OPTIMASI PENGEPAKAN SEGI EMPAT MENGGUNAKAN ALGORITMA GENETIK DAN ALGORITMA BOTTOM-LEFT

SKRIPSI

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

oleh:
HANIF ROBBANI
0310960035-96

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

2009


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

OPTIMASI PENGEPAKAN SEGI EMPAT MENGGUNAKAN ALGORITMA GENETIK DAN ALGORITMA BOTTOM-LEFT

Oleh :
HANIF ROBBANI
0310960035-96

Setelah dipertahankan di depan Majelis Penguji pada tanggal 30 juni 2009 dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Drs. Achmad Ridok, M.Kom	**Djoko Pramono, ST.**
NIP. 132 090 392	**NIP. 132 313 603**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, M.Sc.
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Hanif Robbani**
NIM	**: 0310960035-96**
Jurusan	**: Matematika**
Program Studi	**: Ilmu Komputer**
Penulis skripsi berjudul	**: Optimasi Pengepakan Segi Empat Menggunakan Algoritma Genetik dan Algoritma Bottom-left.**

Dengan ini menyatakan bahwa :

1. **Isi dari skripsi 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 skripsi ini.**
2. **Apabila dikemudian hari ternyata skripsi 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, 30 Juni 2009
Yang menyatakan,

Hanif Robbani
NIM. 0310960035-96

OPTIMASI PENGEPAKAN SEGI EMPAT MENGGUNAKAN ALGORITMA GENETIK DAN ALGORITMA BOTTOM-LEFT

ABSTRAK

Optimasi pengepakan segi empat diperlukan untuk mendapatkan luas sisa yang maksimal. Algoritma genetik yang dikombinasikan dengan algoritma *bottom-left* merupakan alternatif solusi untuk menghasilkan pengepakan yang optimal. Algoritma *bottom-left* digunakan untuk memetakan segi empat ke dalam bidang datar yang akan dipotong, sedangkan algoritma genetik digunakan untuk menentukan hasil dari algoritma *bottom-left* yang paling optimal. Fokus penelitian ini untuk mengetahui perbedaan yang didapatkan dari beberapa probabilitas *crossover* yang berbeda. Dari hasil pemecahan masalah optimasi pengepakan segi empat, nilai probabilitas *crossover 50%* menghasilkan rata-rata *fitness* terbaik. Di dalam kasus ini algoritma genetik yang dikombinasikan dengan algoritma *bottom-left* selalu dapat memecahkan dan menyelesaikan sebaik mungkin.

RECTANGULAR INTERSECTION OPTIMIZATION USING GENETIC AND BOTTOM-LEFT ALGORITHM

ABSTRACT

Rectangular intersection optimization required to achieve maximum area. The combination of both genetic and bottom-left algorithm is an alternative solution to achieve optimal intersection. Bottom-left algorithm is used for rectangular mapping into two-dimensional model for intersection. In order to select the most optimized result from bottom-left algorithm; genetic algorithm takes its place. Focus of this research is to know the result difference between a few crossover probability. As the results of rectangular intersection optimization, crossover probability with value 50% generate the best average fitness. In this case, the combinations of genetic and bottom-left algorithm always solve the problem.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahnya, Tugas Akhir yang berjudul "**Optimasi Pengepakan Segi Empat Menggunakan Algoritma Genetik Dan Algoritma Bottom-Left**" ini dapat diselesaikan. Tugas Akhir ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya yang baik dan suci dengan rahmat yang berkah-Nya menyelamatkan kita pada hari akhirat.

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bapak Drs. Achmad Ridok, M.Kom dan Bapak Djoko Pramono, ST selaku pembimbing. Terima kasih atas semua saran, bantuan, kritikan, waktu, dorongan semangat dan bimbingannya.
2. Bapak Wayan F. Mahmudy, S.Si, MT selaku Ketua Program Studi Ilmu Komputer Unibraw Malang.
3. Bapak, Ibu dan Kakak, yang senantiasa memberi inspirasi, dukungan dan semangat.
4. Semua teman-teman Ilmu Komputer angkatan 2003. Terima kasih atas semangat dan doanya.
5. Pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Semoga penulisan laporan tugas akhir ini bermanfaat bagi pembaca sekalian. Dengan tidak lupa kodratnya sebagai manusia, penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, dan mengandung banyak kekurangan, sehingga

dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 30 Juni 2009

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan yang dicapai saat ini oleh ilmu kecerdasan buatan dalam menyelesaikan persoalan-persoalan yang ada semakin berkembang pesat. Ilmu kecerdasan buatan merupakan implementasi dari kemampuan cara berpikir manusia dalam menyelesaikan masalah-masalah yang dihadapinya. Munculnya algoritma genetik sebagai salah satu dari bentuk kecerdasan buatan semakin memperkaya cakupan bidang dari kecerdasan buatan. Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme seleksi alami dan genetika alami (Gen,1997). Pencarian dilakukan diantara alternatif titik optimal berdasarkan fungsi probabilistik. Kualitas solusi yang dihasilkan oleh algoritma genetik dapat dijamin karena pemilihan terus dilakukan terhadap alternatif solusi yang terbaik dari penyimpanan informasi dari jalur yang pernah dilalui.

Algoritma genetik dapat digunakan untuk melakukan optimasi pada masalah pengepakan segi empat. Pengepakan segi empat diperlukan untuk mengatur tata letak segi empat dengan berbagai ukuran pada sebuah segi empat yang lebih besar. Semakin banyak jumlah segi empat yang akan diletakkan akan semakin rumit proses pengepakan yang dilakukan (Callaghan,1999).

Bin packing atau pengepakan merupakan salah satu elemen penting yang sering digunakan pada industri-industri. Pengepakan yang efisien baik dari segi waktu dan ruang, sangat berpengaruh terhadap biaya produksi suatu industri. Maka dari itu optimisasi dalam hal pengepakan sangat dibutuhkan.

Pengepakan segi empat dapat ditemui pada industri plat baja maupun industri tekstil. Pengepakan persegi panjang atau lebih dikenal dengan *orthogonal bin packing* digunakan untuk memotong sebuah segi empat yang besar menjadi segi empat yang lebih kecil dalam berbagai ukuran.

Telah terdapat banyak penelitian mengenai optimasi pengepakan dengan berbagai macam metode dan pendekatan, seperti penggunaan metode algoritma *Greedy* (Liu dan Chen, 2006), metode *Corner-occupying action* (Chen dan Huang, 2006), juga dengan algoritma *bottom-left* seperti yang telah

dibahas oleh Callaghan, Nair dan Lewis dalam jurnalnya yang berjudul "*An Extension of the Orthogonal Packing Problem Through Dimensional Flexibility"*.

Dalam penelitian ini akan dibahas tentang pengepakan segi empat dengan menggunakan algoritma genetik yang dikombinasikan dengan algoritma *bottom-left* yang digunakan untuk meletakkan segi empat pada bidang datar sekaligus untuk mencari *fitness* dari kromosom.

1.2 Rumusan Masalah

Rumusan masalah yang diajukan didalam penyelesaian tugas akhir ini :

- Bagaimana melakukan pemindahan atau penggeseran segi empat yang akan dioptimasikan pada bidang datar dengan menggunakan algoritma *Bottom-Left (BL-algorithm)*.
- Bagaimana mendapatkan optimasi pengepakan segi empat pada bidang datar dengan menggunakan algoritma genetik dan dikombinasikan dengan algoritma *Bottom-Left* yang menghasilkan luas sisa maksimal dan dengan tinggi minimial dari benda segi empat yang disusun
- Bagaimana hasil evaluasi yang dihasilkan probabilitas *crossover* yang berbeda

1.3 Batasan Masalah

Pada skripsi ini akan dibatasi beberapa hal antara lain :

- Benda yang akan disusun berbentuk segi empat siku-siku (persegi panjang)
- Bidang datar sebagai media penyusunan berbentuk persegi panjang dan ditentukan panjang dan lebarnya.
- Total luas benda yang disusun tidak lebih besar dari bidang datar media penyusunan.

1.4 Tujuan

Adapun tujuan yang hendak dicapai pada tugas akhir ini adalah :

- Memahami permasalahan optimasi pengepakan benda berbentuk segi empat pada suatu bidang datar, permasalahan optimasi yang dimaksud adalah untuk mendapatkan luas sisa maksimal dan dengan tinggi minimal yang didapat dari

benda segi empat yang sudah tersusun di bidang datar setelah melakukan optimasi pengepakan.

- Memahami konsep dasar dan mekanisme kerja algoritma genetik. Pemahaman mekanisme kerja algoritma genetik secara umum dan juga pemahaman penggunaan algoritma genetik sehingga dapat digunakan untuk menyelesaikan permasalahan optimasi yang komplek.
- Memahami konsep dasar algoritma genetik untuk penyelesaian dari permasalahan optimasi pengepakan. Setelah memahami mekanisme secara umum kemudian dilanjutkan dengan penerapannya didalam penyelesaian optimasi pengepakan benda segi empat pada bidang datar dengan mengahasilkan luas sisa maksimal dan dengan tiggi minimal

1.5 Manfaat Penelitian

Manfaat yang akan dicapai dari tugas akhir ini untuk membantu masalah pengepakan segi empat pada bidang datar sehingga diperoleh luas sisa maksimal dan dengan tinggi minimal.

1.6 Sistematika Penulisan

Pembuatan tugas akhir ini dilakukan dengan pembagian bab sebagai berikut:

BAB I: PENDAHULUAN

Pada bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, serta sistematika penulisan tugas akhir.

BAB II: TINJAUAN PUSTAKA

Bab ini menjelaskan tentang penjadwalan secara umum, teori dasar algoritma genetika, dan penjadwalan menggunakan algoritma genetika. Adapun literatur yang digunakan meliputi buku referensi dan dokumentasi internet.

BAB III: METODOLOGI DAN PERANCANGAN

Pada bab ini dijelaskan metode-metode yang digunakan dalam menyelesaikan masalah penjadwalan mata kuliah menggunakan algoritma genetika.

BAB IV: HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan implementasi aplikasi, uji coba probabilitas perkawinan silang dan analisa hasil.

BAB V: PENUTUP

Bab lima berisi kesimpulan dari pembahasan dan saran yang diharapkan bermanfaat untuk pengembangan tugas akhir ini selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Algoritma Genetika

Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme seleksi alami dan genetika alami (Gen,1997).

Konsep dasar yang mengilhami timbulnya algoritma genetika adalah teori evolusi alam yang dikemukakan oleh Charles Darwin. Dalam teori tesebut dijelaskan bahwa pada proses evolusi alami, setiap individu harus melakukan adaptasi terhadap lingkungan disekitarnya agar dapat bertahan hidup (Setiawan,2003). Empat kondisi yang sangat mempengaruhi proses evaluasi :

- kemampuan organisme untuk melakukan reproduksi
- keberadaan populasi organisme yang bisa melakukan reproduksi
- keberagaman organisme dalam suatu populasi
- perbedaan kemampuan untuk bertahan

Individu yang lebih kuat akan memiliki tingkat ketahanan dan tingkat reproduksi yang lebih tinggi jika dibandingkan dengan individu yang kurang fit. Sehingga populasi secara keseluruhan akan lebih banyak memuat organisme yang fit (Kusumadewi,2003).

Algoritma genetika pertama kali dikembangkan pada tahun 1975 oleh John Holland bersama rekan kerja dan murid-muridnya dari Universitas Michigan. Pada penelitian tersebut dilakukan uji coba untuk memanfaatkan prinsip proses evolusi dalam suatu perangkat lunak untuk memecahkan suatu permasalahan optimasi (Setiawan,2003).

John Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika. Algoritma genetika adalah simulasi dari proses evolusi Darwin dan operasi genetika atas kromosom (Kusumadewi,2003).

2.2 Struktur Umum Algoritma Genetika

Algoritma genetika diilhami oleh ilmu genetika, karena itu istilah yang digunakan dalam algoritma genetika banyak diadaptasi dari ilmu tersebut. Pada algoritma genetika teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin yang dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi disebut

dengan istilah kromosom. Dan, setiap individu hanya mempunyai satu kromosom. Kromosom terdiri atas gen yang tersusun secara linier. Posisi yang ditempati gen dalam kromosom disebut *loci*, sedangkan nilai yang terdapat dalam gen disebut *alle* (Kusumadewi,2003) (Setiawan,2003).

Populasi awal dibangun dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut fungsi *fitness*. Fungsi *fitness* dapat berupa fungsi matematika atau fungsi lainnya dengan melihat permasalahan yang hendak diselesaikan. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dari populasi tersebut. Kromosom yang berkualitas baik mempunyai kemungkinan yang lebih besar untuk terpilih menjadi induk (*parent)* (Kusumadewi,2003) (Setiawan,2003).

Generasi berikutnya dikenal dengan istilah anak (*offspring*) yang terbentuk dari gabungan dua kromosom generasi sekarang yang bertindak sebagai induk dengan menggunakan operator penyilangan (*crossover*). Selain penyilangan, suatu kromosom dapat juga dimodifikasi dengan cara mutasi. Kondisi berhenti dari algoritma genetika apabila solusi yang diberikan telah konvergen atau jumlah generasi yang diminta telah tercapai (Kusumadewi,2003) (Setiawan,2003).

Secara garis besar, algoritma genetika dapat dijelaskan dengan algoritma berikut :

1. [**Mulai**] Membangun populasi secara random sebanyak n kromosom(sesuai dengan masalahnya)
2. **[Fitness**] Mengevaluasi setiap *fitness* f(x) dari setiap kromosom x pada populasi
3. [**Populasi baru**] Membuat populasi baru dengan mengulang langkah-langkah berikut sampai populasi baru lengkap
    1. [**Seleksi**] memilih dua kromosom induk dari populasi berdasarkan *fitness*nya (semakin besar *fitness*nya semakin besar kemungkinannya untuk terpilih)
    2. [**Perkawinan silang**] Sesuai dengan besarnya kemungkinan perkawinan silang, induk terpilih disilangkan untuk membentuk anak. Jika tidak ada

perkawinan silang, maka anak merupakan salinan dari induknya.

3. [**Mutasi**] Sesuai dengan besarnya kemungkinan mutasi, anak dimutasi pada setiap lokus (posisi pada kromosom)
4. [**Penerimaan**] tempatkan anak baru pada populasi baru

4. [**Ganti**] Gunakan populasi yang baru dibentuk untuk proses algoritma selanjutnya.
5. [**Tes**] jika kondisi akhir terpenuhi, berhenti, dan hasilnya adalah solusi terbaik dari populasi saat itu.
6. [**Ulangi**] Ke nomer 2.(Obitko,1999)

Apabila *P*(t) dan *C*(t) merupakan *parent* dan *offspring* pada generasi *t*, maka *pseudo code* dari algoritma genetika dapat dituliskan(Michalewicz,1999) :

```
procedure AlgoritmaGenetika
begin
  t ← 0;
  inisialisasi P(t);
  evaluasi P(t);
  while (bukan kondisi berhenti) do
    kombinasikan P(t) untuk menghasilkan
    C(t);
     evaluasi C(t);
    pilih P(t+1) dari P (t) dan C(t);
    t ← t+1;
  end while
end
```

2.3 Pengkodean

Langkah pertama pada algoritma genetika adalah menerjemahkan/ merepresentasikan masalah riil menjadi terminologi biologi. Cara untuk merepresentasikan masalah ke dalam bentuk kromosom disebut pengkodean. Terdapat bebarapa cara pengkodean, dan pemilihannya berdasarkan masalah yang dihadapi. Berikut adalah beberapa jenis pengkodean yang umum digunakan.

2.3.1 Pengkodean Biner

Pengkodean biner adalah pengkodean yang paling umum dan paling sederhana dalam merepresentasikan masalah pada algoritma genetika. Pada pengkodean biner setiap kromosom terdiri atas barisan string bit 0 atau 1. Sebagai contoh :

Tabel 2.1 Contoh kromosom pada pengkodean biner.

Kromosom A	0101101100010011
Kromosom B	1011010110110101

Contoh masalah yang sesuai menggunakan pengkodean biner adalah masalah nilai *maximize* pada sebuah fungsi matematika.

2.3.2 Pengkodean Permutasi

Pengkodean permutasi dapat digunakan pada masalah pengurutan data (*ordering problems*), seperti wiraniaga (*Travelling Salesman Problem*) atau masalah pengurutan tugas (*Task Ordering Problem*). Pada pengkodean permutasi, setiap kromosom terdiri dari barisan angka, yang merepresentasikan angka pada urutan.

Tabel 2.2 Contoh kromosom pada pengkodean permutasi.

Kromosom A	8549102367
Kromosom B	9102438576

2.3.3 Pengkodean Nilai

Pengkodean nilai dapat digunakan pada masalah yang sangat kompleks, dimana nilai yang dikodekan langsung merupakan representasi dari masalah. Penggunaan pengkodean biner pada tipe masalah yang kompleks akan menjadi lebih susah.

Pada pengkodean nilai, setiap kromosom adalah barisan dari beberapa nilai. Nilai dapat berupa apa saja, seperti bilangan biasa, bilangan riil, karakter sampai dengan obyek-obyek yang rumit.

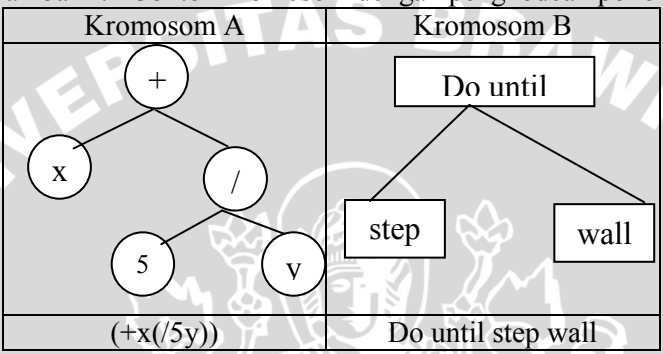
Tabel 2.3 Contoh kromosom pada pengkodean nilai.

Kromosom A	[red], [black], [blue], [yellow], [red]
Kromosom B	1.875, 3.9821, 9.1283, 6.8344, 4.116
Kromosom C	A, B, C, D, E, F, G, H, I, J, K, L, M, N

2.3.4 Pengkodean Pohon

Pengkodean pohon lebih banyak digunakan untuk menyusun program atau ekspresi, bagi pemrograman genetika (*genetic programming*). Pada pengkodean pohon, setiap kromosom merupakan pohon dari sejumlah obyek, seperti fungsi atau perintah pada bahasa pemrograman.

Gambar 2.1 Contoh kromosom dengan pengkodean pohon.



2.4 Seleksi

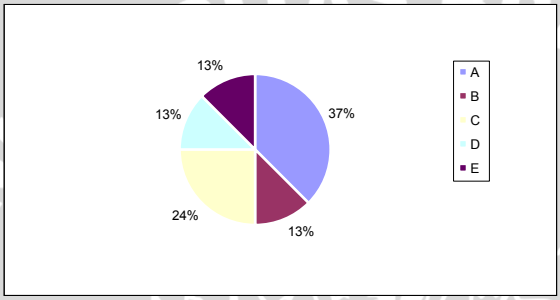
Proses seleksi bertanggung jawab untuk melakukan pemilihan terhadap individu yang hendak diikutkan dalam proses reproduksi. Langkah pertama yang dilakukan dalam seleksi ini adalah pencarian nilai fitness. Seleksi mempunyai tujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang mempunyai nilai *fitness* terbaik. Ada beberapa metode seleksi dari induk, antara lain :

2.4.1 Seleksi Roda *Roulette*

Metode seleksi roda *roulette* juga sering dikenal dengan nama *stochastic sampling with replacement*. Pada metode ini induk dipilih berdasarkan nilai *fitness*nya, semakin besar nilai *fitness* maka akan semakin besar kemungkinannya untuk terpilih menjadi induk. Dimisalkan semua kromosom diletakkan pada sebuah roda *roulette*, besarnya kemungkinan bagi setiap kromosom adalah tergantung dari nilai *fitness*nya seperti pada contoh berikut :

Tabel 2.4 Contoh populasi dengan 5 kromosom

Kromosom	*Fitness*
A	15
B	5
C	10
D	5
E	5



Gambar 2.2 Probabilitas suatu kromosom dalam roda *roulette*

Pada Tabel 2.4 merupakan contoh dalam satu populasi terdiri dari lima kromosom. Pada tiap kromosom memiliki nilai *fitness* yang berbeda-beda. Pada Gambar 2.2 dapat diketahui probabilitas terpilihnya masing-masing kromosom untuk menjadi induk. Pada kromosom A memiliki nilai *fitness* 15 dan nilai tersebut nilai *fitness* tertinggi pada populasi tersebut. Sehingga kromosom A memiliki probabilitas terbesar untuk terpilih menjadi induk.

2.4.2 Seleksi Rangking

Pada metode seleksi roda *roulette* akan bermasalah saat terdapat perbedaan *fitness* yang jauh. Sebagai contoh, jika *fitness* kromosom yang terbaik adalah 90% dari semua roda *roulette* dapat menyebabakan kromosom lain memiliki kesempatan yang kecil untuk dapat terpilih.

Proses dimulai dengan merangking atau mengurutkan kromosom di dalam populasi berdasarkan *fitness*nya kemudian memberi nilai *fitness* baru berdasarkan urutannya. Kromosom dengan nilai terburuk akan memiliki *fitness* baru nilai 1, terburuk kedua bernilai 2 dan begitu seterusnya, sehingga kromosom yang memiliki *fitness* terbaik

akan memiliki nilai *fitness* N, dimana N adalah jumlah kromosom di dalam populasi. Seperti dapat dilihat pada Tabel 2.5.

Tabel 2.5 Kromosom sebelum dirangking

Kromosom	*Fitness*
A	15
B	5
C	10
D	5
E	5

Tabel 2.6 Kromosom setelah dirangking

Kromosom	*Fitness*	*Fitness* baru
B	5	1
D	5	2
E	5	3
C	10	4
A	15	5

Setelah adanya proses seleksi tersebut, maka saat ini seluruh kromosom mempunyai kesempatan untuk dipilih. Akan tetapi, metode ini dapat menyebabkan konvergensi menjadi lambat, karena kromosom terbaik tidak terlalu berbeda dengan yang lainnya.

2.4.3 Seleksi Turnamen

Seleksi tunamen merupakan variasi antara seleksi roda *roulette* dan seleksi rangking. Sejumlah k kromosom tertentu dari populasi beranggota n kromosom ($k \leq n$) dipilih secara acak dengan probabilitas yang sama. Dari k kromosom yang terpilih kemudian akan dipilih satu kromosom dangan *fitness* terbaik, yang diperoleh dari hasil pengurutan rangking *fitness* semua kromosom terpilih. Perbedaan dengan seleksi roda *roulette* adalah pemilihan kromosom

yang akan digunakan untuk berkembangbiak tidak berdasarkan skala *fitness* dari populasi.

2.5 Operator Genetika

Setelah proses seleksi dilakukan, proses selanjutnya adalah operasi genetika. Terdapat 2 operator genetika, yaitu perkawinan silang dan mutasi.

2.5.1 Perkawinan Silang

Proses perkawinan silang (*crossover*) berfungsi untuk menghasilkan keturunan dari dua buah kromosom induk yang terpilih. Kromosom anak yang dihasilkan merupakan kombinasi gen-gen yang dimiliki oleh kromosom induk.

Secara umum, mekanisme tukar silang adalah sebagi berikut:

1. memilih dua buah kromosom sebagai induk.
2. memilih secara acak posisi dalam kromosom, biasa disebut titik perkawinan silang, sehingga masing-masing kromosom induk terpecah menjadi dua segmen.
3. lakukan pertukaran antar segmen kromosom induk untuk menghasilkan kromosom anak.

2.5.1.1 Perkawinan Silang untuk Pengkodean Biner

Pada pengkodean biner terdapat 4 metode penyilangan yang sering digunakan. Metode tersebut adalah:

a. Perkawinan silang satu titik.

Proses perkawinan satu titik dimulai dengan memilih satu titik pada barisan bit kromosom secara acak sebagai titik perkawinan silang. Kromosom baru akan dibentuk dengan cara menyalin barisan bit induk pertama dari bit pertama sampai titik perkawinan silang, sedangkan sisanya disalin dari induk kedua.

Contoh:

Misalkan ada 2 kromosom dengan panjang 12:

Induk 1: 01110 | 0101110

Induk 2: 11010 | 0001101

Posisi penyilangan yang terpilih: misalkan 5

Setelah penyilangan, diperoleh kromosom baru:

Anak 1: 01110 | 0001101

Anak 2: 11010 | 0101110

b. Perkawinan silang dua titik.
Perkawinan silang dua titik dimulai dengan menentukan secara acak dua titik yang akan disilangkan. Kromosom baru akan dibentuk dengan cara menyalin barisan bit kromosom induk pertama dari bit pertama sampai dengan dengan titik perkawinan silang pertama dari titik perkawinan silang kedua sampai dengan bit terakhir, sedangkan sisanya, yaitu titik perkawinan silang pertama sampai titik perkawinan silang kedua disalin dari induk kedua.
Contoh:
Misalkan ada 2 kromosom dengan panjang 12:
Induk 1: 011100101110
Induk 2: 110100001101
Posisi penyilangan yang terpilih: misalkan 3→ 2 6
Setelah penyilangan, diperoleh kromosom baru:
Anak 1: 01 | 0100 | 101110
Anak 2: 11 | 1100 | 001101

c. Perkawinan silang seragam
Sebuah *mask* penyilangan dibuat sepanjang panjang kromosom secara random yang menunjukkna bit-bit dalam *mask* yang mana induk akan mensuplay anak dengan bit-bit yang ada. Disini, anak_1 akan dihasilkan dari induk_1 jika bit *mask* bernilai 1, dan anak_1 akan dihasilkan dari induk_2 jika bit *mask* bernilai 0. Sedangkan anak_2 dihasilkan dari kebalikan *mask*.
Contoh:
Misalkan ada 2 kromosom dengan panjang 12:
Induk 1: 011100101110
Induk 2: 110100001101
Mask bit:
sampel 1: 100111001101
sampel 2: 011000110010
Setelah penyilangan, diperoleh kromosom baru:
Anak 1: 010100001100
Anak 2: 111100101111

d. Perkawinan silang aritmatik

Pada perkawinan silang aritmatik, digunakan beberapa operator aritmatika untuk mendapatkan *offspring* yang baru.

Contoh:

Induk 1	: 11001011
Induk 2	: 11011111
Anak (AND)	: 11001011

2.5.1.2 Perkawinan Silang untuk Pengkodean Permutasi

Pada pengkodean permutasi, perkawinan silang yang sering digunakan adalah perkawinan silang satu titik, karena selain prosesnya yang sederhana juga dapat menjaga konsistensi urutan nilai pada kromosom.

Proses perkawinan silang satu titik dimulai dengan pemilihan satu titik perkawinan silang. Dari permutasi pertama sampai dengan titik perkawinan silang disalin dari induk pertama, sedangkan sisanya didapatkan dengan cara melihat satu persatu nilai pada orang tua kedua, jika belum ada pada kromosom keturunan, maka nilai tersebut ditambahkan. Berikut contoh :

Induk 1	: **1 2 3 4 5** 6 7 8 9
Induk 2	: **4 5 3 6 8** 9 7 2 1
Anak (AND)	: **1 2 3 4 5** 6 8 7 9

2.5.1.3 Perkawinan Silang untuk Pengkodean Nilai

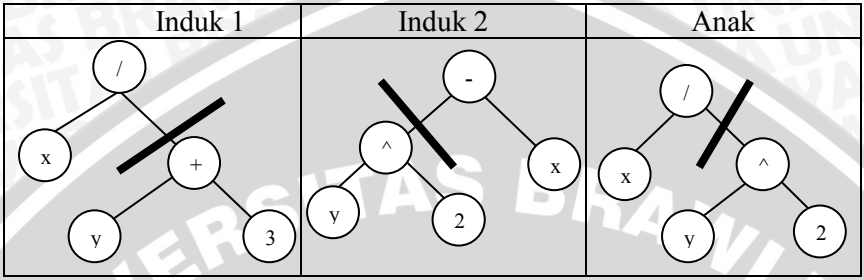
Semua metode perkawinan silang yang terdapat pada pengkodean biner dapat diterapkan pada pengkodean nilai. Hal tersebut dikarenakan pada pengkodean nilai tidak perlu diperhatikan urutan seperti pada pengkodean permutasi. Berikut contoh perkawinan silang pada pengkodean nilai:

Induk1	: **10 39 45 12** 9 65 12 76
Induk 2	: 45 86 75 12 **3 85 64 58**
Anak	: **10 39 45 12 3 85 64 58**

2.5.1.4 Perkawinan Silang untuk Pengkodean Pohon

Perkawinan silang untuk pengkodean pohon dimulai dengan pemilihan satu titik perkawinan silang pada induk, kemudian bagian

dari orang tua yang berada dibawah titik perkawinan silang dipertukarkan untuk menghasilkan anak baru



Gambar 2.3 Contoh perkawinan silang pada pengkodean pohon

2.5.2 Mutasi

Setelah melalui proses perkawinan silang, pada *offspring* dapat dilakukan proses mutasi. Mutasi dilakukan dengan cara melakukan perubahan pada sebuah gen atau lebih dari sebuah individu.

Tujuan dari mutasi adalah agar individu-individu yang ada dalam populasi semakin bervariasi. Mutasi akan sangat berperan jika pada populasi awal hanya ada sedikit solusi yang mungkin terpilih. Sehingga, operasi itu sangat berguna dalam mempertahankan keanekaragaman individu dalam populasi meskipun dengan mutasi tidak dapat diketahui apa yang terjadi pada individu baru.

Peluang mutasi mengendalikan banyaknya gen baru yang akan dimunculkan untuk dievaluasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi. Tetapi bila peluang mutasi terlalu besar, maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya.

2.6 Penjelasan Umum Algoritma Bottom-Left (Callaghan,1999)

Algoritma *Bottom-Left* terpenuhi apabila sudah tidak ada segi empat yang dapat digeser ke bawah atau ke kiri. Pola pengengepakan dapat direpresentasikan dari permutasi π sebagai berikut :

$\Pi = (i_1,\ldots,i_n)$ -> Permutasi

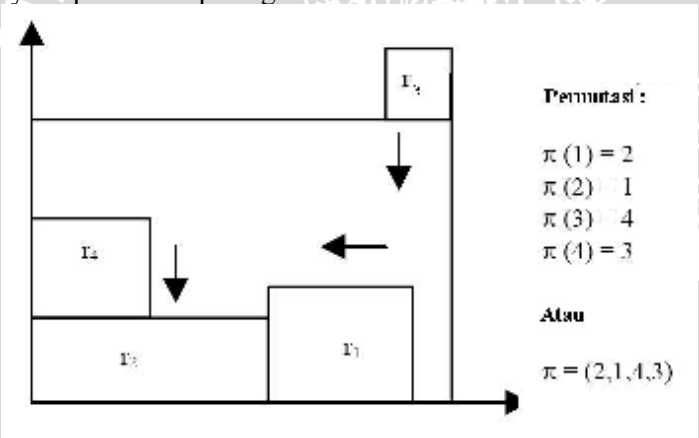
i -> indeks segi empat ($r_{i)}$)

Permutasi merepresentasikan urutan segi empat yang akan disusun. Liu mengembangkan Algoritma *Bottom-Left* sebagai Berikut :

Langkah 1 : tempatkan $r_{\pi(1)}$ pada sudut kiri dari papan optimasi (pada koordinat x=0 dan y=0)

Langkah i : Geser $r_{\pi(i)}$ secara bergantian, dimulai dari sudut kanan atas dari papan optimasi, lalu digeser sejauh mungkin ke arah bawah dan kemudian digeser sejauh mungkin ke arah kiri pada papan optimasi.

Sebagai contoh dapat terdapat 4 buah segi empat yang berbeda akan disusun ke dalam sebuah bidang datar segi empat. Untuk lebih jelasnya dapat dilihat pada gambar berikut :

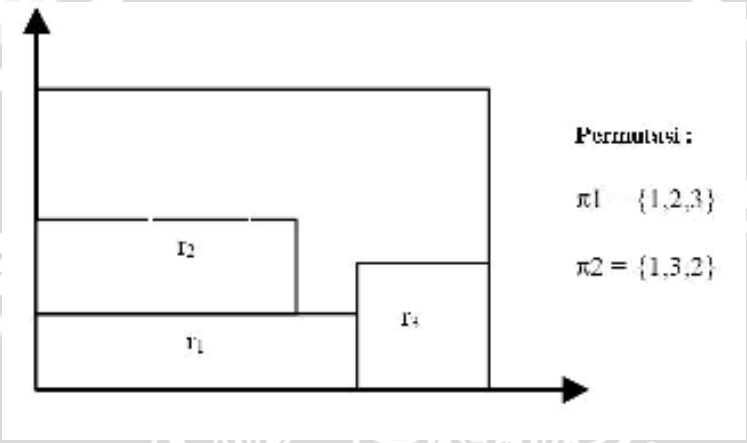


Gambar 2.4 Ilustrasi dari *BL-Algorithm*

Berdasarkan gambar 2.4, segi empat r_2 di tempatkan pertama kali ke bawah dan kemudian ke kiri sejauh mungkin. Segi empat r_1 dan r_4 ditempatkan dengan cara yang sama. Tanda panah menunjukkan pergerakan segi empat r_3 menuju lokasi yang optimal.

Menggunakan struktur data semacam ini, untuk n segi empat, n! Adalah batas atas (upper bound) dari jumlah pola pengepakan yang didapat dari metode algoritma *Bottom-Left* (Liu,1999). Ini menjelaskan bahwa *Orthogonal Packing Problem* adalah masalah permutasi. Tetapi karena ada dua sisi berbeda pada segi empat

sehingga tiap-tiap pola pengepakan dikalikan dengan 2^n (2 pangkat n). Sehingga batas atas $2^n.n!$ dimana n = jumlah segi empat. Tetapi pada prakteknya kurang dari $2^n.n!$ yang akan dibuat, karena dua permutasi dapat menghasilkan pola pengepakan yang sama. Seperti pada gambar 2.5 :



Gambar 2.5 Dua permutasi mengahasilkan pola yang sama

Berdasarkan permutasi π_1 karena lebar r_2 lebih besar daripada ruang yang tersedia di daerah paling bawah, maka r_2 diletakkan di atas r_1. kemudian baru r_3 di tempatkan. Berdasarkan permutasi π_2, r_1 , juga diletakkan pertama kali, diikuti dengan segi empat r_3. Kali ini r_3 muat pada ruang di daerah paling bawah dan r_2 diletakkan diatas r_1. Karena jumlah permutasi yang mungkin sangat besar, algoritma genetik sangat sesuai untuk menyelesaikan masalah ini.


UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas lebih detail mengenai metodologi dan rancangan yang digunakan dalam menentukan pengepakan yang paling optimal pada skripsi ini. Tujuan dari pembahasan ini adalah untuk memberikan gambaran dan penjelasan kepada *user* mengenai sistem serta memberikan gambaran yang jelas tentang program yang telah dibuat berdasarkan tinjauan pustaka pada bab dua.

3.1 Deskripsi Umum Sistem

Sistem yang aka dibuat merupakan sitem untuk mencari bentuk pengepakan yang paling optimal pada pengepakan segi empat. Aplikasi ini dapat digunakan dalam industri-industri yang membutuhkan pengepakan pada bidang datar seperti industri plat baja dan industri tekstil. Optimal dalam hal ini adalah luas sisa yang di dapat besar dan meminimalkan ruang yang terbuang. Sehingga dapat mengurangi pengeluaran karena banyaknya item yang terbuang sia-sia. Sistem ini dibuat untuk memudahkan perusahaan melakukan optimasi segi empat pada bidang datar. Diharapkan dengan adanya sistem ini dapat membantu perusahan dalam melakukan pengepakan dan mengurangi pengeluaran yang tidak perlu.

Dalam sistem ini *user* melakukan masukan berupa jumlah segi empat yang akan di letakkan beserta informasi panjang dan lebarnya, dan juga informasi panjang dan lebar bidang datar yang akan digunakan sebagai tempat peletakan. Sistem akan mengolah data tersebut dan akan membuat keluaran berupa gambar peletakan yang paling optimal.

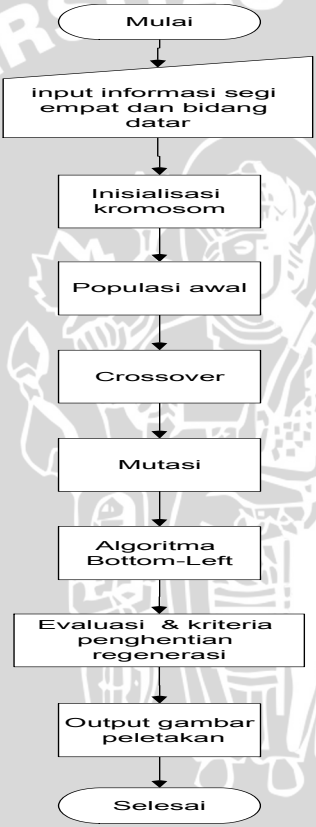
3.3 Perancangan Sistem

Dalam perancangan program perangkat lunak pada tugas akhir ini penulis mengimplementasikan rancangan *interface*, yang terdiri dari empat tahapan dalam proses, untuk menghasilkan *output* yang diinginkan tahapan rancangan antarmuka tersebut, adalah:

1. *User* memasukkan inputan berupa jumlah segi empat yang akan di letakkan beserta informasi panjang dan lebarnya, dan juga informasi panjang dan lebar bidang datar yang akan digunakan sebagai tempat peletakan.

2. Membentuk kromosom yang akan digunakan sebagai populasi awal.
3. Melakukan proses algorima genetik yang dikombinasi dengan algoritma *bottom-left.*
4. *Output* berupa gambar hasil peletakan yang paling optimal

Pada **Gambar 3.1** akan menjelaskan diagram alir dari sistem yang berisikan keseluruhan proses.



Gambar 3.1 *Flowchart* alur proses sistem

3.4 Optimasi Pengepakan Dengan Algoritma Genetik

3.4.1 Pengkodean Kromosom

Pengkodean yang akan digunakan pada pengepakan ini adalah pengkodean nilai. Pemilihan pengkodean nilai karena jika dilakukan pengkodean biner akan membuat barisan bit semakin kompleks, dimana suatu barisan bit yang panjang sehingga diperlukan pengkodean ulang untuk mendapatkan nilai sebenarnya dari aturan yang direpresentasikan.

Pada masalah pengepakan ini dibuat kromosom berdasarkan urutan kotak yang akan di tata menggunakan algoritma *Bottom-Left*. Pengkodean kromosom dibuat sederhana sehingga pada tahap mutasi dan *crossover* tidak memerlukan evaluasi yang rumit.

Kromosom akan dibuat dengan array satu dimensi berisi urutan penataan. Tiap-tiap gen berisi nama kotak dengan status kotak. Status kotak disini menggunakan simbol minus (-) atau tidak sebagai tanda kotak dirotasi 90^o.

Kotak yang dirotasi 90^o maka nilai panjang dan lebarnya akan ditukar. Nilai panjang akan di ganti dengan nilai lebarnya dan nilai lebar akan diganti dengan nilai panjangnya.

Kromosom akan dibuat berdasarkan urutan penyususan sebagai contoh (1,3,2,-4) , (3,4,1,2) , (3,2,-1,4). Dengan inputan nilai yang sama dengan tabel 3.1 berikut bentuk kromosom yang akan dibuat

$\prod_i \rightarrow$

1	3	2	-4

Pada gen ke-4 berisi nilai -4 (minus empat) yang berarti kotak ke empat panjang dan lebarnya ditukar. Nilai panjangnya menjadi 10 dan lebarnya menjadi 50.

3.4.2 Inisialisasi Kromosom

Setelah terbentuknya individu-individu proses selanjutnya adalah Inisialisasi kromosom. Dalam inisialisasi ini kromosom direpresentasikan dalam bentuk *array* dengan tipe data *integer* yang berisi urutan peletakan segi empat. Panjang dari kromosom adalah sebanyak segi empat yang akan diletakkan.

3.4.3 Populasi Awal

Pemilihan populasi awal berpengaruh cukup besar terhadap hasil optimasi yang diperoleh. Sehingga dibutuhkan pemilihan populasi awal yang baik dan mendekati hasil yang optimum.

Untuk populasi awal diurutkan nilai panjang dan lebar kotak mulai dari yang terkecil digunakan sebagai nilai horisontal segi empat. Urutan tersebut kemudian digunakan sebagai kromosom individu pertama ($\prod_1$).

Selain itu diurutkan juga nilai panjang dan lebar kotak mulai dari yang terbesar digunakan sebagai nilai horisontal segi empat. Urutan tersebut digunakan sebagai kromosom indidu kedua ($\prod_2$).

Pada kromosom ketiga diurutkan juga nilai panjang dan lebar kotak mulai dari yang terkecil digunakan sebagai nilai vertikal segi empat Urutan tersebut digunakan sebagai kromosom indidu ketiga ($\prod_3$).

Pada kromosom keempat diurutkan juga nilai panjang dan lebar kotak mulai dari yang terbesar digunakan sebagai nilai vertikal segi empat Urutan tersebut digunakan sebagai kromosom indidu keempat ($\prod_4$).

Untuk kromosom individu kelima dan seterusnya akan diambil secara acak. Berikut contoh hasil populasi awal dari data tabel 3.1. :

Tabel 3.1 Tabel struktur data

Kotak	Panjang	Lebar
1	25	15
2	20	30
3	60	20
4	50	10

$\prod_1$ → | -4 | -1 | 2 | -3 |

$\prod_2$ → | 3 | 4 | -2 | 1 |

$\prod_3$ → | 4 | 1 | -2 | 3 |

$\prod_4$ → | -3 | -4 | 2 | -1 |

3.4.4 Seleksi

Seleksi mempunyai peranan penting dalam algoritma genetika, karena pada proses ini dipilih induk yang digunakan untuk menghasilkan individu baru. Seleksi yang digunakan adalah seleksi *ranking* dan seleksi roda *roulette*. Seleksi *rangking* digunakan untuk menentukan individu terbaik dan sekaligus individu pertama pada populasi selanjutnya.

Sedangkan seleksi roda *roulette* digunakan untuk menentukan individu kedua dan seterusnya pada populasi selanjutnya sehingga setiap kromosom memiliki kemungkinan untuk muncul pada populasi selanjutnya. Funsinya adalah agar *variasi* individu yang dimiliki tetap beragam.

3.4.5 Operator *Crossover*

Apabila proses seleksi telah dilaksanakan dan sudah terpilih induk baru, maka operator berikutnya adalah *crossover*. *Crossover* adalah cara mengkombinasikan gen-gen induk untuk menghasilkan keturunan baru.

Proses *crossoever* yang digunakan adalah *two-Cut-Point crossover*. Proses ini digunakan untuk menukar gen pada posisi tertentu kedua induk.Dalam proses *crossover* dipilih secara acak nomer gen awal sebagai posisi *crossover*(P). Kemudian di acak jumlah gen yang akan di*crossover* (Q). Agar hasil penukaran sesuai dengan aturan kromosom yang berlaku, pada kromosom kedua dicari gen-gen yang belum terdapat pada kromosom hasil.

$\prod_i = (1,3,2,6,5,4)$
$\prod_J = (6,4,2,5,3,1)$

Random P = 2 dan Q = 3

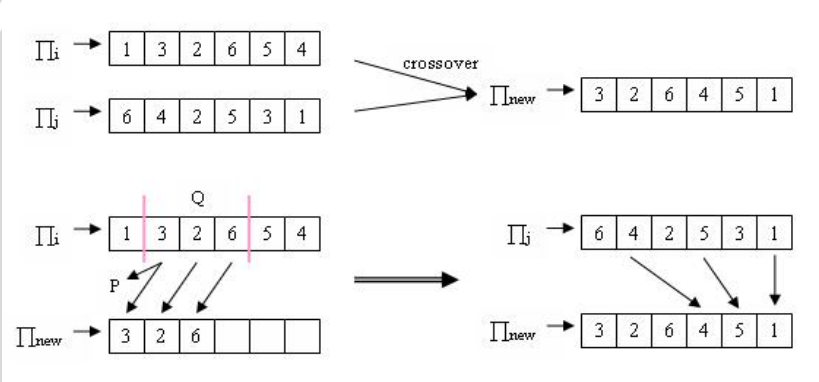
P = nomer gen awal yang akan di crossover
Q = jumlah gen yang akan di crossover

Diperoleh :

$\prod_{new}(1) = \prod_i(P) = \prod_i(2) = 3$
$\prod_{new}(2) = \prod_i(P+1) = \prod_i(3) = 2$
$\prod_{new}(3) = \prod_i(P+2) = \prod_i(4) = 6$

Kemudian $\prod_{new}$ diisi dengan gen-gen dari $\prod_j$ selain yang diatas

$\prod_{new}(4) = \prod_j(2) = 4$
$\prod_{new}(5) = \prod_j(4) = 5$
$\prod_{new}(6) = \prod_j(6) = 1$
Sehingga terbentuk kromosom baru yaitu $\prod_{new} = (3,2,6,4,5,1)$



Gambar 3.2 Ilustrasi *crossover*

Dengan sistem *crossover* seperti diatas dapat menghindari terjadinya kromosom ilegal. Sehingga tidak memerlukan proses pengecekan terhadap kromosom hasil *crossover*.

3.4.6 Operator Mutasi

Untuk operator mutasi digunakan dua jenis mutasi yaitu normal mutasi dan rotate mutasi.

3.4.6.1 Normal Mutasi

Normal mutasi dapat dilakukan dengan dua cara, yaitu cara *inversion* dan cara *swap* atau penukaran. Mutasi cara pertama adalah dengan menentukan dua titik batas atas dan batas bawah. Setelah itu nilai yang terdapat pada batas-batas tersebut dibalik urutannya. Pada

cara kedua adalah dengan menukar langsung nilai dari gen. Pemilihan cara mutasi dilakukan secara random.
Ilustrasi mutasi cara *inversion*:

Dimisalkan batas bawah : 3 dan batas atas : 6

Sebelum mutasi:

3	2	**1**	**5**	**4**	**7**	6

Sesudah mutasi:

3	2	**7**	**4**	**5**	**1**	6

Gambar 3.3 Ilustrasi mutasi cara *inversion*

Ilustrasi mutasi cara *swap*:

Sebelum mutasi:

3	4	**6**	2	7	**5**	1

Sesudah mutasi:

3	4	**5**	2	7	**6**	1

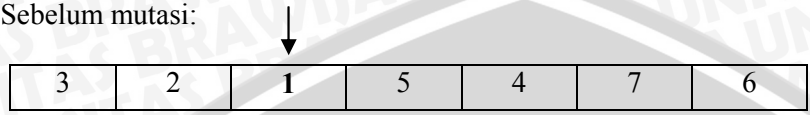
Gambar 3.4 Ilustrasi mutasi cara *swap*

3.4.6.2 Rotate Mutasi

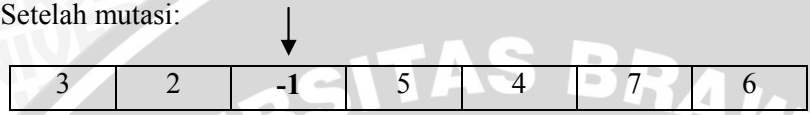
Rotate mutasi digunakan untuk memutar segi empat sejauh 90^o. sehingga diperoleh kromosom yang lebih bervariasi. Caranya dengan mengacak nilai gen yang akan dimutasi setelah itu gen tersebut diberi tanda minus (-) jika belum ada tanda minusnya atau sebaliknya.

Dimisalkan didapat nilai acak : 3

Sebelum mutasi:



Setelah mutasi:

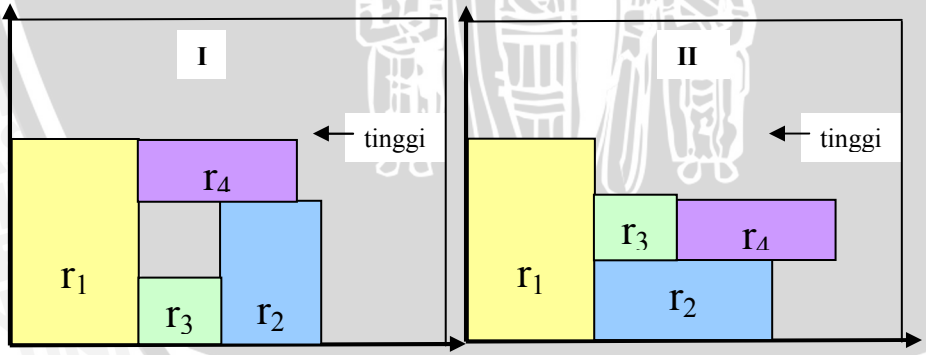


Gambar 3.5 Ilustrasi Rotate Mutasi

3.4.7 Evaluasi Pola Pengepakan (Fungsi Fitnes)

Untuk mengevaluasi kromosom yang dihasilkan digunakan algoritma *Bottom-Left* untuk meletakkan segi empat pada bidang datar. Proses pada algoritma *Bottom-Left* telah dijelaskan pada bab 2. Setelah digunakan algoritma *Bottom-Left*, dapat dihitung tinggi maksimal darikotak yang disusun. Tinggi maksimal dihitung sebagai fitnes dari tiap-tiap populasi.

Karena ada kemungkinan ada dua individu yang memiliki tinggi maksimal yang sama maka dihitung juga luas sisa bidang penyusunan sebagai fitnes. Sebagai contoh dua individu yaitu populasi I (1,3,2,4) dan populasi II (1,-2,3,4) memiliki tinggi yang sama sepeti pada gambar berikut



Gambar 3.6 Contoh dua populasi dengan tinggi sama

Berdasarkan gambar 3.6 kedua individu memiliki tinggi maksimal yang sama. Tetapi setelah dihitung luas sisa ternyata individu II memiliki luas sisa lebih besar sehingga dipilih individu II sebagai populasi yang paling optimum.

Penghitungan luas sisa dihitung dari luar kotak-kotak yang disusun. Sehingga individu dengan penyusunan yang memberikan banyak ruang kosong di antara kotak yang disusun dianggap kurang baik.

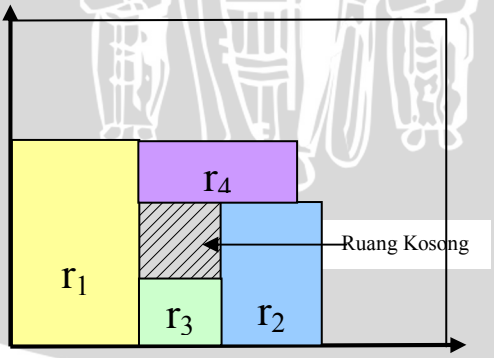
Berdasarkan perhitungan yang lebih memberatkan pada tinggi dari peletakan dari pada luas sisa maka didapat humus sebagai berikut :

$$f(g) = 2(T - t) + \frac{L}{l}$$

Perhitungan fungsi *fitness* lebih menekankan kepada tinggi maksimal peletakan(t). Sehingga dicari tinggi sisa yang didapat dari tinggi bidang datar (T) dikurangi tinggi maksimal (t) kemudian dikalikan dua (2) untuk lebih menekankan prioritasnya. Kemudian ditambah dengan luas sisa (L) dibagi lebar bidang datar (l), untuk menjaga jika ada individu yang memiliki tinggi maksimal yang ada.

3.4.8 Pencarian Luas Sisa

Luas Sisa adalah luas bidang datar dikurangi jumah luas segi empat yang diletakkan dan jumlah luas ruang kosong yang berada ditengah-tengah antara segi empat yang diletakkan.



Gambar 3.7 Contoh ruang kosong

Berdasarkan gambar 3.7, maka sistem perlu mendeteksi letak dan luas ruang kosong yang ada. Sehingga sistem dapat menghitung jumlah luas sisa yang dihasilkan.

Maka dilakukan deteksi ruang kosong setelah penggunaan algoritma *Bottom-Left* dijalankan. Cara nya dengan mendeteksi bidang datar setelah dilakukan algoritma *Bottom-Left*. Berdasakan gambar 3.7 maka akan didapat array sebagai berikut.

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

Gambar 3.8 Contoh array hasil *bottom-left*

Dari gambar diatas ditunjukkan bahwa bidang yang telah terisi dengan segi empat ditunjukkan dengan angka 1 (satu) sedangkan yang kosong diisi dengan angka 0 (nol). Deteksi dilakukan dengan memeriksa bidang yang berisi angka 0 (nol). Bidang yang dianggap ruang kosong akan diisi dengan angka 9 (sembilan)

Setelah ditemukan bidang yang bernilai 0 (nol) maka akan dilakukan pengecekan nilai dari bidang di sebelah atas dan bidang di sebelah kirinya, apabila bidang di sebelah atas dan di sebelah kirinya benilai 1 (satu) atau 9 (sembilan) maka bidang tersebut dianggap sebagai ruang kosong awal.

Kemudian dilakukan proses tersebut hingga ditemukan ruang kosong akhir dengan cara memeriksa bidang disebelah kanan bidang yang bidang di sebelah atas bernilai 1 (satu) atau 9 (sembilan) dan bidang disebelah kanannya bernilai 1 (satu), maka bidang tersebut dianggap sebagai ruang kosong akhir.

Bidang mulai dari ruang kosong awal hingga ruang kosong akhir nilainya kemudian diganti dengan angka 9 (sembilan). Sehingga bidang yang dianggap sebagai ruang kosong. Dari gambar 3.8 akan menghasilkan array sebagai berikut.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	9	9	9	9	9	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	9	9	9	9	9	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	9	9	9	9	9	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	9	9	9	9	9	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

Gambar 3.9 Contoh array hasil pencarian ruang kosong.

Setelah didapat hasil seperti gambar 3.9 pencarian luas sisa akan lebih mudah. Dilakukan dengan menghitung jumlah bidang yang bernilai 0 (nol) sehingga didapat nilai dari luas sisa.

3.4.9 Berhentinya Algoritma Genetika

Prinsip kerja algoritma genetika adalah berulang-ulang sampai ketentuan yang ditetapkan. Pada kasus ini algoritma genetik akan berhenti setelah mencapai generasi tertentu yang telah ditentukan.

3.5 Contoh Permasalahan

Contoh permasalahan pada pengepakan segi empat. Berikut ini tabel yang berisi informasi segi empat yang akan diletakkan.

Tabel 3.2 Contoh permasalahan

Nama	Panjang	Lebar
1	20	10
2	15	7
3	13	5
4	20	15

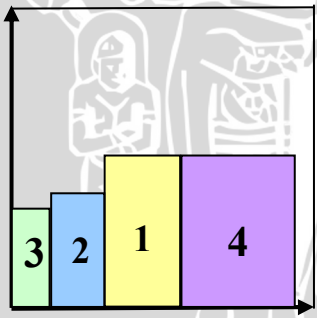
Selain itu juga terdapat informasi tambahan berupa panjang dan lebar bidang datar tempat peletakan. Pada kasus ini dimasukkan nilai panjang bidang datar adalah 40 (empat puluh) dan lebarnya 40 (empat puluh).

Setelah data terkumpul dilakukan proses pembentukan populasi awal hingga didapat empat individu.

Tabel 3.3 Populasi awal

Individu	Kromosom
1	-3 -2 -1 -4
2	1 4 2 3
3	2 -1 -3 4
4	4 1 -2 3

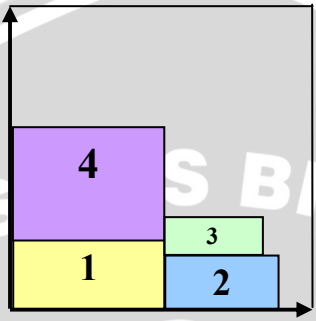
Untuk individu 1 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.10 Hasil Bottom-Left individu 1

Nilai fitnesnya : f(1) = 2(40-20)+930/40
= 40+ 23,25
= 63,25

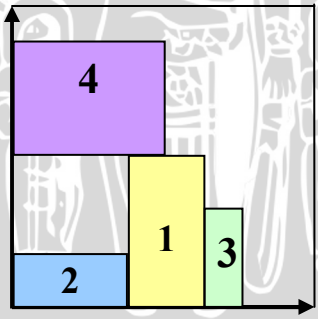
Untuk individu 2 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.11 Hasil Bottom-Left individu 2

Nilai fitnesnya : f(1) = 2(40-25)+930/40
= 30+23,25
= 53,25

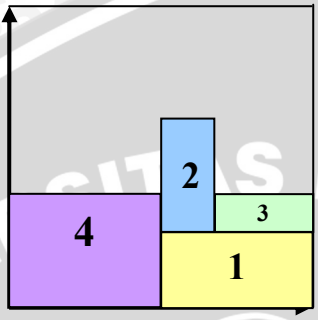
Untuk individu 3 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.12 Hasil Bottom-Left individu 3

Nilai fitnesnya : f(1) = 2(40-35)+735/40
= 10+18,375
= 28,375

Untuk individu 4 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.13 Hasil Bottom-Left individu 4

Nilai fitnesnya : f(1) = 2(40-25)+930/40
= 30+23,25
= 53,25

Dari individu pada **populasi** dipilih secara random dua individu untuk proses crossover, misal kromosom yang dipilih adalah kromosm dua dan tiga, untuk lebih jelasnya dapat dilihat proses berikut

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

Kemudian dirandom nilai kromosom awal dan jumlah gen yang akan di *crossover* pada kasus ini didapat nilai kromosom awal dua dan jumlah gen dua

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

Maka didapat

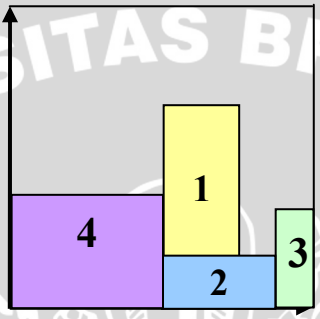
child 1

4	2	-1	-3

child 2

-1	-3	2	4

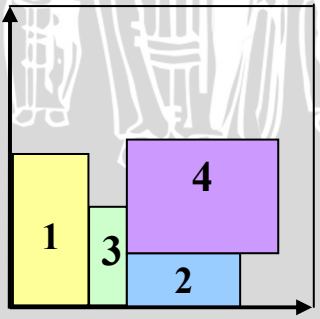
Untuk child 1 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.14 Hasil Bottom-Left Child 1

Nilai fitnesnya : f(1) = 2(40-27)+930/40
= 26+23,25
= 49,25

Untuk child 2 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.15 Hasil Bottom-Left Child 2

Nilai fitnesnya : f(1) = 2(40-22)+930/40
= 36+23,25
= 59,25

Kemudian dipilih secara random satu individu untuk proses normal mutasi, misal kromosom yang dipilih adalah kromosm satu, untuk lebih jelasnya dapat dilihat proses berikut

Parent 1	-3	-2	-1	-4

Lalu dirandom dua nilai sebagai batas atas dan batas bawah. Pada kasus ini didapat nilai satu dan empat. Nilai ini digunakan sebagai batas atas dan bawah pada *inversion* dan nomer gen yang aka ditukar pada *swap*

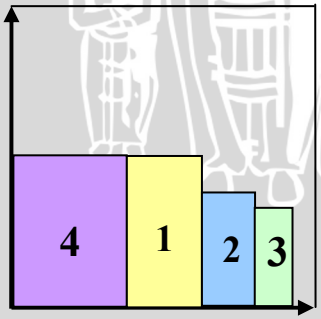
Inversion

Child 3	-4	-1	-2	-3

Swap

Child 4	-4	-2	-1	-3

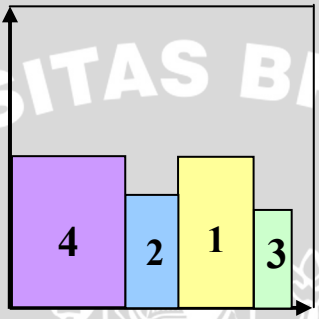
Untuk child 3 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.16 Hasil Bottom-Left Child 3

Nilai fitnesnya : f(1) = 2(40-20)+930/40
= 40+ 23,25
= 63,25

Untuk child 4 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.17 Hasil Bottom-Left Child 4

Nilai fitnesnya : f(1) = 2(40-20)+930/40
= 40+ 23,25
= 63,25

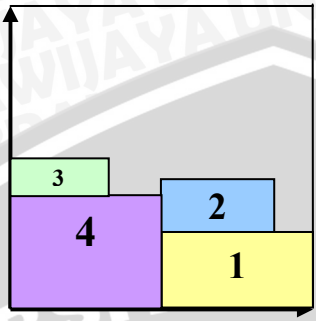
Kemudian dilakukan pemilihan random lagi pada individu populasi untuk digunakan pada proses mutasi rotasi, pada kasus ini dipilih individu empat. Dan juga dipilih secara random gen yang akan dirotasi, pada kasus ini dipilih gen ke tiga.

Parent 1	4	1	-2	3

Maka dihasilkan *child* seperti berikut

child 5	4	1	2	3

Untuk *child* 5 dilakukan metode bottom-left sehingga mendapat hasil sebagai berikut :



Gambar 3.18 Hasil Bottom-Left Child 5

Nilai fitnesnya : f(1) = 2(40-20)+930/40
= 40+23,25
= 63,25

Setelah dilakukan satu itersi pada contoh kasus ini dengan menggunakan proses *crossover* dan *mutasi* diperoleh hasil seperti pada tabel 3.4

Tabel 3.4 Hasil iterasi pertama

Individu	Kromosom	*Fitnes*
1	-3 -2 -1 -4	63,25
2	1 4 2 3	53,25
3	2 -1 -3 4	28,375
4	4 1 -2 3	53,25
Child 1	4 2 -1 -3	49,25
Child 2	-1 -3 2 4	*59,25*
Child 3	-4 -1 -2 -3	*63,25*
Child 4	-4 -2 -1 -3	*63,25*
Child 5	4 1 2 3	63,25
	Jumlah	496,375

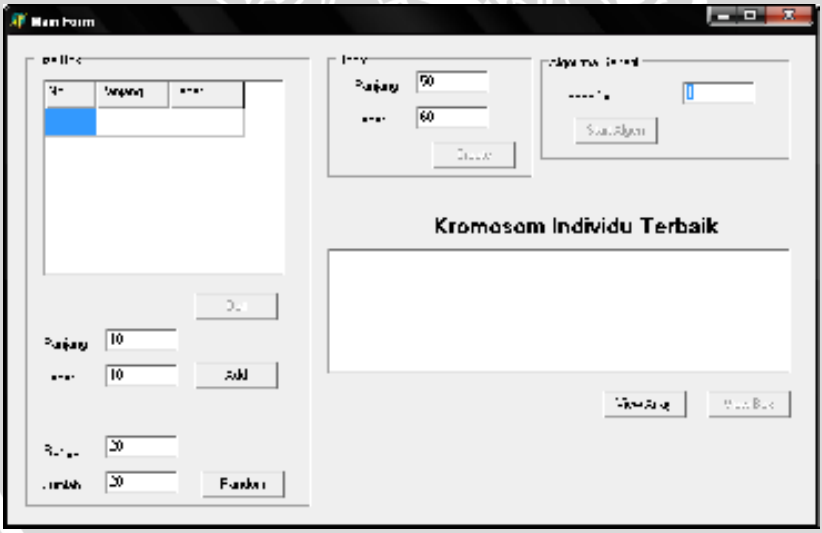
BAB IV
IMPLEMENTASI DAN UJI COBA SISTEM

4.1 Implementasi

Aplikasi optimasi pengepakan segi empat menggunakan algoritma genetik dan algoritma bottom-left hanya memiliki satu bentuk tampilan utama yang juga merupakan halaman input data.

4.1.1 Input Data

Input data terdiri dari komponen-komponen yang dibutuhkan dalam optimasi pengepakan segi empat menggunakan algooritma genetik dan algoritma *Bottom-Left* yang terdiri dari jumlah segi empat yang akan di optimasi, ukuran segi empat yang akan di optimasi, dan ukuran bidang peletakan segi empat. Selain itu di*input*kan juga jumlah iterasi yang akan dilakukan. Tampilan setelah dilakukan input data akan menjadi seperti berikut:



Gambar 4.1 Form Input Data

4.1.2 Deskripsi Program

4.1.2.1 Struktur Data

Struktur data akan dibagi menjadi tiga(3) kelompok. Yaitu stuktur data algoritma genetik, struktur data bidang, dan struktur data segi empat. Struktur data algoritma genetik direpresentasikan sebagai berikut :

```
boxFinal:array[1..100] of kotak;
kandidatBaru:integer;
range :integer;
jml_kromosom:integer;
kromosom:array[0..10,1..100] of integer;
fitnes:array[1..10] of real;
```

Gambar 4.2 Struktur data algoritma genetik

Tabel 4.1 Keterangan Struktur data algoritma genetik

boxFinal	Digunakan untuk membantu dalam menampilkan hasil akhir proses optimasi
kandidatBaru	Digunakan untuk menghitung jumlah individu baru
range	Jumlah gen dalam satu (1) kromosom
jml_kromosom	Jumlah kromosom dalam satu (1) populasi
kromosom	Untuk menyimpan data kromosom, nomer kromosom dan gennya.
fitnes	Untuk menyimpan fitness tiap-tiap kromosom

Berikut representasi dari struktur data bidang:

```
room = class
public
  value :array[1..1000,1..1000] of Integer;
  Panjang,Lebar :integer;
  boxSave:array[1..100] of kotak;
```

Gambar 4.3 Struktur data bidang

Tabel 4.2 Keterangan struktur data bidang

value	Menyimpan array bidang
Panjang,Lebar	Menyimpan ukuran panjang dan lebar bidang
boxSave	Variable pembantu dalam menampilkan hasil dari proses optimasi

Berikut reprensentasi dari struktur data segi empat:

```
type
  luas = class
  private
     panjangLebar: array [1..100,1..2] of integer;
  public
     jmlArray:integer;
     count:integer;
  end;

type
  kotak = class
  public
    panjang,lebar :integer;
    posx,posy:integer;
    nama   : string;
    value:integer;
  end;

type
  setkotak = class
  private
     setBox: array [1..100] of kotak;
  public
     jumlahKotak:integer;
     count:integer;
  end;
```

Gambar 4.4 Struktur data segi empat

Tabel 4.3 Keterangan struktur data segi empat

luas	Menampung ukuran panjang dan lebar segi empat
kotak	Perwujudan segi empat dari ukuran panjang dan lebar
Setkotak	Untuk menampung kumpulan segi empat

4.1.2.2 Pembentukan Populasi Awal

Pembentukan populasi awal dilakukan dengan dua (2) cara. Yaitu dengan pembentukan individu Hybrid dan individu random.

Individu *hybrid* didapatkan dengan mengurutkan terlebih dahulu nilai-nilai panjang dan lebar segi empat yang akan dioptimasi. Untuk individu pertama diambil nilai segi empat dari terkecil hingga terbesar digunakan sebagai nilai panjang segi empat yang akan dioptimasi. Untuk individu kedua diambil nilai segi empat dari yang terbesar digunakan untuk nilai panjang segi empat yang akan dioptimasi. Sedang untuk individu ketiga dan keempat dilakukan hal yang sama seperti individu satu(1) dan dua(2), tetapi digunakan sebagai nilai lebar segi empat yang akan dioptimasi.

```
countHybrid:=range*2;
SetLength(hybrid,countHybrid+1);
SetLength(tempKromosom,range+1);
for i:=1 to range do
  hybrid[i]:=size.viewPanjang(i);
for j:=1 to range do
  hybrid[j+range]:=size.viewLebar(j);

for i:=1 to countHybrid-1 do
  for j:=i+1 to countHybrid do
    if(hybrid[i] > hybrid[j]) then
    begin
      hybrid[0]:=hybrid[i];
      hybrid[i]:=hybrid[j];
      hybrid[j]:=hybrid[0];
    end;
```

Gambar 4.5 Pengurutan nilai panjang dan lebar segi empat

Setelah didapatkan nilai panjang dan lebar segi empat yang telah diurutkan. Kemudian diambil nilai dari yang terkecil hingga terbesar untuk untuk individu 1 dan 3. Setelah itu diambil nilai dari yang terbesar hingga terkecil untuk individu 2 dan 4. Berikut contoh *source* code inisialisasi individu 1 dan 3.

```
  //inisialisasi individu hybrid 1 dan 3
  n:=1;
  countK:=0;
  for i:=1 to countHybrid do
  begin
     tempH:=hybrid[i];
     tempK:=0;
     for j:=1 to range do
     begin
       if(tempH=size.viewPanjang(j))  then
          tempK:=j
       else if (tempH=size.viewLebar(j)) then
          tempK:=j*(-1);
       if(tempK<>0) then
       begin
         ada:=0;
         for k:=1 to countK do
         begin
if(tempKromosom[k]=tempK)or(tempKromosom[k]=tempK*(-
1))  then
           begin
             ada:=1;
             break;
           end;
         end;

         if(ada=1) then
           continue
         else
           break;
       end;
     end;
     if(ada=0) then
     begin
         countK:=countK+1;
         tempKromosom[countK]:=tempK;
         kromosom[n][countK]:=tempK;
         kromosom[n+2][countK]:=tempK*(-1);
     end;
  end;
  for k:=1 to countK do
     tempKromosom[k]:=0;
```

Gambar 4.6 Insialisasi individu 1 dan 3

Setelah didapat individu 1 dan 3 diteruskan kepada individu 2 dan 4 dengan menggunakan cara yang sama sebagai nilai lebar dari segi empat yang dioptimasi.

Populasi random didapatkan dengan melakukan randomisasi tiap-tiap gen dalam kromosom. Tiap gen mewakili segiempat yang akan dioptimasi. Berikut contoh source code untuk menghasilkan individu random.

```
procedure algen.firstGenerateRandom();
var j,i,count,hasil,ada,ismin:integer;
begin
randomize;
for j:=1 to jml_kromosom do
begin
  count:=1;
  while(count<=range) do
  begin
    hasil:=random(range)+1;
    ada:=0;
    for i:=1 to count do
    begin

if((kromosom[j][i]=hasil)or(kromosom[j][i]=hasil*(-
1))) then
      begin
        ada:=1;
        break;
      end;
    end;

    if(ada=0) then
    begin
      ismin:=random(100);
      if(ismin<50) then
         kromosom[j][count]:=hasil*(-1)
      else
         kromosom[j][count]:=hasil;
      count:=count+1;
    end;
  end;
  countFitnes(j);

end;
end;
```

Gambar 4.7 Generate individu random

4.1.2.3 Perkawinan Silang

Setelah didapat individu ayah dan individu ibu melalui proses *randomisasi*. Dilakukan randomisasi nilai p dan q sebagai nilai awal dan akhir kromosom yang akan dikawin silangkan. Kemudian proses kawin silang seperti yang telah dijelaskan pada bab 3. Berikut source code untuk porses kawin silang untuk mengahasilkan *child* 1.

```
//child 1
count:=0;
while(count<q) do
begin
  count:=count+1;
  kromosom[0][count]:=kromosom[ayah][p+count-1];
end;
for i:=1 to range do
begin
  ada:=0;
  for j:=1 to count do

if((kromosom[0][j]=kromosom[ibu][i])or(kromosom[0][j
]*(-1)=kromosom[ibu][i])) then
    begin
      ada:=1;
      break;
    end;

  if(ada=0) then
  begin
    count:=count+1;
    kromosom[0][count]:=kromosom[ibu][i]
  end;
end;
 kandidatBaru:=kandidatBaru+1;
 kromosom[jml_kromosom+kandidatBaru]:=kromosom[0];
 countFitnes(jml_kromosom+kandidatBaru);
//clear temp
for i:=1 to range do
  kromosom[0][i]:=0;
```

Gambar 4.8 Proses kawin silang

setelah didapat *child* 1 di gunakan cara yang sama pada kromosom ibu untuk menghasilkan *child* 2.

4.1.2.4 Mutasi

Langkah selanjutnya adalah melakukan mutasi. Dilakukan dua(2) proses mutasi untuk mendapatkan individu anak, yaitu mutasi normal dan mutasi rotasi . Proses mutasi dilakukan setelah menentukan kromosom yang akan dimutasi dengan cara memilih secara acak kromosom yang akan dimutasi.

4.1.2.4.1 Mutasi Normal

Setelah melakukan pemilihan individu yang akan dimutasi maka dilakukan proses mutasi. Kemudian dilakukan pemilihan secara acak nilai awal dan nilai akhir mutasi. Mutasi normal dilakukan dengan dua(2) cara yaitu mutasi *Inverse* dan mutasi *swap.*

Mutasi inverse dilakukan dengan membalik urutan *gen-gen* antara gen awal dan gen akhir. Berikut *source code* untuk proses mutasi *inverse*.

```
//child 3
kromosom[0]:=kromosom[individu];
for i:=start to stop-1 do
  for j:=start+1 to stop do
  begin
    tempi:=kromosom[0][i];
    tempj:=kromosom[0][j];
    if tempi<0 then
      tempi:=tempi*(-1);
    if tempj<0 then
      tempj:=tempj*(-1);
    if(tempi<tempj) then
    begin
      temp:=kromosom[0][i];
      kromosom[0][i]:=kromosom[0][j];
      kromosom[0][j]:=temp;
    end;
  end;
```

Gambar 4.9 Proses mutasi *inverse*

Kemudian dilakukan mutasi swap untuk mendapatkan anak ke empat(4). Mutasi swap dilakukan dengan menukar gen pada posisi awal dengan gen pada posisi akhir. Berikut *souce code* untuk melakukan mutasi swap.

```
//child 4
kromosom[0]:=kromosom[individu];
temp:=kromosom[0][start];
kromosom[0][start]:=kromosom[0][stop];
kromosom[0][stop]:=temp;
```

Gambar 4.10 Proses mutasi swap

Setelah selesai dilakukan proses mutasi swap akan didpat 4 (empat) individu anak dari proses perkam\winan silang dan mutasi normal.

4.1.2.4.2 Mutasi Rotasi

Mutasi rotasi didapatkan dengan melakukan rotasi segi empat yang akan dioptimasi dalam hal ini dengan mengkalikan nilai gen dengan minus satu(-1). Berikut *source code* untuk melakukan proses mutasi rotasi.

```
procedure algen.mutasiRotasi(individu:integer);
var i,j,ran:integer;
begin
randomize;
ran:=random(range)+1;
//child 5
kromosom[0]:=kromosom[individu];
kromosom[0][ran]:=kromosom[0][ran]*(-1);

kandidatBaru:=kandidatBaru+1;
kromosom[jml_kromosom+kandidatBaru]:=kromosom[0];
countFitnes(jml_kromosom+kandidatBaru);

end;
```

Gambar 4.11 Proses mutasi rotasi

Dengan demikian suruh proses perkawinan silang dan murasi dalam populasi selesai dengan menghasilkan 5 (lima) anak yang kemungkinan akan digunakan sebagai individu pada populasi selanjutnya

4.1.2.5 Algoritma *Bottom-Left*

Untuk melakukan pengepakan segi empat kedalam bidang digunakan algoritma *Bottom-left*. Algoritma *Bottom-left* dijalankan setiap dihasilkan kromosom baru. Segi empat akan dimasukkan satu per satu dari pojok kanan atas menuju ke pojok kiri bawah. Pertama-tama segiempat akan digeser sejauh mungkin ke arah bawah hingga berhimpit dengan dasar bidang dalam atau berhimpit dengan segiempat lain atau dalam arti lain tidak dapat digeser ke bawah lagi. Berikut *source code* untuk melakukan algoritma *Botoom-left*.

```
procedure room.moveStart(box:kotak);
var
l,p,ll,pp,Pointx,pointy,coastx,coasty,mentokDown,ment
okLeft,tempx,tempy,pan:integer;
begin
 //start from right top
Pointy:=box.lebar;
pan:=box.panjang;
Pointx:=Panjang-pan+1;

while(value[pointy][pointx+box.panjang-1]<>0) do
 pointx:=pointx-1;

mentokDown:=0;
mentokLeft:=0;

while(mentokDown=0) or (mentokLeft=0) do
begin
//Bergerak ke bawah
if (mentokDown=0) then
begin
  if (Pointy>=lebar) then
      mentokDown:=1
  else
  begin

   for l:=pointy to lebar do
   begin

     for p:=pointx to pointx+box.panjang-1 do
       if(value[l+1][p]<>0) then
       begin
         mentokDown:=1;
         break;
       end;
```

```
      if(mentokDown=0) then
        mentokLeft:=0
      else
       break;

    end;
    pointy:=l;
   end;
end;
```

Gambar 4.12 Proses Algoritma *Bottom-left* geser bawah

Setelah segiempat tidak dapat digeser ke bawah, dalam hal ini ditunjukkan pada

```
if (Pointy>=lebar) then
    mentokDown:=1
```

untuk segiempat yang berhimpit dengan dasar bidang atau

```
if(value[l+1][p]<>0) then
begin
mentokDown:=1;
```

untuk segiempat yang berhimpit dengan segiempat lain sehingga tidak dapat lagi untuk digeser ke bawah. Kemudian segiempat akan digeser kekiri sejauh mungkin hingga berhimpit dengan tepi bidang sebelah kiri atau berhimpit dengan segiempat lain. Berikut source code untuk pergeseran ke arah kiri.

```
//Bergerak ke ke kiri
if (mentokLeft=0) then
begin
  if (Pointx<=1) then
      mentokLeft:=1
  else
  begin
    for p:=pointx downto 1 do
    begin
     for l:=pointy downto pointy-box.lebar+1 do
        if(value[l][p-1]<>0) then
        begin
          mentokLeft:=1;
          break;
        end;
     if(mentokLeft=0) then
       mentokDown:=0
```

```
      else
        break;
     end;
     pointx:=p;
   end;
end;
end;
box.posx:=Pointx;
box.posy:=pointy;
fill(box);
end;
```

Gambar 4.13 Proses Algoritma *Bottom-left* geser kiri

Setelah segiempat tidak dapat digeser ke kiri kemudian akan diulangi menggeser sejauh mungkin ke bawah dan ke kiri lagi hingga akhirnya segi empat tidak dapat digeser ke bawah dan ke kiri lagi. Kemudian akan dilanjutkan dengan segiempat berikutnya hingga semua segiempat dapat ditata kedalam bidang.

4.1.2.6 Perhitungan Luas Sisa

Untuk mendapatkan hasil luas sisa perlu dideteksi ruang kosong yang tidak ikut dihitung dalam perhitungan luas sisa. Seperti yang dijelaskan pada bab 3, pencarian ruang kosong dilakukan dengan mendeteksi array bidang yang telah diisi segi empat menggunakan algoritma *Bottom-left*. Berikut *source code* yang digunakan untuk medeteksi ruang kosong yang ada.

```
procedure room.xscan();
var p,l,s,top,now,xstart,xstop:integer;
begin
for l:=findTmax+1 to lebar-1 do
begin
  top:=l;
  now:=top+1;
  p:=1;
  xstart:=0;
  xstop:=0;
  while(p<=panjang) do
  begin
     if (value[now][p]=0) then
     begin
```

```
       if (((value[top][p]=1)or(value[top][p]=9)) and
((value[now][p-1]=1)or(p=1))) then
       begin
         if ((value[now][p+1]=1)or(p=panjang)) then
            value[now][p]:=9
         else
            xstart:=p;
       end
       else if(value[top][p]=0) then
         xstart:=0
       else if(xstart<>0) then
       begin
          if (((value[top][p]=1)or(value[top][p]=9))
and ((value[now][p+1]=1)or(p=panjang))) then
             xstop:=p
       end;
       if((xstart<>0)and(xstop<>0)) then
         for s:=xstart to xstop do
            value[now][s]:=9;
     end;
     p:=p+1;
  end;
end;
end;
```

Gambar 4.14 Proses deteksi ruang kosong

Seperti ditunjukkan pada bab 3, deteksi ruang kosong dilakukan dengan mendeteksi bidang yang bernilai nol (0). Apabila bidang tersebut berhimpitan sebelah atas dan sebelah kiri dengan sisi tepi atau bidang bernilai satu (1) atau sembilan (9) maka bidang tersebut dianggap sebagai ruang kosong awal (xstart). Kemudian dicari bidang bernilai nol (0) disebelah kanannya yang berhimpitan sebelah atas dengan bidang yang bernilai satu (1) atau sembilan dan berhimpitan sebelah kanan dengan bidang bernilai satu (1) atau sisi tepi bidang, maka bidang tersebut dianggap sebagai ruang kosong akhir (xstop). Setelah itu nilai bidang dari xstart hingga x stop diubah menjadi sembilan (9).

Proses berikutnya adalah menghitung luas sisa seperti yang digambarkan pada *source code* berikut.

```
function room.countLuasSisa():integer;
var p,l,count:integer;
begin
xscan;
count:=0;
for l:=1 to Lebar do
  for p:=1 to Panjang do
    if(value[l][p]=0) then
      count:=count+1;

result:=count;
end;
```

Gambar 4.15 proses menghitung luas sisa

Dengan demikian sudah dapt diketahui luas sisa dari bidang. Untuk menghitung luas ruang kosong didapat dari hasil pengurangan luas bidang dengan luas sisa yang ditambahkan dengan jumlah luas seluruh segi empat.

4.1.2.7 Perhitungan Fitnes

Perhitungan fitnes berpengaruh besar terhadap penentuan pemilihan kromosom terbaik. Untuk optimasi pengepakan segiempat menggunakan algoritma *Bottom-left* semakin besar fitnes yang dihasilkan semakin besar pula kemungkinan kromosom untuk terpilih menjadi individu pada populasi berikutnya dan juga terpilih sebagai individu terbaik pada akhir generasi algoritma genetik.

Fitnes didapatkan dengan memanggil *procedure* penghitung fitness yaitu *countfitnes.* Fungsi *fitness* yaitu dihitung dari jumlah antara dua (2) kali tinggi maksimal segiempat pada bidang dengan luas sisa bidang yang telah dibagi dengan panjang bidang datar dan fungsi *fitness* yang hanya berdasarkan tinggi sisa. Seperti yang ditunjukkan oleh source berikut.

```
fitnes[indexKromosom]:= (2*tsisa)+(luassisa/roompanjang)
```

Gambar 4.16 Proses menghitung fitnes

4.1.2.8 Kromosom Terbaik

Pada pencarian kromosom terbaik yang akan digunakan sebagai individu pada populasi selanjutnya digunakan dua metode yaitu metode seleksi *ranking* dan roda *roulette.* Pertama-tama kromosom yang ada diduplikasi dulu sementara ke *temporary* kromosom. Kemudian dicari fitnes tertinggi yang akan dijadikan individu pertama pada populasi selanjutnya.

```
procedure algen.nextKandidat();
var
i,j,n,indexCand,randFitnes,countIndex,totalrow:integer;
    max,sumFitnes,jml:real;
    indexKandidat:array[1..20] of smallint;
    tempfitnes:array[0..20] of real;
    tempkromosom:array[0..20,1..100] of integer;
begin
totalrow:=jml_kromosom+kandidatBaru;

//duplicate kromosom and fitness
for i:=1 to totalrow do
begin
  for j:=1 to range do
     tempkromosom[i][j]:=kromosom[i][j];
  tempfitnes[i]:=fitnes[i];
end;

//find the biggest fitnest
max:=0;
for i:=1 to totalrow do
  if(tempfitnes[i]>max) then
  begin
     max:= tempfitnes[i];
     indexCand:=i;
  end;

//Get the best candidat
countIndex:=1;
fitnes[countIndex]:=max;
for n:=1 to range do
  kromosom[countIndex][n]:=tempkromosom[indexCand][n];
tempfitnes[indexCand]:=0;
```

Gambar 4.17 Seleksi ranking

Setelah didapat individu pertama, maka individu selanjutnya didapat dengan metode seleksi roda *roulette*. Proses seleksi roda *roulette* telah dijelaskan sebelumnya pada bab 2.

```
//Get the 2nd,3rd candidate and the other
Randomize;
for i:=1 to jml_kromosom do
begin
  sumFitnes:=0;
  for j:=1 to totalrow do
    sumFitnes:=sumFitnes+tempfitnes[j];

  randFitnes:=random(Floor(sumFitnes))+1;
  jml:=0;
  for j:=1 to totalrow do
  begin
     jml:=jml+tempfitnes[j];
     if(jml>=randFitnes) then
     begin
       countIndex:=countIndex+1;
       fitnes[countIndex]:=tempfitnes[j];
       for n:=1 to range do
         kromosom[countIndex][n]:=tempkromosom[j][n];
       tempfitnes[j]:=0;
       break;
     end;
  end;
end;

end;
```

Gambar 4.18 Seleksi roda roulette

Setelah semua individu yang dibituhkan untuk populasi selanjutnya didapatkan, maka dilakukan proses eliminsai kromosom. Yaitu menghapus kromosom yang selain kormosom awal untuk populasi berikutnya.

Caranya dengan menggunakan kromosom ke 20 (dua puluh) yang tidak pernah digunakan sehingga nilai kromosom-kromosom itu kembali menjadi kosong. Berikut source code untuk melakukan proses eliminasi kromosom.

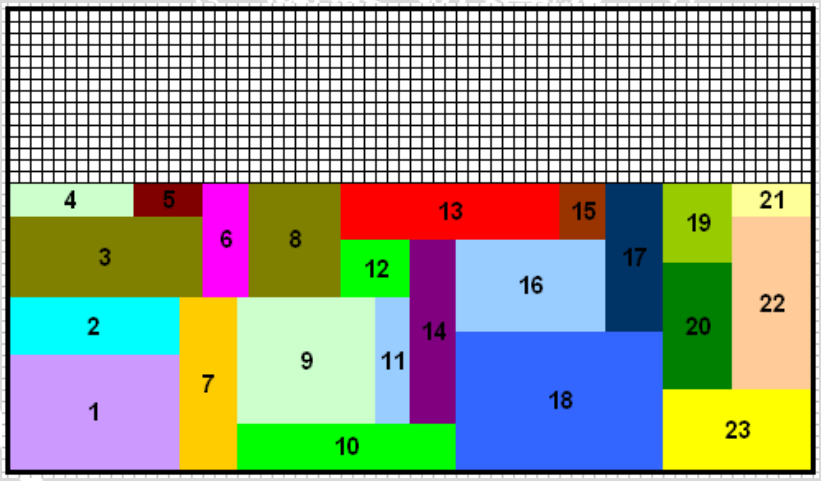
```
procedure algen.eliminateKandidat();
var i:integer;
begin
  kandidatBaru:=0;
  for i:=jml_kromosom+1 to 19 do
  begin
     kromosom[i]:=kromosom[20];
     fitnes[i]:=0;
  end;
end;
```

Gambar 4.19 Eliminasi kromosom

4.2 Penerapan Aplikasi

Aplikasi diterapkan dengan memasukkan data sesuai dengan keinginan user. Dalam kasus ini data yang dimasukkan adalah data segiempat yang akan dioptimasi dan juga data bidang datar sebagai wadah optimasi.

Untuk proses uji coba, terlebih dahulu dibuat secara manual gambar segi empat yang telah dioptimasi secara optimal dan dianggap sebagai hasil yang terbaik. Berikut gambar pembuatan secara manual:



Gambar 4.20 Gambar Pola manual

Berdasarkan gambar diatas dapat dijelaskan nilai horisontal (panjang) dari bidang datar yang digunakan adalah 70 sedangkan nilai vertikal (lebar) dari bidang datar yang digunakan adalah 50.

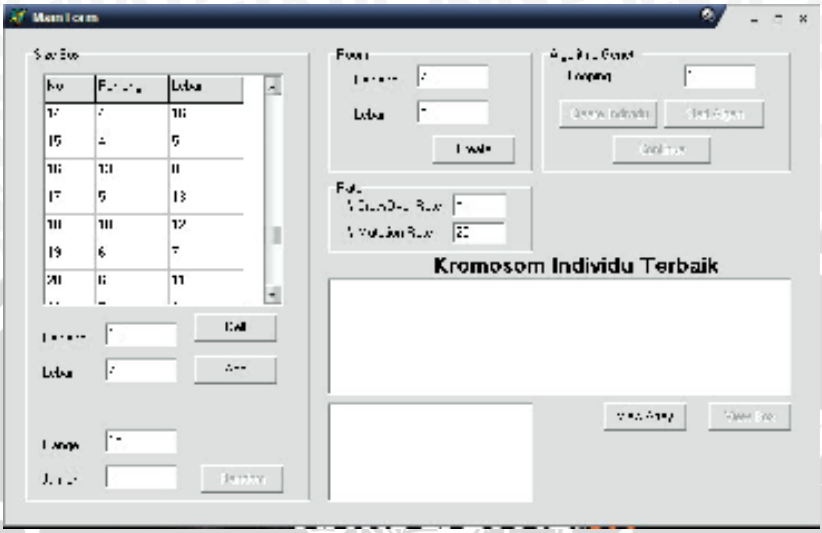
Selain itu juga dapat dilihat bahwa terdapat 23 segi empat yang ukurannya adalah sebagai berikut:

Tabel 4.4 Data segi empat yang dioptimasi

No.	panjang(H)	lebar(V)
1	15	10
2	15	5
3	17	7
4	11	3
5	6	3
6	4	10
7	5	15
8	8	10
9	12	11
10	19	4
11	3	11
12	6	5
13	19	5
14	4	16
15	4	5
16	13	8
17	5	13
18	18	12
19	6	7
20	6	11
21	7	3
22	7	15
23	13	7

Sehinga didapat nilai tinggi sisa adalah 25 dan luas sisa adalah 1750. Pada kasus ini fragmentasi eksternalnya sama dengan luas sisa sedangkan luas ruang kosongnya adalah 0.

Kemudian data diatas dimasukkan kedalam aplikasi agar dapat diproses. Berikut tampilan antar muka aplikasi setelah diberi input data:

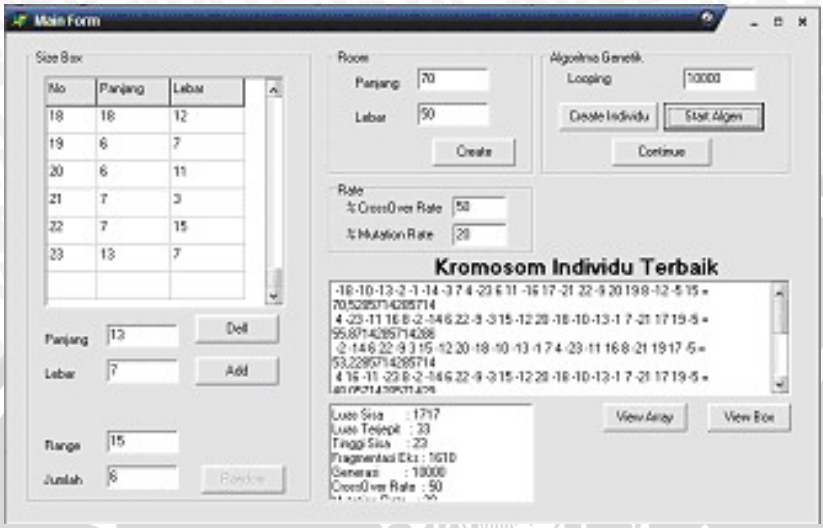


Gambar 4.21 *Form* input data terisi

Apabila nilai parameter sudah disimpan, maka proses selanjutnya optimasi pengepakan segi empat. Pada proses populasi awal, data yang dihasilkan akan tersimpan agar dapat digunakan pada percobaan berikutnya. Program dijalankan untuk melakukan sebanyak 20000 iterasi.

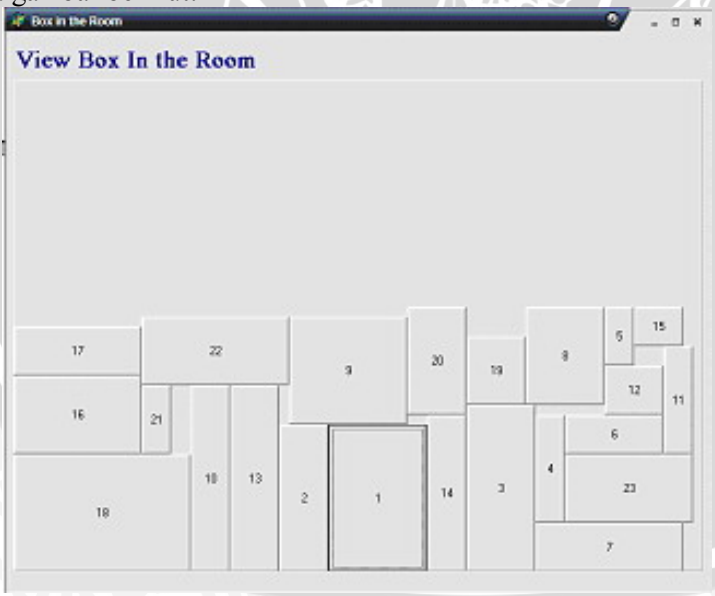
Setelah program dijalankan akan terlihat perubahan nilai-nilai gen pada kromosom yang berarti juga perubahan pada urutan segi empat yang diproses melalui lgoritma *bottom-left*. Selain itu juga akan terlihat perubahan pada nilai luas sisa dan nilai tinggi sisa, dimana nilai tinggi sisa berbanding terbalik dengan nilai tinggi peletakan. Semakin besar nilai luas sisa dan nilai tinggi sisa semakin baik hasil optimasi yang didapatkan.

Pada akhir proses genetik, akan ditampilkan nilai dari kromosom yang memiliki *fitness* terbaik. Nilai tersebut berupa nilai luas sisa, luas ruang kosong (luas terjepit), tinggi sisa, dan fragmentasi eksternal. Seperti yang digambarkan pada gambar berikut:



Gambar 4.22 *Form* hasil akhir

Berdasarkan gambar diatas dapat dilihat kromosom terbaik terletak pada baris pertama. Dari kromosom terbaik itu dibuat visualisasi penataan segi empat pada bidang datar yang dapat dilihat pada gambar berikut:



Gambar 4.23 Gambar segi empat hasil optimasi

Pada Gambar 4.23 dapat terlihat hasil optimasi segi empat pada bidang datar. Sehingga dapat di bandingkan dengan gambar yang dibuat secara manual.

4.3 Analisa Hasil

Perubahan nilai *fitness* dari inisialisasi sampai menjadi *fitness* terbaik dipengaruhi oleh parameter dasar yang penting yaitu probabilitas *crossover* dan mutasi. Probabilitas *crossover* menyatakan seberapa sering proses *crossover* akan terjadi diantara dua kromosom *parent*. Jika tidak terjadi *crossover*, keturunan merupakan salinan mutlak dari kromosom *parent*. Jika terjadi *crossover*, keturunan yang dihasilkan merupakan campuran dari kedua kromosom *parent*.

Nilai optimasi maksimal seperti pada Gambar 4.20 didapatkan apabila nilai luas sisa sama dengan nilai fragmentasi eksternal dan nilai luas ruang kosong (luas terjepit) sama dengan nol. Dimana nilai fragmentasi eksternal didapat dari nilai tinggi sisa dikalikan dengan panjang bidang.

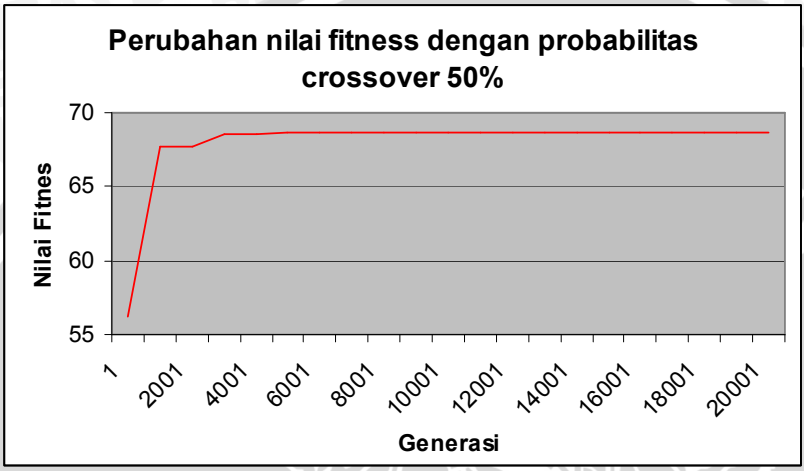
Pada analisa hasil akan dilakukan uji coba dengan mengganti nilai parameter probabilitas *crossover*. Uji coba dilakukan untuk mengetahui rata-rata nilai luas sisa, tinggi sisa dan nilai *fitness* yang terbaik. Nilai probabilitas *crossover* yang diujikan adalah 50, 60, 70, 80, dan 90. Uji coba dilakukan masing-masing 10 kali percobaan untuk setiap nilai probabilitas *crossover*. Hasil dari uji coba perubahan nilai probabilitas *crossover* dapat dilihat pada Tabel 4.5

Tabel 4.5 Hasil uji coba dengan probabilitas *crossover* yang berbeda-beda

No	Prob	Luas Sisa	Luas Ruang Kosong	Tinggi Sisa	Fragmentasi Eksternalks	Fitnes
1	50%	1707,1	42,9	22,1	1547	68,59
2	60%	1699,5	50,5	21,8	1526	67,88
3	70%	1709,4	40,6	21,9	1533	68,22
4	80%	1703,8	46,2	22	1540	68,34
5	90%	1689,7	60,3	22,1	1547	68,34

Pada tabel 4.5 dapat diketahui nilai rata-rata *fitness* dari setiap uji coba dengan nilai parameter *crossover* yang berbeda-beda.

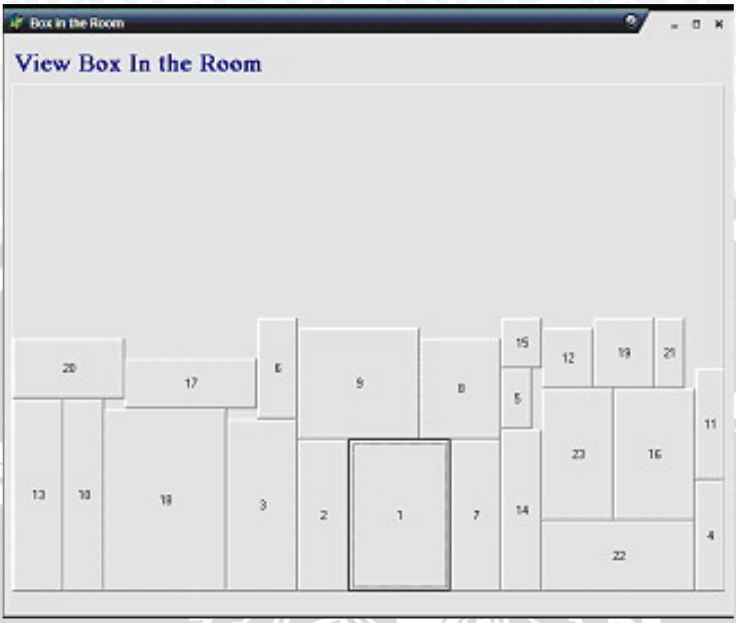
Pada uji coba dengan probabilitas *crossover* 50% didapatkan hasil yang paling optimal dengan nilai rata-rata *fitnees* tertinggi adalah 68,59



Gambar 4.24 Perubahan nilai fitness dengan probabilitas *crossover* 50%

Pada Gambar 4.23 dapat diketahui adanya kenaikan nilai *fitness* saat terjadi kenaikan generasi. Pada generasi tertentu nilai *fitnees* tidak mengalami perubahan. Berdasarkan uji coba yang dilakukan, dapat diambil kesimpulan bahwa nilai *fitness* akan semakin tinggi dengan semakin mendekati generasi ke 20000.

Setelah uji coba dilakukan, didapatkan hasil terbaik melalui perhitungan menggunakan probabilitas *crossover* 50% yang dapat dilihat pada gambar berikut :



Gambar 4.25 Hasil terbaik proses optimasi.

Hasil terbaik didapatkan pada percobaan ke 6 menggunakan pada probabilitas *crossover 50%*, dimana nilai tinggi sisanya adalah 23 sedangkan nilai luas sisanya adalah 1739 dimana nilai fragmentasi eksternalnya adalah 1610 dan nilai luas ruang kosongnya adalah 11. jika dibandingkan dengan nilai yang didapatkan dari gambar yang dibuat secara manual, nilai tersebut sudah cukup mendekati hasil maksimal.


UNIVERSITAS BRAWIJAYA

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari tugas akhir ini adalah :

1. Pada tugas akhir ini telah dibuat model genetika untuk masalah pengepakan segi empat pada bidang datar
2. Algoritma genetika yang digabungkan dengan algoritma *bottom-left* dapat digunakan sebagai alternatif solusi untuk menyelesaikan masalah pengepakan segi empat pada bidang datar.
3. Pada kasus pengepakan segi empat, nilai probabilitas *crossover* yang optimal digunakan adalah 50%. Pada probabilitas Crossover 50% didapatkan nilai *fitnees* tertinggi.

5.2 Saran

Aplikasi yang dibangun masih belum sempurna. Hal yang dapat bermanfaat untuk mengembangkan aplikasi ini adalah :

a. Aplikasi pengepakan segi empat pada bidang datar ini dapat dikembangkan menjadi sebuah sistem informasi yang memiliki kemampuan melakukan optimasi pengepakan segi empat pada bidang datar sesungguhnya yang ada pada industri plat baja atau industri tekstil.
b. Perubahan metode yang digunakan pada proses seleksi, crossover dan mutasi berpengaruh pada hasil akhir yang dicapai pada akhir generasi. Dapat dilakukan suatu penelitian pemilihan metode yang paling tepat sehingga dapat memaksimalkan hasil akhir dari proses pengepakan segi empat pada bidang datar dengan Algoritma Genetika.
c. Aplikasi pengepakan segi empat pada bidang datar ini dapat disempurnakan agar dapat memproses tidak hanya satu bidang datar tetapi juga beberapa bidang datar sekaligus.


UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

[1] Baker, B.S., Coffman, E.G., and Rivest, R.L., "Orthogonal Processing Letters 11/1 (1980) 37-39

[2] Callaghan, A.R., Nair, A.R., Lewis, K.E. *An Extension of the Orthogonal Packing Problem Through Dimensional Flexibility.* ASME Design Engineering Technical Conferences. Las Vegas, Nevada. 1999.

[3] Chen, Duanbing. Huang, Wenqi. *A Novel Quasi-human Heuristic Algorithm for Two-dimensional Rectangle Packing Problem.* IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.12, December 2006.

[4] Gen, Mitsuo and Runwei, Cheng. *Genetic Algorithms And Engineering Design*. John Wiley & Sons, Inc. New York: 1997.

[5] Kusumadewi, Sri. *Artificial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu. Yogyakarta: 2003

[6] Liu, D. & Teng, H. *An Improved BL-Algorithm for Genetic Algorithm of the Orthogonal Packing af Rectangles*. European Journal of Operational Research, Vol. 112, pp. 413-420: 1999.

[7] Liu, Yanbing. Chen, Duanbing. *A Novel Greedy Computing Algorithm for Rectangle Packing Problems.* IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.4, April 2006.

[8] Michalewicz , Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer. New York:1999

[9] Obitko, M. 1998. *Introduction to Genetic Algorithm*, http://cs.felk.cvut.cz/~xobitko/ga/. Tanggal akses: 28 Februari 2006

[10] Setiawan, Kuswara. *Paradigma Sistem Cerdas (jaringan saraf tiruan, logika fazi dan algoritma genetik)*. Bayumedia publishing. Malang: 2003.