1132.989	0.000883	0.000882	75	13	0.176306
4	1130.861	1136.79	0.000884	0.000879	75	6	0.176347
5	1130.723	1133.788	0.000884	0.000881	75	10	0.176578
6	1123.732	1128.703	0.000889	0.000885	75	12	0.176651
7	1135.232	1137.973	0.00088	0.000878	75	7	0.178586
8	1136.691	1138.38	0.000879	0.000878	75	6	0.177187
9	1132.65	1135.145	0.000882	0.00088	75	8	0.177187
10	1134.349	1138.155	0.000881	0.000878	75	5	0.177804
Rata-rata	**1132.154**	**1135.164**	**0.000883**	**0.00088**	**75**	**8.7**	**0.176954**

Tabel 10 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1126.971	1131.213	0.000887	0.000883	75	5	0.175562
2	1129.486	1132.026	0.000885	0.000883	75	13	0.175337
3	1138.295	1139.317	0.000878	0.000877	75	8	0.177049
4	1121.566	1124.636	0.000891	0.000888	75	10	0.176158
5	1134.804	1137.292	0.00088	0.000879	75	8	0.177109
6	1127.447	1132.306	0.000886	0.000882	75	8	0.176624
7	1129.167	1130.605	0.000885	0.000884	75	9	0.176235
8	1125.361	1129.384	0.000888	0.000885	75	9	0.176261
9	1130.449	1131.703	0.000884	0.000883	75	11	0.17719
10	1128.771	1131.268	0.000885	0.000883	75	8	0.174585
Rata-rata	**1129.232**	**1131.975**	**0.000885**	**0.000883**	**75**	**8.9**	**0.176211**

Tabel 11 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1127.334	1130.82	0.000886	0.000884	75	9	0.176486
2	1134.27	1138.956	0.000881	0.000877	75	10	0.17846
3	1132.3	1133.186	0.000882	0.000882	75	10	0.176182
4	1130.583	1134.596	0.000884	0.000881	75	7	0.178674
5	1129.018	1130.831	0.000885	0.000884	75	6	0.176188
6	1131.355	1133.231	0.000883	0.000882	75	13	0.176551
7	1126.325	1129.023	0.000887	0.000885	75	13	0.175802
8	1130.885	1132.638	0.000883	0.000882	75	6	0.176091
9	1118.921	1123.311	0.000893	0.000889	75	20	0.176592
10	1128.762	1130.25	0.000885	0.000884	75	10	0.175629
Rata-rata	**1128.975**	**1131.684**	**0.000885**	**0.000883**	**75**	**10.4**	**0.176666**

Tabel 12 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 1.0

	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1125.633	1129.929	0.000888	0.000884	75	14	0.176219
2	1119.601	1123.062	0.000892	0.00089	75	20	0.17679
3	1138.013	1139.143	0.000878	0.000877	75	7	0.179136
4	1123.494	1124.812	0.000889	0.000888	75	30	0.178945
5	1124.609	1126.344	0.000888	0.000887	75	29	0.178315
6	1127.121	1133.147	0.000886	0.000882	75	9	0.178536
7	1124.72	1126.274	0.000888	0.000887	75	17	0.178328
8	1123.322	1125.797	0.000889	0.000887	75	11	0.177165
9	1132.728	1134.675	0.000882	0.000881	75	9	0.18184
10	1128.431	1131.764	0.000885	0.000883	75	14	0.178221
Rata-rata	**1126.767**	**1129.495**	**0.000887**	**0.000885**	**75**	**16**	**0.17835**

Tabel 13 Hasil uji metode *crossover* memperhatikan *fitness* induk dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	990.6682	990.6696	0.001008	0.001008	75	13	0.161329
2	990.7583	990.7738	0.001008	0.001008	75	12	0.165826
3	969.9708	970.1097	0.00103	0.00103	75	12	0.159809
4	993.4679	993.5194	0.001006	0.001006	75	13	0.16184
5	1015.547	1016.029	0.000984	0.000983	75	13	0.164286
6	1024.498	1024.506	0.000975	0.000975	75	13	0.165665
7	961.4448	962.1708	0.001039	0.001038	25	13	0.158937
8	979.5403	979.8048	0.00102	0.00102	75	13	0.161556
9	983.8605	983.9371	0.001015	0.001015	75	13	0.166719
10	998.5269	998.5459	0.001	0.001	75	12	0.163207
Rata-rata	**990.8282**	**991.0066**	**0.001009**	**0.001008**	**75**	**12.7**	**0.162918**

Tabel 14 Hasil uji metode *crossover* memperhatikan *fitness* induk dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	974.9611	974.9962	0.001025	0.001025	75	20	0.161612
2	1022.016	1022.024	0.000978	0.000977	75	19	0.166395
3	967.3246	967.5994	0.001033	0.001032	75	19	0.159048
4	980.9232	980.9261	0.001018	0.001018	75	20	0.165267
5	1020.856	1021.299	0.000979	0.000978	75	19	0.166753
6	989.8848	989.892	0.001009	0.001009	75	19	0.166009
7	1015.702	1015.713	0.000984	0.000984	25	19	0.168421
8	1012.352	1012.405	0.000987	0.000987	75	19	0.164815
9	976.6492	976.7321	0.001023	0.001023	75	20	0.159445
10	1018.835	1018.943	0.000981	0.00098	75	19	0.162324
Rata-rata	**997.9504**	**998.0531**	**0.001001**	**0.001001**	**75**	**19.3**	**0.164009**

Tabel 15 Hasil uji metode *crossover* memperhatikan *fitness* induk dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1011.692	1011.751	0.000987	0.000987	75	25	0.175822
2	1017.878	1017.937	0.000981	0.000981	75	27	0.177594
3	1005.096	1005.209	0.000994	0.000994	75	26	0.180317
4	1014.668	1014.69	0.000985	0.000985	75	26	0.178818
5	962.1835	962.2909	0.001038	0.001038	75	27	0.177434
6	1013.609	1013.642	0.000986	0.000986	75	26	0.178839
7	996.322	997.2131	0.001003	0.001002	25	26	0.179349
8	982.8173	982.8301	0.001016	0.001016	75	27	0.178095
9	998.5667	998.7682	0.001	0.001	75	26	0.175977
10	1011.835	1011.927	0.000987	0.000987	75	26	0.178466
Rata-rata	**1001.467**	**1001.626**	**0.000998**	**0.000998**	**75**	**26.2**	**0.178071**

Tabel 16 Hasil uji metode *crossover* memperhatikan *fitness* induk dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	968.329	968.3992	0.001032	0.001032	75	26	0.177306
2	1020.425	1020.455	0.000979	0.000979	75	26	0.177052
3	976.4236	976.4836	0.001023	0.001023	75	26	0.176842
4	1002.673	1002.685	0.000996	0.000996	75	26	0.17893
5	1010.649	1010.711	0.000988	0.000988	75	26	0.175765
6	1011.618	1011.632	0.000988	0.000988	75	26	0.177339
7	1009.892	1009.898	0.000989	0.000989	25	26	0.177033
8	1012.383	1012.392	0.000987	0.000987	75	26	0.17809
9	1007.43	1007.741	0.000992	0.000991	75	26	0.176309
10	1017.545	1017.578	0.000982	0.000982	75	26	0.17577
Rata-rata	**1003.737**	**1003.797**	**0.000996**	**0.000995**	**75**	**26**	**0.177044**

Tabel 17 Hasil uji metode *crossover* memperhatikan *fitness* induk dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1003.06	1003.137	0.000996	0.000996	75	26	0.17552
2	965.5301	965.7581	0.001035	0.001034	75	26	0.177189
3	1013.024	1013.562	0.000986	0.000986	75	26	0.177813
4	1000.999	1001.08	0.000998	0.000998	75	27	0.177813
5	978.4002	978.5609	0.001021	0.001021	75	26	0.176714
6	993.4297	993.7183	0.001006	0.001005	75	26	0.176242
7	967.9797	968.0229	0.001032	0.001032	25	27	0.176242
8	1005.625	1005.637	0.000993	0.000993	75	26	0.178901
9	1004.791	1005.028	0.000994	0.000994	75	26	0.177642
10	1010.796	1011.209	0.000988	0.000988	75	25	0.178262
Rata-rata	**994.3634**	**994.5712**	**0.001005**	**0.001005**	**75**	**26.1**	**0.177234**

Tabel 18 Hasil uji metode *crossover* memperhatikan *fitness* induk dan ordinat dan mutasi pertukaran dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	966.7324	966.738	0.001033	0.001033	75	32	0.156141
2	1005.978	1006.006	0.000993	0.000993	75	32	0.16536
3	963.6343	963.7057	0.001037	0.001037	75	33	0.163333
4	1006.358	1006.842	0.000993	0.000992	75	35	0.166212
5	1012.861	1012.936	0.000986	0.000986	75	34	0.164099
6	965.9568	966.1847	0.001034	0.001034	75	35	0.161232
7	994.3874	994.9663	0.001005	0.001004	25	33	0.162093
8	994.2183	994.4154	0.001005	0.001005	75	34	0.164273
9	1013.578	1013.691	0.000986	0.000986	75	33	0.164542
10	1001.442	1001.446	0.000998	0.000998	75	32	0.164183
Rata-rata	**992.5147**	**992.6931**	**0.001007**	**0.001007**	**75**	**33.3**	**0.163147**

Tabel 19 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pengantian dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1103.42	1103.42	0.000905	0.000905	75	13	0.175169
2	1097.294	1097.487	0.000911	0.00091	75	14	0.173571
3	1114.755	1114.758	0.000896	0.000896	75	7	0.173492
4	1103.782	1103.878	0.000905	0.000905	75	9	0.175738
5	1090.002	1090.597	0.000917	0.000916	75	13	0.17022
6	1098.032	1098.813	0.00091	0.000909	75	12	0.174783
7	1090.706	1091.071	0.000916	0.000916	75	13	0.172548
8	1098.451	1098.462	0.00091	0.00091	75	13	0.173837
9	1091.293	1091.333	0.000916	0.000915	75	12	0.171753
10	1101.066	1101.07	0.000907	0.000907	75	9	0.175193
Rata-rata	**1098.88**	**1099.089**	**0.000909**	**0.000909**	**75**	**11.5**	**0.17363**

Tabel 20 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pengantian dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1073.276	1073.413	0.000931	0.000931	75	24	0.171482
2	1087.344	1087.346	0.000919	0.000919	75	22	0.172484
3	1085.444	1085.483	0.00092	0.00092	75	23	0.173933
4	1082.641	1082.668	0.000923	0.000923	75	24	0.171809
5	1070.935	1070.974	0.000933	0.000933	75	23	0.169498
6	1088.348	1088.352	0.000918	0.000918	75	23	0.171983
7	1082.119	1082.132	0.000923	0.000923	75	22	0.170998
8	1083.178	1083.186	0.000922	0.000922	75	20	0.170131
9	1082.007	1082.148	0.000923	0.000923	75	23	0.170297
10	1094.95	1094.95	0.000912	0.000912	75	15	0.172397
Rata-rata	**1083.024**	**1083.065**	**0.000923**	**0.000922**	**75**	**21.9**	**0.171501**

Tabel 21 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pengantian dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1077.038	1077.062	0.000928	0.000928	75	23	0.169443
2	1086.115	1086.115	0.00092	0.00092	75	18	0.169785
3	1089.226	1089.415	0.000917	0.000917	75	22	0.171253
4	1083.046	1083.07	0.000922	0.000922	75	23	0.169187
5	1072.203	1072.225	0.000932	0.000932	75	24	0.169424
6	1092.752	1092.753	0.000914	0.000914	75	20	0.172693
7	1079.948	1079.988	0.000925	0.000925	75	24	0.169115
8	1072.79	1073.072	0.000931	0.000931	75	23	0.166011
9	1078.618	1078.654	0.000926	0.000926	75	24	0.168656
10	1085.19	1085.289	0.000921	0.000921	75	23	0.16974
Rata-rata	**1081.693**	**1081.764**	**0.000924**	**0.000924**	**75**	**22.4**	**0.169531**

Tabel 22 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pengantian dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1071.672	1071.832	0.000932	0.000932	75	23	0.167487
2	1061.963	1062.002	0.000941	0.000941	75	24	0.167081
3	1072.371	1072.941	0.000932	0.000931	75	23	0.167807
4	1071.014	1071.043	0.000933	0.000933	75	24	0.167743
5	1085.118	1085.123	0.000921	0.000921	75	24	0.170545
6	1085.745	1085.814	0.00092	0.00092	75	24	0.170144
7	1091.537	1091.569	0.000915	0.000915	75	24	0.17269
8	1084.951	1084.995	0.000921	0.000921	75	24	0.171378
9	1082.48	1082.484	0.000923	0.000923	75	21	0.171132
10	1076.805	1076.806	0.000928	0.000928	75	24	0.167736
Rata-rata	**1078.366**	**1078.461**	**0.000927**	**0.000926**	**75**	**23.5**	**0.169374**

Tabel 23 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pengantian dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1076.758	1076.843	0.000928	0.000928	75	23	0.170914
2	1075.682	1076.021	0.000929	0.000928	75	24	0.171674
3	1075.895	1076.015	0.000929	0.000928	75	23	0.167157
4	1079.612	1079.646	0.000925	0.000925	75	19	0.170467
5	1069.33	1069.431	0.000934	0.000934	75	23	0.168511
6	1093.032	1093.032	0.000914	0.000914	75	23	0.170467
7	1088.426	1088.45	0.000918	0.000918	75	24	0.1708
8	1092.737	1092.755	0.000914	0.000914	75	23	0.169867
9	1083.621	1083.627	0.000922	0.000922	75	22	0.169747
10	1087.974	1088.012	0.000918	0.000918	75	21	0.168981
Rata-rata	**1082.307**	**1082.383**	**0.000923**	**0.000923**	**75**	**22.5**	**0.169858**

Tabel 24 Hasil uji metode *crossover* satu titik potong biasa dan mutasi pengantian dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1095.318	1095.365	0.000912	0.000912	75	22	0.172612
2	1072.439	1072.873	0.000932	0.000931	75	35	0.172575
3	1067.843	1067.913	0.000936	0.000936	75	35	0.167479
4	1082.422	1082.476	0.000923	0.000923	75	29	0.173029
5	1074.891	1074.91	0.000929	0.000929	75	31	0.17187
6	1064.474	1064.477	0.000939	0.000939	75	35	0.170014
7	1080.223	1080.243	0.000925	0.000925	75	27	0.171499
8	1078.961	1078.973	0.000926	0.000926	75	28	0.170165
9	1058.512	1058.684	0.000944	0.000944	75	35	0.167344
10	1068.727	1068.747	0.000935	0.000935	75	35	0.167223
Rata-rata	**1074.381**	**1074.466**	**0.00093**	**0.00093**	**75**	**31.2**	**0.170381**

Tabel 25 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi penggantian dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1130.724	1133.828	0.000884	0.000881	75	7	0.18043
2	1124.464	1127.342	0.000889	0.000886	75	7	0.176506
3	1122.8	1134.36	0.00089	0.000881	75	4	0.177782
4	1132.066	1133.303	0.000883	0.000882	75	5	0.179337
5	1136.886	1138.319	0.000879	0.000878	75	4	0.178952
6	1125.752	1129.786	0.000888	0.000884	75	7	0.177005
7	1131.071	1133.55	0.000883	0.000881	75	6	0.177078
8	1135.662	1138.531	0.00088	0.000878	75	3	0.181101
9	1134.518	1137.638	0.000881	0.000878	75	4	0.178811
10	1125.854	1128.375	0.000887	0.000885	75	6	0.175937
Rata-rata	**1129.98**	**1133.503**	**0.000884**	**0.000881**	**75**	**5.3**	**0.178294**

Tabel 26 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi penggantian dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1128.594	1131.381	0.000885	0.000883	75	12	0.179366
2	1130.344	1132.24	0.000884	0.000882	75	15	0.179935
3	1131.665	1133.922	0.000883	0.000881	75	7	0.176564
4	1135.423	1136.392	0.00088	0.000879	75	11	0.177196
5	1123.103	1125.844	0.00089	0.000887	75	20	0.177813
6	1132.315	1134.408	0.000882	0.000881	75	5	0.178356
7	1127.271	1131	0.000886	0.000883	75	14	0.179825
8	1129.651	1133.075	0.000884	0.000882	75	5	0.177018
9	1122.85	1127.872	0.00089	0.000886	75	11	0.176995
10	1132.997	1137.046	0.000882	0.000879	75	9	0.177642
Rata-rata	**1129.421**	**1132.318**	**0.000885**	**0.000882**	**75**	**10.9**	**0.178071**

Tabel 27 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi penggantian dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1134.374	1135.173	0.000881	0.00088	75	9	0.174979
2	1125.13	1128.095	0.000888	0.000886	75	15	0.174907
3	1132.676	1133.485	0.000882	0.000881	75	7	0.174476
4	1133.937	1135.393	0.000881	0.00088	75	6	0.174289
5	1126.45	1129.007	0.000887	0.000885	75	14	0.176713
6	1129.297	1131.715	0.000885	0.000883	75	8	0.172702
7	1136.345	1139.511	0.000879	0.000877	75	7	0.171853
8	1134.729	1135.743	0.00088	0.00088	75	7	0.174422
9	1131.777	1134.737	0.000883	0.00088	75	15	0.176125
10	1130.597	1132.992	0.000884	0.000882	75	6	0.174003
Rata-rata	**1131.531**	**1133.585**	**0.000883**	**0.000881**	**75**	**9.4**	**0.174447**

Tabel 28 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi penggantian dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1124.499	1128.078	0.000888	0.000886	75	11	0.174436
2	1131.457	1133.553	0.000883	0.000881	75	13	0.173514
3	1122.709	1124.47	0.00089	0.000889	75	15	0.174896
4	1129.104	1129.842	0.000885	0.000884	75	8	0.175057
5	1124.268	1125.593	0.000889	0.000888	75	14	0.172263
6	1137.239	1138.572	0.000879	0.000878	75	9	0.175459
7	1132.556	1133.836	0.000882	0.000881	75	6	0.174617
8	1125.331	1129.43	0.000888	0.000885	75	8	0.176149
9	1137.049	1138.018	0.000879	0.000878	75	12	0.172014
10	1134.315	1137.083	0.000881	0.000879	75	5	0.172987
Rata-rata	**1129.853**	**1131.848**	**0.000884**	**0.000883**	**75**	**10.1**	**0.174139**

Tabel 29 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi penggantian dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1136.126	1139.644	0.000879	0.000877	75	9	0.175095
2	1136.011	1140.223	0.000879	0.000876	75	5	0.174657
3	1124.858	1129.286	0.000888	0.000885	75	7	0.176208
4	1126.837	1127.963	0.000887	0.000886	75	14	0.172791
5	1128.074	1131.879	0.000886	0.000883	75	8	0.174736
6	1128.962	1130.707	0.000885	0.000884	75	12	0.17295
7	1126.083	1130.576	0.000887	0.000884	75	11	0.174958
8	1133.448	1135.373	0.000881	0.00088	75	8	0.175443
9	1129.807	1133.013	0.000884	0.000882	75	13	0.172126
10	1134.365	1135.888	0.000881	0.00088	75	8	0.174882
Rata-rata	**1130.457**	**1133.455**	**0.000884**	**0.000881**	**75**	**9.5**	**0.174385**

Tabel 30 Hasil uji metode *crossover* satu titik potong pembalikan absis dan ordinat dan mutasi penggantian dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1138.629	1139.541	0.000877	0.000877	75	7	0.178548
2	1125.82	1127.537	0.000887	0.000886	75	14	0.1761
3	1126.977	1130.231	0.000887	0.000884	75	12	0.179862
4	1120.753	1121.995	0.000891	0.00089	75	11	0.181423
5	1125.023	1128.987	0.000888	0.000885	75	8	0.176759
6	1107.989	1108.735	0.000902	0.000901	75	20	0.173549
7	1114.597	1122.048	0.000896	0.00089	75	12	0.176394
8	1130.605	1133.899	0.000884	0.000881	75	8	0.177377
9	1126.601	1128.855	0.000887	0.000885	75	11	0.177307
10	1122.825	1125.725	0.00089	0.000888	75	19	0.175591
Rata-rata	**1123.982**	**1126.755**	**0.000889**	**0.000887**	**75**	**12.2**	**0.177291**

Tabel 31 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi penggantian dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	997.7425	997.7425	0.001001	0.001001	75	3	0.162525
2	1001.063	1001.514	0.000998	0.000997	75	3	0.162913
3	967.8075	967.8075	0.001032	0.001032	75	3	0.159724
4	990.7817	991.2706	0.001008	0.001008	75	3	0.165134
5	1000.977	1000.977	0.000998	0.000998	75	3	0.161855
6	965.7905	965.7905	0.001034	0.001034	75	3	0.160977
7	1000.193	1001.07	0.000999	0.000998	25	3	0.167177
8	973.1516	973.1516	0.001027	0.001027	75	3	0.162888
9	995.182	995.182	0.001004	0.001004	75	3	0.163305
10	1042.418	1043.512	0.000958	0.000957	75	3	0.17079
Rata-rata	**993.5107**	**993.8018**	**0.001006**	**0.001006**	**75**	**3**	**0.163729**

Tabel 32 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi penggantian dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	970.9396	970.9396	0.001029	0.001029	75	7	0.160791
2	994.5607	995.3172	0.001004	0.001004	75	4	0.162616
3	991.1286	991.6784	0.001008	0.001007	75	5	0.163412
4	1023.274	1023.274	0.000976	0.000976	75	6	0.16504
5	1012.202	1012.202	0.000987	0.000987	75	7	0.164233
6	993.9363	993.9363	0.001005	0.001005	75	5	0.166703
7	1041.684	1041.684	0.000959	0.000959	25	5	0.168169
8	1022.135	1022.135	0.000977	0.000977	75	6	0.162916
9	1005.796	1005.796	0.000993	0.000993	75	11	0.162905
10	998.835	999.8493	0.001	0.000999	75	4	0.16624
Rata-rata	**1005.449**	**1005.681**	**0.000994**	**0.000994**	**75**	**6**	**0.164302**

Tabel 33 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi penggantian dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	986.991	986.991	0.001012	0.001012	75	8	0.164425
2	976.4298	976.4298	0.001023	0.001023	75	11	0.161584
3	1024.438	1024.438	0.000975	0.000975	75	8	0.162972
4	1013.089	1013.089	0.000986	0.000986	75	7	0.160152
5	1000.642	1001.199	0.000998	0.000998	75	6	0.164428
6	1032.377	1032.377	0.000968	0.000968	75	10	0.168458
7	1041.894	1042.568	0.000959	0.000958	25	6	0.170483
8	1046.603	1046.603	0.000955	0.000955	75	6	0.173029
9	1017.268	1018.038	0.000982	0.000981	75	6	0.163076
10	1008.061	1008.638	0.000991	0.00099	75	11	0.162066
Rata-rata	**1014.779**	**1015.037**	**0.000985**	**0.000985**	**75**	**7.9**	**0.165067**

Tabel 34 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi penggantian dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1001.088	1001.088	0.000998	0.000998	75	15	0.16484
2	1003.489	1003.489	0.000996	0.000996	75	6	0.166254
3	987.1572	987.6256	0.001012	0.001012	75	6	0.160163
4	1014.812	1014.812	0.000984	0.000984	75	9	0.166295
5	1008.839	1010.071	0.00099	0.000989	75	11	0.160849
6	1024.693	1024.942	0.000975	0.000975	75	6	0.165263
7	995.4849	996.5435	0.001004	0.001002	25	6	0.159972
8	1015.612	1015.612	0.000984	0.000984	75	6	0.167544
9	1034.951	1035.763	0.000965	0.000965	75	6	0.169231
10	1022.577	1022.894	0.000977	0.000977	75	5	0.164807
Rata-rata	**1010.87**	**1011.284**	**0.000988**	**0.000988**	**75**	**7.6**	**0.164522**

Tabel 35 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi penggantian dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	986.6052	986.6053	0.001013	986.6053	75	6	0.162149
2	999.0925	999.0926	0.001	0.001	75	13	0.165632
3	1036.833	1036.833	0.000964	0.000964	75	7	0.167494
4	1001.841	1001.841	0.000997	0.000997	75	16	0.163849
5	1025.37	1025.37	0.000974	0.000974	75	6	0.163144
6	977.1089	978.1923	0.001022	0.001021	75	9	0.159407
7	996.8873	998.2604	0.001002	0.001001	25	10	0.160877
8	986.093	986.093	0.001013	0.001013	75	7	0.162156
9	1005.157	1005.157	0.000994	0.000994	75	16	0.163655
10	1012.497	1013.629	0.000987	0.000986	75	11	0.166341
Rata-rata	**1002.749**	**1003.108**	**0.000997**	**98.66142**	**75**	**10.1**	**0.16347**

Tabel 36 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi penggantian dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1022.849	1022.849	0.000977	0.000977	75	11	0.16472
2	1000.146	1000.735	0.000999	0.000998	75	17	0.162759
3	1010.249	1010.249	0.000989	0.000989	75	30	0.165481
4	1009.314	1009.89	0.00099	0.000989	75	9	0.161611
5	986.3515	987.2408	0.001013	0.001012	75	7	0.162469
6	1016.937	1016.937	0.000982	0.000982	75	10	0.166925
7	981.6614	981.6614	0.001018	0.001018	25	12	0.160909
8	1034.362	1035.042	0.000966	0.000965	75	7	0.16659
9	1036.05	1036.405	0.000964	0.000964	75	7	0.165259
10	1005.582	1006.439	0.000993	0.000993	75	15	0.162833
Rata-rata	**1010.35**	**1010.745**	**0.000989**	**0.000989**	**75**	**12.5**	**0.163956**

Tabel 37 Hasil uji metode *crossover* dengan titik potong biasa dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1108.868	1110.537	0.000901	0.0009	75	12	0.178246
2	1109.225	1109.815	0.000901	0.0009	75	12	0.177952
3	1101.99	1103.177	0.000907	0.000906	75	12	0.177943
4	1096.596	1097.101	0.000911	0.000911	75	12	0.176109
5	1105.345	1106.623	0.000904	0.000903	75	12	0.176406
6	1099.765	1101.124	0.000908	0.000907	75	12	0.176204
7	1095.192	1096.107	0.000912	0.000911	75	12	0.175523
8	1099.67	1100.878	0.000909	0.000908	75	12	0.17481
9	1103.734	1104.648	0.000905	0.000904	75	11	0.176288
10	1109.433	1110.158	0.000901	0.0009	75	12	0.176923
Rata-rata	**1102.982**	**1104.017**	**0.000906**	**0.000905**	**75**	**11.9**	**0.17664**

Tabel 38 Hasil uji metode *crossover* dengan titik potong biasa dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1105.58	1105.837	0.000904	0.000903	75	24	0.178011
2	1092.509	1092.751	0.000914	0.000914	75	24	0.176187
3	1102.95	1103.322	0.000906	0.000906	75	24	0.176518
4	1104.027	1104.427	0.000905	0.000905	75	24	0.176093
5	1102.25	1103.03	0.000906	0.000906	75	23	0.176213
6	1095.72	1096.506	0.000912	0.000911	75	24	0.174774
7	1105.036	1105.253	0.000904	0.000904	75	24	0.178285
8	1099.676	1100.019	0.000909	0.000908	75	24	0.177393
9	1092.188	1092.585	0.000915	0.000914	75	24	0.175545
10	1105.109	1106.715	0.000904	0.000903	75	23	0.177416
Rata-rata	**1100.504**	**1101.045**	**0.000908**	**0.000907**	**75**	**23.8**	**0.176643**

Tabel 39 Hasil uji metode *crossover* dengan titik potong biasa dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1100.892	1101.16	0.000908	0.000907	75	24	0.178438
2	1100.322	1100.984	0.000908	0.000907	75	25	0.17593
3	1100.091	1100.657	0.000908	0.000908	75	24	0.179374
4	1089.609	1090.158	0.000917	0.000916	75	24	0.17338
5	1098.485	1098.736	0.00091	0.000909	75	24	0.17516
6	1097.327	1097.945	0.00091	0.00091	75	24	0.178154
7	1101.375	1102.114	0.000907	0.000907	75	25	0.17838
8	1107.423	1108.102	0.000902	0.000902	75	24	0.176635
9	1095.808	1096.564	0.000912	0.000911	75	23	0.177364
10	1083.567	1083.893	0.000922	0.000922	75	24	0.176071
Rata-rata	**1097.49**	**1098.031**	**0.00091**	**0.00091**	**75**	**24.1**	**0.176889**

Tabel 40 Hasil uji metode *crossover* dengan titik potong biasa dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1105.476	1105.71	0.000904	0.000904	75	24	0.175939
2	1110.297	1110.921	0.0009	0.000899	75	24	0.177424
3	1096.541	1096.84	0.000911	0.000911	75	28	0.178743
4	1083.519	1084.382	0.000922	0.000921	75	25	0.175592
5	1099.66	1100.798	0.000909	0.000908	75	25	0.176852
6	1100.891	1101.137	0.000908	0.000907	75	25	0.17553
7	1103.082	1103.415	0.000906	0.000905	75	24	0.177778
8	1100.171	1100.401	0.000908	0.000908	75	24	0.177127
9	1102.447	1102.886	0.000906	0.000906	75	24	0.177636
10	1108.652	1109.679	0.000901	0.0009	75	24	0.176633
Rata-rata	**1101.073**	**1101.617**	**0.000907**	**0.000907**	**75**	**24.7**	**0.176925**

Tabel 41 Hasil uji metode *crossover* dengan titik potong biasa dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1091.548	1091.823	0.000915	0.000915	75	24	0.177025
2	1091.495	1091.942	0.000915	0.000915	75	24	0.1758
3	1098.493	1098.887	0.00091	0.000909	75	29	0.178824
4	1096.607	1097.485	0.000911	0.00091	75	25	0.176152
5	1097.752	1097.943	0.00091	0.00091	75	23	0.177313
6	1101.622	1102.133	0.000907	0.000907	75	24	0.175679
7	1099.946	1100.415	0.000908	0.000908	75	23	0.176144
8	1093.288	1094.008	0.000914	0.000913	75	24	0.175577
9	1112.556	1112.817	0.000898	0.000898	75	24	0.177532
10	1099.922	1100.292	0.000908	0.000908	75	25	0.177503
Rata-rata	**1098.323**	**1098.774**	**0.00091**	**0.000909**	**75**	**24.5**	**0.176755**

Tabel 42 Hasil uji metode *crossover* dengan titik potong biasa dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1084.264	1084.474	0.000921	0.000921	75	36	0.174868
2	1100.516	1100.738	0.000908	0.000908	75	36	0.176326
3	1093.874	1094.126	0.000913	0.000913	75	35	0.176711
4	1103.563	1103.891	0.000905	0.000905	75	36	0.178768
5	1097.994	1098.184	0.00091	0.00091	75	35	0.177751
6	1112.643	1113.462	0.000898	0.000897	75	35	0.17984
7	1088.051	1088.185	0.000918	0.000918	75	36	0.174272
8	1101.025	1101.26	0.000907	0.000907	75	35	0.175854
9	1090.8	1091.191	0.000916	0.000916	75	36	0.178458
10	1093.356	1094.206	0.000914	0.000913	75	36	0.17632
Rata-rata	**1096.609**	**1096.972**	**0.000911**	**0.000911**	**75**	**35.6**	**0.176917**

Tabel 43 Hasil uji metode *crossover* dengan titik potong pembalikan absis dan ordinat dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1122.821	1125.805	0.00089	0.000887	25	5	0.179614
2	1121.677	1125.225	0.000891	0.000888	25	2	0.179533
3	1123.389	1127.108	0.000889	0.000886	25	3	0.17962
4	1118.41	1124.797	0.000893	0.000888	25	5	0.176941
5	1121.901	1123.666	0.000891	0.000889	25	6	0.178767
6	1126.8	1133.565	0.000887	0.000881	25	2	0.179245
7	1119.196	1123.404	0.000893	0.000889	25	4	0.176265
8	1122.96	1126.024	0.00089	0.000887	25	5	0.178906
9	1125.471	1127.36	0.000888	0.000886	25	10	0.179669
10	1122.804	1127.45	0.00089	0.000886	25	3	0.180706
Rata-rata	**1122.543**	**1126.44**	**0.00089**	**0.000887**	**25**	**4.5**	**0.178927**

Tabel 44 Hasil uji metode *crossover* dengan titik potong pembalikan absis dan ordinat dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1122.399	1127.878	0.00089	0.000886	25	8	0.17959
2	1122.596	1125.387	0.00089	0.000888	25	7	0.179794
3	1113.491	1117.108	0.000897	0.000894	25	14	0.180654
4	1119.039	1123.169	0.000893	0.00089	25	9	0.179864
5	1125.352	1127.623	0.000888	0.000886	25	9	0.181471
6	1119.126	1122.686	0.000893	0.00089	25	11	0.178446
7	1121.369	1123.048	0.000891	0.00089	25	9	0.179042
8	1118.374	1125.494	0.000893	0.000888	25	6	0.178251
9	1123.368	1124.979	0.000889	0.000888	25	13	0.179966
10	1118.45	1121.251	0.000893	0.000891	25	21	0.177187
Rata-rata	**1120.356**	**1123.862**	**0.000892**	**0.000889**	**25**	**10.7**	**0.179426**

Tabel 45 Hasil uji metode *crossover* dengan titik potong pembalikan absis dan ordinat dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1119.922	1122.398	0.000892	0.00089	25	9	0.176263
2	1121.085	1127.393	0.000891	0.000886	25	7	0.177959
3	1115.751	1124.301	0.000895	0.000889	25	5	0.183081
4	1127.86	1128.85	0.000886	0.000885	25	7	0.180286
5	1118.083	1122.007	0.000894	0.00089	25	12	0.17861
6	1123.003	1124.122	0.00089	0.000889	25	14	0.179559
7	1127.741	1130.531	0.000886	0.000884	25	9	0.181112
8	1118.908	1122.646	0.000893	0.00089	25	13	0.1797
9	1122.428	1125.257	0.00089	0.000888	25	10	0.178936
10	1125.003	1127.698	0.000888	0.000886	25	5	0.179276
Rata-rata	**1121.978**	**1125.52**	**0.00089**	**0.000888**	**25**	**9.1**	**0.179478**

Tabel 46 Hasil uji metode *crossover* dengan titik potong pembalikan absis dan ordinat dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1118.092	1121.016	0.000894	0.000891	25	7	0.17979
2	1118.512	1123.792	0.000893	0.000889	25	10	0.181133
3	1123.648	1128.348	0.000889	0.000885	25	8	0.178001
4	1128.697	1130.681	0.000885	0.000884	25	11	0.181094
5	1121.565	1125.271	0.000891	0.000888	25	15	0.176576
6	1123.685	1125.94	0.000889	0.000887	25	13	0.178262
7	1119.923	1123.965	0.000892	0.000889	25	12	0.178053
8	1121.944	1124.768	0.000891	0.000888	25	6	0.179084
9	1111.251	1115.635	0.000899	0.000896	25	22	0.177025
10	1114.27	1119.14	0.000897	0.000893	25	15	0.176801
Rata-rata	**1120.159**	**1123.856**	**0.000892**	**0.000889**	**25**	**11.9**	**0.178582**

Tabel 47 Hasil uji metode *crossover* dengan titik potong pembalikan absis dan ordinat dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1107.969	1119.346	0.000902	0.000893	25	10	0.178544
2	1118.016	1127.523	0.000894	0.000886	25	6	0.178571
3	1110.362	1115.46	0.0009	0.000896	25	11	0.176539
4	1119.427	1123.904	0.000893	0.000889	25	7	0.179522
5	1122.347	1124.965	0.00089	0.000888	25	7	0.178708
6	1118.686	1124.163	0.000893	0.000889	25	9	0.179034
7	1113.557	1116.289	0.000897	0.000895	25	9	0.17754
8	1127.453	1129.4	0.000886	0.000885	25	5	0.178841
9	1118.854	1123.949	0.000893	0.000889	25	8	0.179183
10	1122.779	1124.844	0.00089	0.000888	25	8	0.180238
Rata-rata	**1117.945**	**1122.984**	**0.000894**	**0.00089**	**25**	**8**	**0.178672**

Tabel 48 Hasil uji metode *crossover* dengan titik potong pembalikan absis dan ordinat dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	1109.753	1115.668	0.0009	0.000896	25	18	0.176152
2	1115.459	1117.53	0.000896	0.000894	25	16	0.177028
3	1117.82	1119.834	0.000894	0.000892	25	13	0.178751
4	1120.243	1122.856	0.000892	0.00089	25	9	0.180371
5	1121.494	1124.585	0.000891	0.000888	25	8	0.180235
6	1111.254	1117.114	0.000899	0.000894	25	16	0.176458
7	1117.338	1120.232	0.000894	0.000892	25	18	0.178587
8	1123.004	1124.344	0.00089	0.000889	25	9	0.181115
9	1121.921	1122.915	0.000891	0.00089	25	13	0.178887
10	1121.104	1124.229	0.000891	0.000889	25	16	0.177592
Rata-rata	**1117.939**	**1120.931**	**0.000894**	**0.000891**	**25**	**13.6**	**0.178517**

Tabel 49 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.5

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	906.5638	906.7938	0.001102	0.001102	75	15	0.155624
2	963.3741	963.3741	0.001037	0.001037	75	11	0.159306
3	949.913	950.1779	0.001052	0.001051	75	8	0.159559
4	952.7661	952.8807	0.001048	0.001048	75	13	0.156733
5	965.8226	965.8226	0.001034	0.001034	75	7	0.16078
6	936.4304	936.4305	0.001067	0.001067	75	12	0.155474
7	944.3943	944.8707	0.001058	0.001057	25	6	0.156543
8	959.7654	960.1878	0.001041	0.00104	75	15	0.160784
9	992.4938	992.4938	0.001007	0.001007	75	11	0.164784
10	947.5672	947.5721	0.001054	0.001054	75	9	0.158806
Rata-rata	**951.9091**	**952.0604**	**0.00105**	**0.00105**	**75**	**10.7**	**0.158839**

Tabel 50 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.6

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	979.4697	979.4697	0.00102	0.00102	75	16	0.165131
2	957.061	957.061	0.001044	0.001044	75	19	0.163622
3	956.3639	956.3639	0.001045	0.001045	75	8	0.159713
4	975.2377	975.59	0.001024	0.001024	75	22	0.161245
5	980.3901	980.4544	0.001019	0.001019	75	22	0.160577
6	978.5133	978.6751	0.001021	0.001021	75	22	0.163159
7	979.7003	979.777	0.00102	0.00102	25	23	0.163112
8	981.9313	981.9707	0.001017	0.001017	75	22	0.167002
9	985.8144	985.9385	0.001013	0.001013	75	11	0.162874
10	972.9036	972.9035	0.001027	0.001027	75	18	0.160636
Rata-rata	**974.7385**	**974.8204**	**0.001025**	**0.001025**	**75**	**18.3**	**0.162707**

Tabel 51 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.7

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	948.5696	949.001	0.001053	0.001053	75	16	0.160501
2	972.4722	972.7198	0.001027	0.001027	75	18	0.160063
3	987.3749	987.4479	0.001012	0.001012	75	23	0.162454
4	975.0316	975.3802	0.001025	0.001024	75	20	0.160876
5	964.054	964.2943	0.001036	0.001036	75	23	0.162452
6	940.4236	940.8	0.001062	0.001062	75	13	0.156975
7	982.9164	982.9164	0.001016	0.001016	25	21	0.164673
8	974.1794	974.2021	0.001025	0.001025	75	24	0.162311
9	975.3129	975.4025	0.001024	0.001024	75	13	0.161644
10	929.9665	930.2171	0.001074	0.001074	75	16	0.157685
Rata-rata	**965.0301**	**965.2381**	**0.001036**	**0.001035**	**75**	**18.7**	**0.160963**

Tabel 52 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.8

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	920.576	920.7609	0.001085	0.001085	75	26	0.156287
2	983.3734	983.4509	0.001016	0.001016	75	23	0.164626
3	974.5396	974.651	0.001025	0.001025	75	24	0.162659
4	912.328	912.3809	0.001095	0.001095	75	16	0.154734
5	981.2001	981.3187	0.001018	0.001018	75	23	0.166394
6	926.9161	926.9161	0.001078	0.001078	75	20	0.158156
7	949.1278	949.1278	0.001052	0.001052	25	25	0.158331
8	951.9419	952.141	0.001049	0.001049	75	15	0.158505
9	973.9915	974.4852	0.001026	0.001025	75	27	0.16361
10	974.5344	974.7137	0.001025	0.001025	75	17	0.162617
Rata-rata	**954.8529**	**954.9946**	**0.001047**	**0.001047**	**75**	**21.6**	**0.160592**

Tabel 53 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 0.9

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	980.4043	980.4891	0.001019	0.001019	75	22	0.163909
2	984.5753	984.5753	0.001015	0.001015	75	12	0.164189
3	960.8863	961.2496	0.00104	0.001039	75	27	0.159495
4	972.446	972.6033	0.001027	0.001027	75	24	0.160704
5	965.6011	965.6716	0.001035	0.001034	75	20	0.160045
6	978.3657	978.5581	0.001021	0.001021	75	18	0.160376
7	1024.831	1025.108	0.000975	0.000975	25	13	0.168333
8	965.3236	965.4165	0.001035	0.001035	75	12	0.163191
9	991.8972	991.9227	0.001007	0.001007	75	19	0.163665
10	937.3942	937.655	0.001066	0.001065	75	16	0.160481
Rata-rata	**976.1725**	**976.325**	**0.001024**	**0.001024**	**75**	**18.3**	**0.162439**

Tabel 54 Hasil uji metode *crossover* dengan memperhatikan *fitness* induk dan mutasi pencarian optimum lokal dengan probabilitas *crossover* 1.0

No	Min drms	Avg Drms	Max Fitness	Avg Fitness	Ukuran (KB)	Waktu (s)	MSE
1	911.4175	911.5439	0.001096	0.001096	75	21	0.15597
2	933.7331	933.8983	0.00107	0.00107	75	23	0.160943
3	969.8979	970.2576	0.00103	0.00103	75	25	0.160743
4	989.1744	989.6135	0.00101	0.001009	75	27	0.165222
5	974.5792	974.6605	0.001025	0.001025	75	21	0.161516
6	968.4136	968.4138	0.001032	0.001032	75	13	0.161215
7	954.3594	954.4359	0.001047	0.001047	25	29	0.159519
8	966.7141	967.0739	0.001033	0.001033	75	29	0.163056
9	987.5811	987.6821	0.001012	0.001011	75	19	0.166349
10	978.0253	978.1086	0.001021	0.001021	75	32	0.161687
Rata-rata	**963.3896**	**963.5688**	**0.001038**	**0.001037**	**75**	**23.9**	**0.161622**