3. PERANCANGAN DAN PEMBUATAN PROGRAM APLIKASI

3.1. Perancangan Database

Seperti telah dijelaskan pada bagian pendahuluan bahwa fasilitas *database* yang digunakan dalam tugas akhir ini adalah Microsoft Access 2000. Adapun *database*-nya bernama "opt.mdb" dengan spesifikasi tabel-tabelnya sebagai berikut :

- Tabel Ruang

  Tabel Ruang berfungsi untuk menyimpan data-data tentang ruang yang akan menjadi tempat proses optimasi. Struktrur dari tabel Ruang dapat dilihat pada tabel berikut ini :

Tabel 3.1. Struktur Tabel Ruang

Nama Field	Tipe Data	Ukuran Field	Keterangan
Kode_Ruang	Text	10	Simpan Kode Ruang
Panjang	Number	Long Integer	Simpan Panjang Ruang
Lebar	Number	Long Integer	Simpan Lebar Ruang
Tinggi	Number	Long Integer	Simpan Tinggi Ruang
Berat	Text	10	Simpan Beban Maks Ruang
Volume	Number	Long Integer	Simpan Volume Ruang
Keterangan	Text	35	Keterangan Ruang

- Tabel Barang

  Tabel Barang berfungsi untuk menyimpan data-data tentang barang yang nantinya akan dioptimalkan pola penyusunannya dalam ruangan tertentu. Struktur dari tabel Barang dapat dilihat pada tabel berikut ini :

Tabel 3.2. Struktur Tabel Barang

Nama Field	Tipe Data	Ukuran Field	Keterangan
Kode_Barang	Text	10	Simpan Kode Barang
Panjang	Number	Long Integer	Simpan Panjang Barang
Lebar	Number	Long Integer	Simpan Lebar Barang
Tinggi	Number	Long Integer	Simpan Tinggi Barang

Volume	Number	Long Integer	Simpan Volume Barang
Berat	Number	Long Integer	Simpan Berat Barang
Posisi	Text	5	Simpan Posisi barang
Berat_Tumpukan	Text	10	Simpan Beban Maks Barang
Rotasi	Text	15	Simpan Pilihan Rotasi

- Tabel Campuran

Tabel Campuran berfungsi untuk menggabungkan antara *field* Kode_Ruang sebagai *master* dan *field-field* dari tabel Barang sebagai *detail*. Artinya bahwa dalam satu ruang dapat diisi oleh bermacam macam ukuran barang dengan jumlah sesuai keinginan *user*. Struktur dari tabel Campuran dapat dilihat pada tabel berikut ini :

Tabel 3.3. Struktur Tabel Campuran

Nama Field	Tipe Data	Ukuran Field	Keterangan
Kode_Ruang	Text	10	Sebagai Master
Kode_Barang	Text	10	Sebagai Detail
Berat	Number	Long Integer	Lihat Berat Barang
Volume	Number	Long Integer	Lihat Volume Barang
Jumlah	Number	Long Integer	Lihat Jumlah Barang
Panjang	Number	Long Integer	Lihat Panjang Barang
Lebar	Number	Long Integer	Lihat Lebar Barang
Tinggi	Number	Long Integer	Lihat Tinggi Barang

- Tabel Solusi

Tabel Solusi berfungsi untuk menyimpan solusi-solusi tentang letak koordinat barang-barang didalam ruang yang akan dioptimasi oleh program aplikasi. Struktur dari tabel Solusi dapat dilihat pada tabel berikut ini :

Tabel 3.4. Struktur Tabel Solusi

Nama Field	Tipe Data	Ukuran Field	Keterangan
Kode_Ruang	Text	10	Sebagai Master
Generasi	Text	5	Simpan Generasi
Kromosom	Text	5	Simpan Kromosom
Kode_Barang	Text	10	Simpan Kode Barang
Posisi	Text	5	Simpan Posisi Barang
X	Text	5	Simpan Koord. X Barang
Y	Text	5	Simpan Koord. Y Barang
Z	Text	5	Simpan Koord. Z Barang

Operator	Text	30	Simpan Operator GA

- Tabel Sisa

Tabel Sisa berfungsi untuk menyimpan jumlah dari barang-barang yang tidak memenuhi syarat untuk dimasukkan ke dalam ruang yang akan dioptimalkan pengisiannya. Struktur dari tabel Sisa dapat dilihat pada tabel berikut ini :

Tabel 3.5. Struktur Tabel Sisa

Nama Field	Tipe Data	Ukuran Field	Keterangan
Kode_Ruang	Text	10	Sebagai Master
Generasi	Text	5	Simpan Generasi
Kromosom	Text	5	Simpan Kromosom
Kode_Barang	Text	10	Simpan Kode Barang
Sisa	Text	5	Simpan Sisa Barang

- Tabel Fitness

Tabel Fitness berfungsi untuk menyimpan seluruh data-data *fitness* yang dicari yaitu jumlah satuan ruang yang terisi oleh barang, jumlah barang yang posisinya seimbang (tidak roboh), dan jumlah barang yang tidak rusak (utuh) dari semua kromosom dalam tiap generasi yang ada setelah proses optimasi. Struktur dari tabel Fitness dapat dilihat pada tabel berikut ini :

Tabel 3.6. Struktur Tabel Fitness

Nama Field	Tipe Data	Ukuran Field	Keterangan
Kode_Ruang	Text	10	Sebagai Master
Berat_Max	Text	15	Lihat Beban Maks Ruang
Generasi	Number	Long Integer	Simpan Generasi
Kromosom	Number	Long Integer	Simpan Kromosom
Berat_Total	Number	Long Integer	Simpan Berat Total Barang
Fitness_Total	Number	Long Integer	Fitness Total Kromosom
¾_Ruang	Number	Long Integer	Jumlah Satuan Ruang Kosong
½_Ruang	Number	Long Integer	Jumlah Satuan Ruang Kosong
¼_Ruang	Number	Long Integer	Jumlah Satuan Ruang Kosong
Keseimbangan	Number	Long Integer	Jumlah Barang Seimbang
Beban	Number	Long Integer	Jumlah Barang Utuh

- Tabel Fitness_Final

  Tabel Fitness_Final berfungsi untuk menyimpan data-data *fitness* terbaik dari tiap generasi yang akan diperlihatkan kepada *user* dalam bentuk grafik *fitness*. Struktur dari tabel Fitness_Final dapat dilihat pada tabel berikut ini :

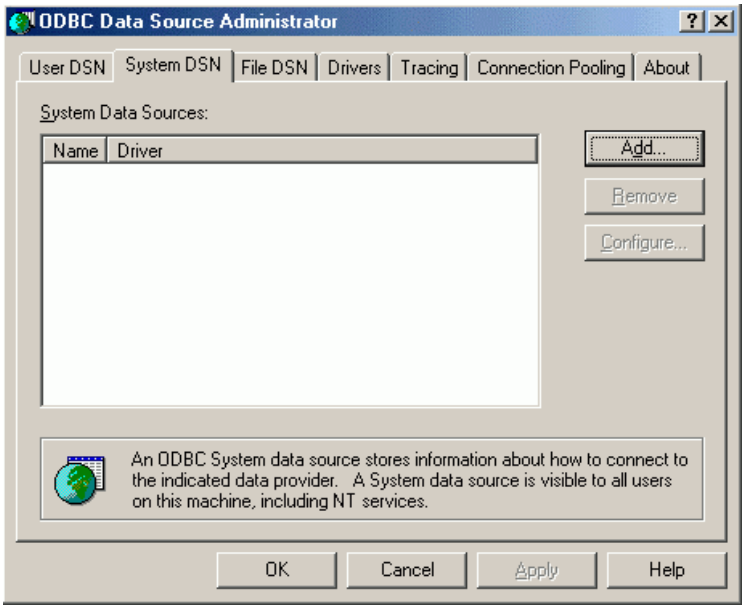
Tabel 3.7. Struktur Tabel Fitness_Final

Nama Field	Tipe Data	Ukuran Field	Keterangan
Kode_Ruang	Text	10	Sebagai Master
Generasi	Number	Long Integer	Lihat Generasi
Kromosom	Number	Long Integer	Lihat Kromosom
Fitness	Number	Long Integer	Lihat Fitness

3.2. Perancangan Koneksi Program Aplikasi Ke Database

Untuk menghubungkan program aplikasi dengan *database* yang ada digunakan fasilitas ODBC (*Open Database Connectivity*), yang cara inisialisasinya adalah sebagai berikut :

- Masuk ke *Control Panel*.
- Pilih ODBC *Data Sources* (32 bit).
- Pilih *System* DSN.
- Pilih Add



Gambar 3.1. Tampilan ODBC Data Source (32 bit)

- Pilih Microsoft Access Driver (*.mdb).
- Pilih *Finish*.
- *Data Source Name* diisi dengan “opt” yang menunjukkan nama dari *database* yang dipakai.
- Pilih *Select*.
- Pilih direktori tempat menyimpan *database* dengan nama “opt”
- Pilih OK

3.3. Perancangan DFD (*Data Flow Diagram*)

DFD dalam tugas akhir ini berfungsi untuk menampilkan alur data yang terjadi selama proses optimasi berlangsung. Adapun data-data yang perlu di *input*-kan oleh *user* sebelum proses optimasi adalah sebagai berikut:

- *Option* Ruang

  Berisi tentang definisi ruang tempat proses optimasi meliputi panjang, lebar, tinggi, dan beban maksimal yang dapat ditampung oleh ruang tersebut.

- *Option* Barang

  Berisi tentang definisi barang yang akan dioptimalkan pola penyusunannya meliputi panjang, lebar, tinggi, beban maksimal yang dapat ditampung, serta posisi rotasi yang dapat terjadi pada barang tersebut.

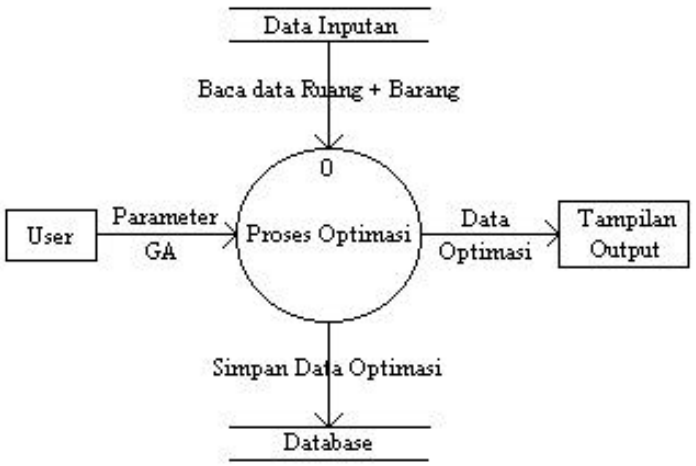
- *Costomize* Ruang

  Berisi tentang barang mana saja yang akan dimasukkan ke dalam ruang untuk dioptimalkan pola penyusunannya beserta dengan jumlahnya.

- *Optimize* Ruang

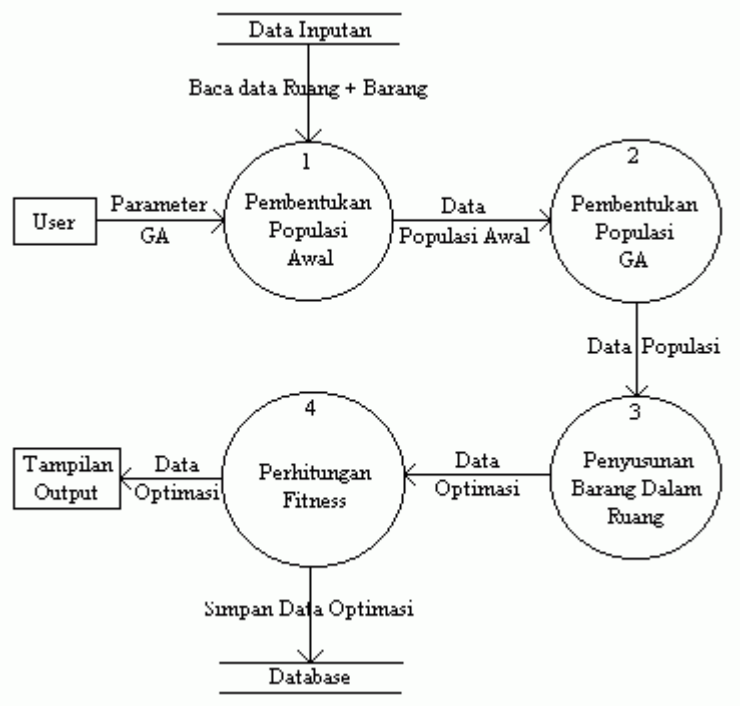
  Berisikan tentang parameter-parameter yang diperlukan dalam menjalankan metode Algoritma Genetik meliputi jumlah individu (m), probabilitas reproduksi (Pr), probabilitas pindah silang (Pc), probabilitas mutasi (Pm), dan jumlah generasi.

DFD Level 0



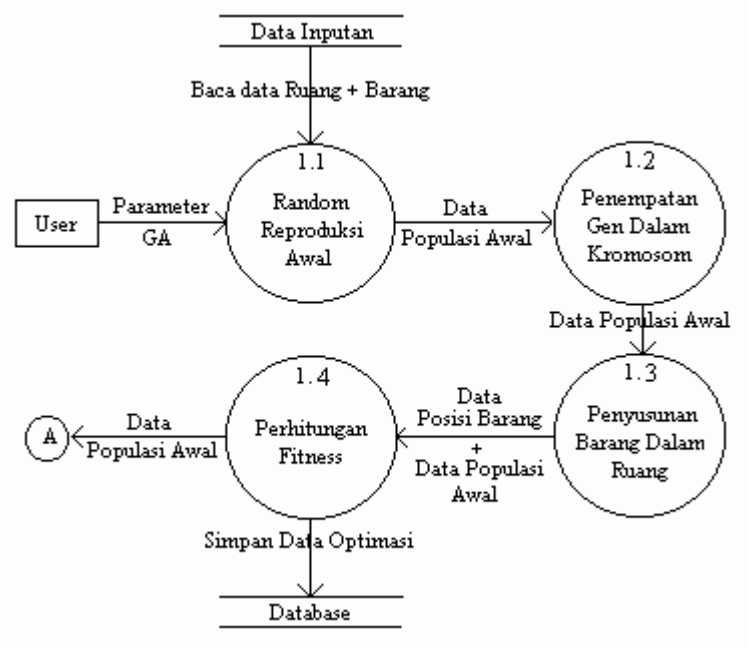
Gambar 3.2. DFD Level 0

DFD Level 1



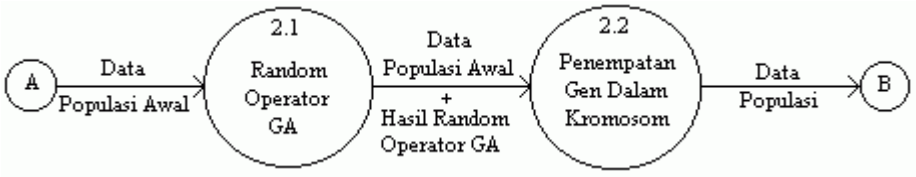
Gambar 3.3. DFD Level 1

DFD Level 2 (Pembentukan Populasi Awal)



Gambar 3.4. DFD Level 2.1

DFD Level 2 (Pembentukan Populasi GA)

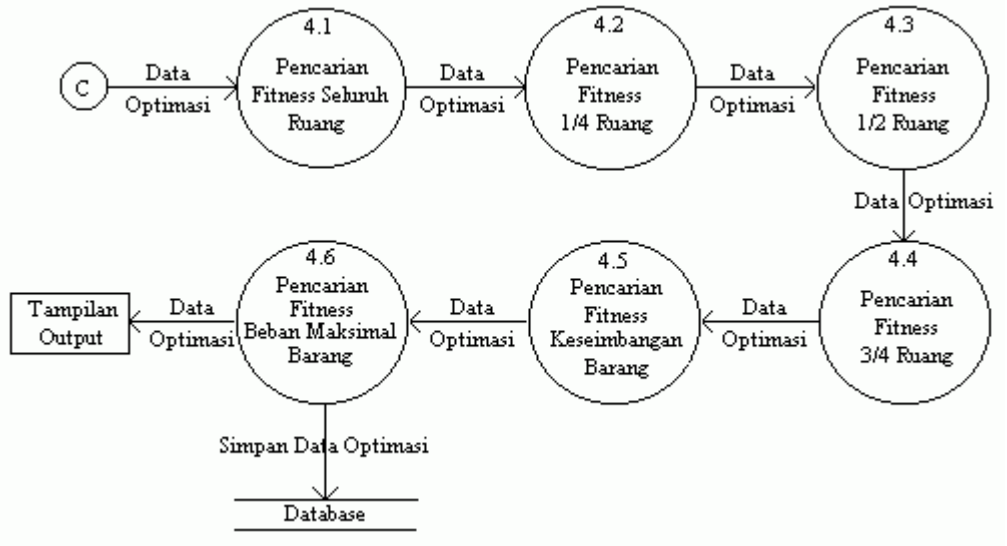


Gambar 3.5. DFD Level 2.2

DFD Level 2 (Penyusunan Barang Dalam Ruang)

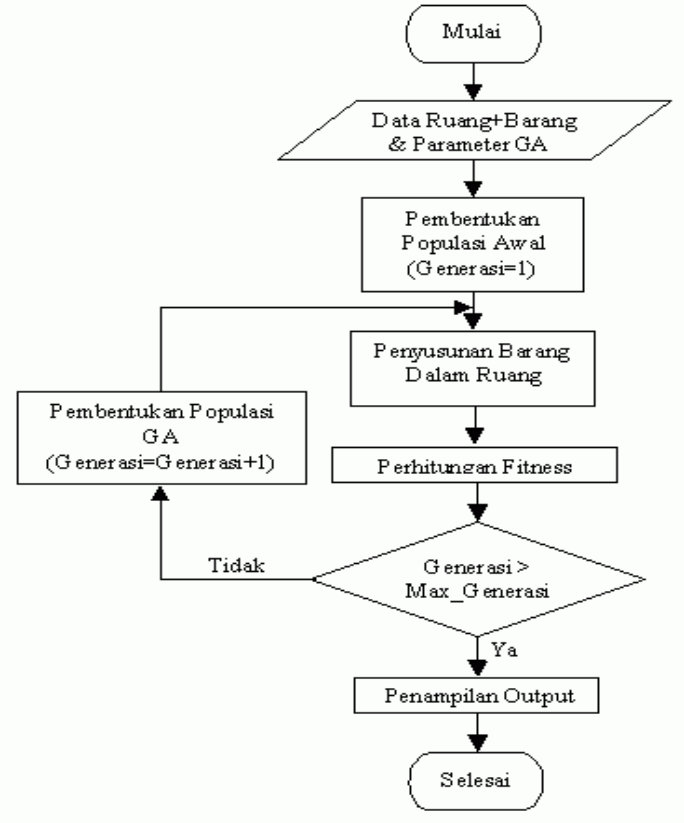
Gambar 3.6. DFD Level 2.3

DFD Level 2 (Perhitungan Fitness)



Gambar 3.7. DFD Level 2.4

Berikut ini adalah *flowchart* yang secara umum menjelaskan proses perulangan pada *Data Flow Diagram* diatas :



Gambar 3.8. *FlowChart* Umum Program Aplikasi

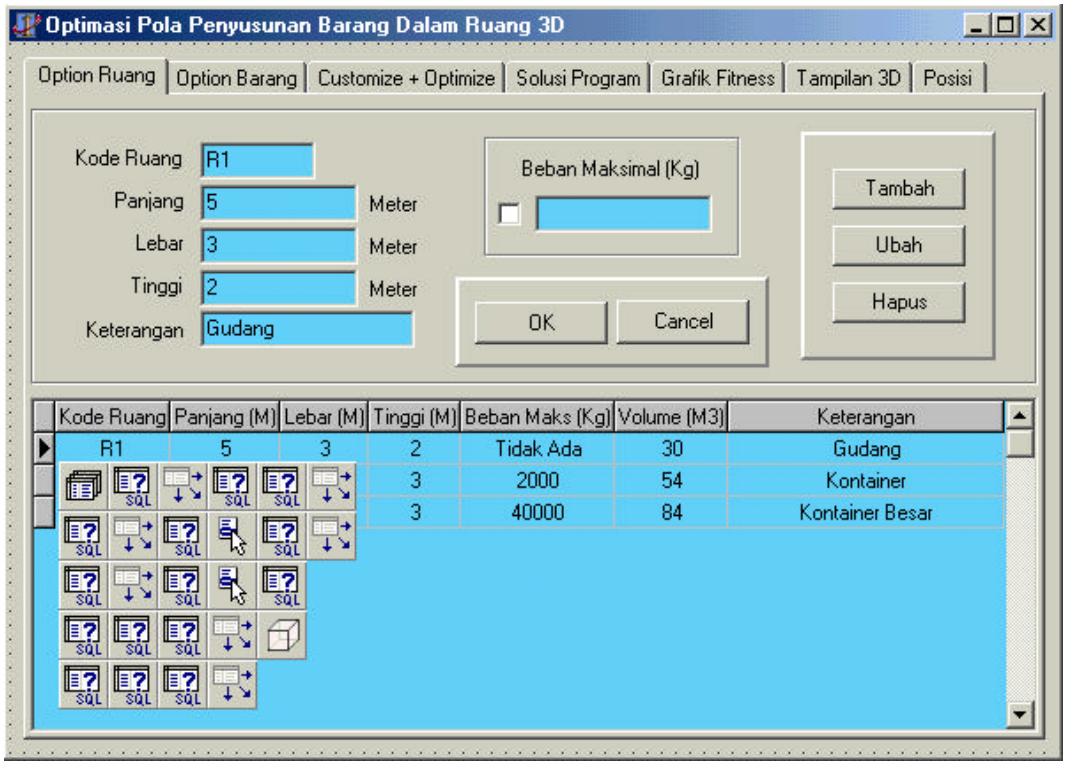
3.4. Pembuatan Program Aplikasi

Berikut ini adalah *source code* utama dan arah aliran data dari prosedur-prosedur penting yang terdapat dalam program aplikasi :

3.4.1. Data Masukan (*Input*)

- *Option* Ruang

  Dalam *Option* Ruang *user* bebas dalam menentukan definisi-definisi ruang yang dikehendaki untuk proses optimasi dengan fasilitas yang ada yaitu "Tambah", "Ubah", "Hapus". Tampilan dari *Option* Ruang dapat dilihat pada gambar berikut ini :



Gambar 3.9. Tampilan *Option* Ruang

- *Button* Tambah

```
procedure TMain.Button1Click(Sender: TObject);
begin
  status := 1;
  if not (QueryRuang.State = dsInsert) then
     QueryRuang.Append;
end;
```

- *Button* Ubah

```
procedure TMain.Button2Click(Sender: TObject);
begin
  status :=2;
  if not (QueryRuang.state = dsEdit) then
     QueryRuang.Edit;
end;
```

- *Button* Hapus

```
procedure TMain.Button3Click(Sender: TObject);
begin
  status:=3;
end;
```

- *Button* OK

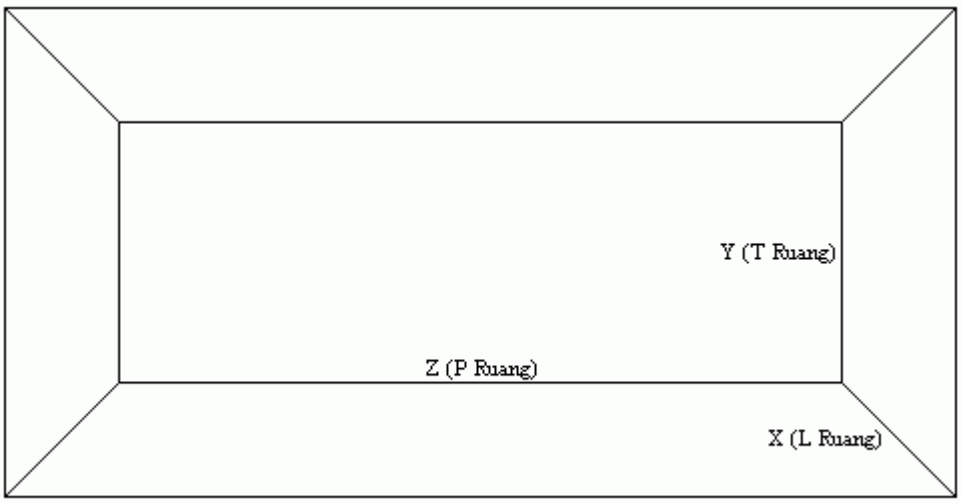
```
procedure TMain.Button4Click(Sender: TObject);
begin
  case status of
    1 :begin
                  +
                  +
         QueryRuang.Post;
         Queryruang.SQL.Clear;
         Queryruang.SQL.Add('Select * from Ruang      order
              by Kode_ruang;');
         QueryRuang.ExecSQL;
         QueryRuang.close;
         QueryRuang.open;
         QueryRuang.Append;
                  +
                  +
        end;
    2 :begin
                  +
                  +
         If not (QueryRuang.state = dsEdit) then
                 QueryRuang.Edit;
         QueryRuang.Post;
         Queryruang.SQL.Clear;
         Queryruang.SQL.Add('Select * from Ruang      order
              by Kode_ruang;');
         QueryRuang.ExecSQL;
         QueryRuang.close;
         QueryRuang.open;
                  +
                  +
       end;
    3 :begin
                  +
                  +
         QueryRuangHapus.SQL.Clear;
         QueryRuangHapus.SQL.Add('Delete  from  Ruang  where
              Kode_Ruang ='+chr(39)+DBEdit1.Text+chr(39));
         QueryRuangHapus.ExecSQL;
         QueryRuang.close;
         QueryRuang.open;
                  +
```

```
                    +
        end;
   end;
end;
```

- *Button* Cancel

```
procedure TMain.Button5Click(Sender: TObject);
begin
      +
      +
  status:=0;
  QueryRuang.Close;
  QueryRuang.sql.clear;
  QueryRuang.SQL.Add('select   *   from   Ruang   order   by
        Kode_Ruang');
  QueryRuang.open;
end;
```

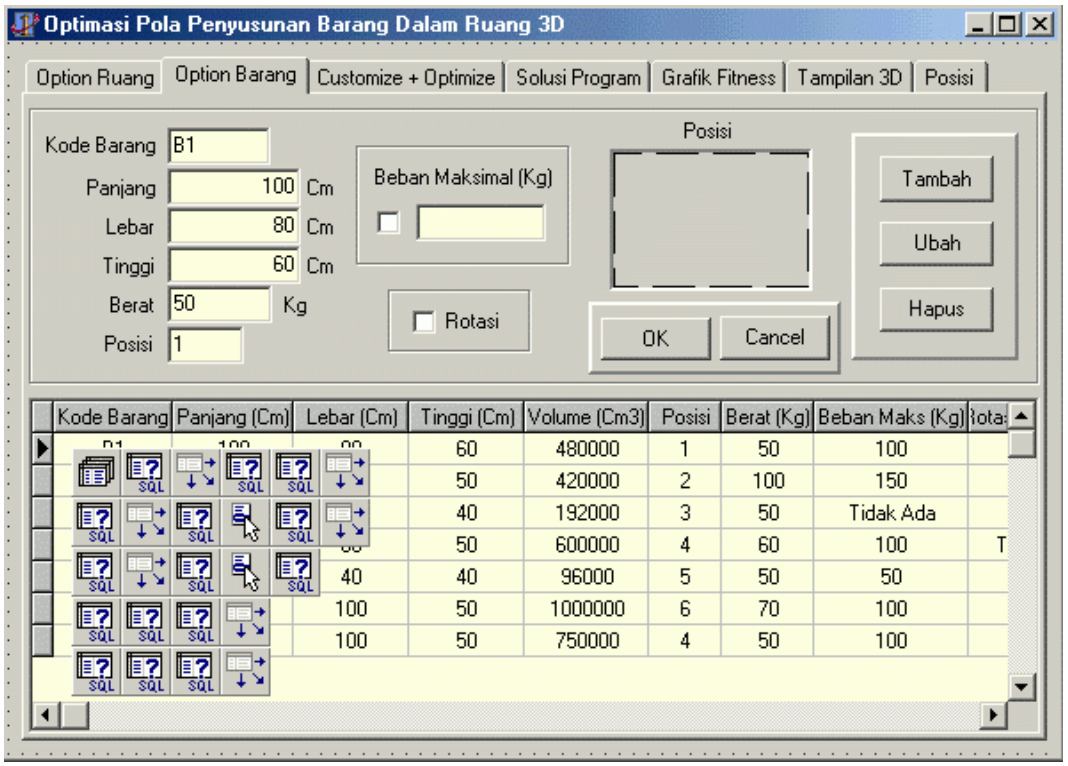
Gambar berikut ini adalah tampilan ruang dalam bentuk tiga dimensi dengan sumbu X mewakili “Lebar Ruang”, sumbu Y mewakili “Tinggi Ruang” dan sumbu Z mewakili “Panjang Ruang”.



Gambar 3.10. Tampilan Ruang Tiga Dimensi

- *Option* Barang

  Dalam *Option* Barang *user* bebas dalam menentukan definisi-definisi barang yang dikehendaki untuk proses optimasi dengan fasilitas yang ada yaitu “Tambah”, “Ubah”, “Hapus”. Tampilan dari *Option* Barang dapat dilihat pada gambar berikut ini :



Gambar 3.11. Tampilan *Option* Barang

- *Button* Tambah

```
procedure TMain.Button6Click(Sender: TObject);
begin
  status := 4;
  if not (QueryBarang.State = dsInsert) then
     QueryBarang.Append;
end;
```

- *Button* Ubah

```
procedure TMain.Button7Click(Sender: TObject);
begin
  status :=5;
  if not (QueryBarang.state = dsEdit) then
     QueryBarang.Edit;
end;
```

- *Button* Hapus

```
procedure TMain.Button8Click(Sender: TObject);
begin
  status:=6;
end;
```

- *Button* OK

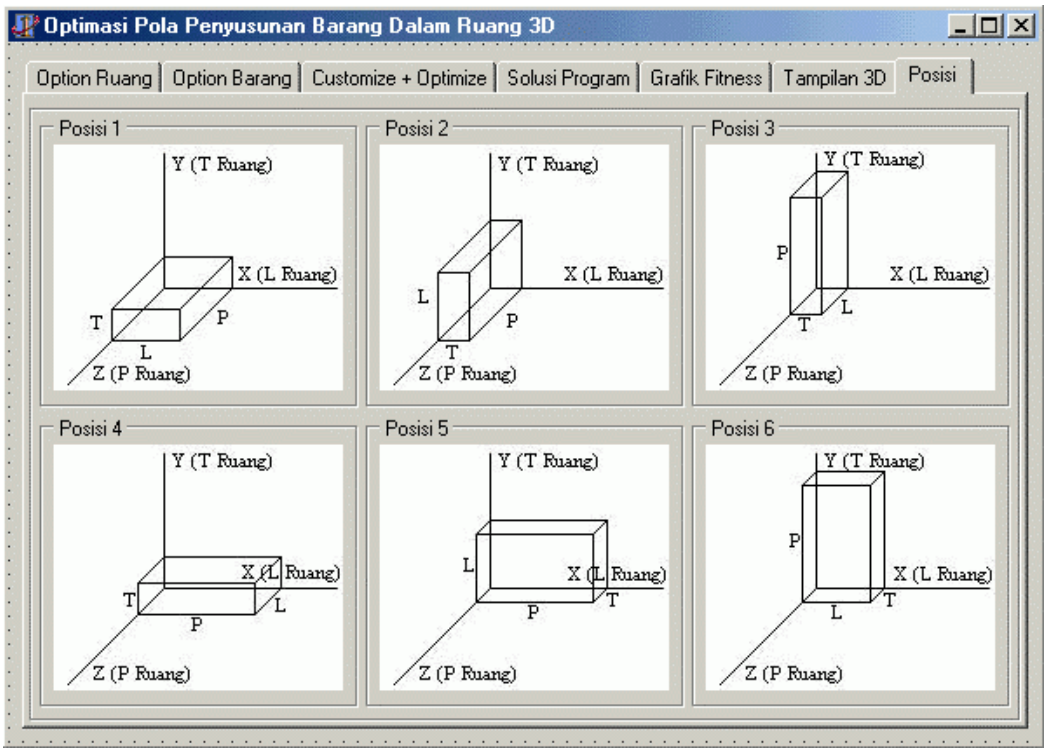
```
procedure TMain.Button9Click(Sender: TObject);
begin
  case status of
    4 :begin
                  +
                  +
          QueryBarang.Post;
          QueryBarang.SQL.Clear;
          QueryBarang.SQL.Add('Select    *    from    Barang
              order by Kode_Barang;');
          QueryBarang.ExecSQL;
          QueryBarang.close;
          QueryBarang.open;
          QueryBarang.Append;
                  +
                  +
        end;
    5 :begin
                  +
                  +
          If not (QueryBarang.state = dsEdit) then
                   QueryBarang.Edit;
          QueryBarang.Post;
          QueryBarang.SQL.Clear;
          QueryBarang.SQL.Add('Select    *    from    Barang
              order by Kode_Barang;');
          QueryBarang.ExecSQL;
          QueryBarang.close;
          QueryBarang.open;
                  +
                  +
       end;
    3 :begin
                  +
                  +
          QueryBarangHapus.SQL.Clear;
          QueryBarangHapus.SQL.Add('Delete    from    Barang
              where                          Kode_Barang
              ='+chr(39)+DBEdit6.Text+chr(39));
          QueryBarangHapus.ExecSQL;
          QueryBarnag.close;
          QueryBarang.open;
                  +
                  +
       end;
  end;
end;
```

- *Button* Cancel

```
procedure TMain.Button10Click(Sender: TObject);
begin
      +
      +
  status:=0;
  QueryBarang.Close;
```

```
    QueryBarang.sql.clear;
    QueryBarang.SQL.Add('select   *   from   Barang   order   by
     Kode_barang');
    Querybarang.open;
  end;
```

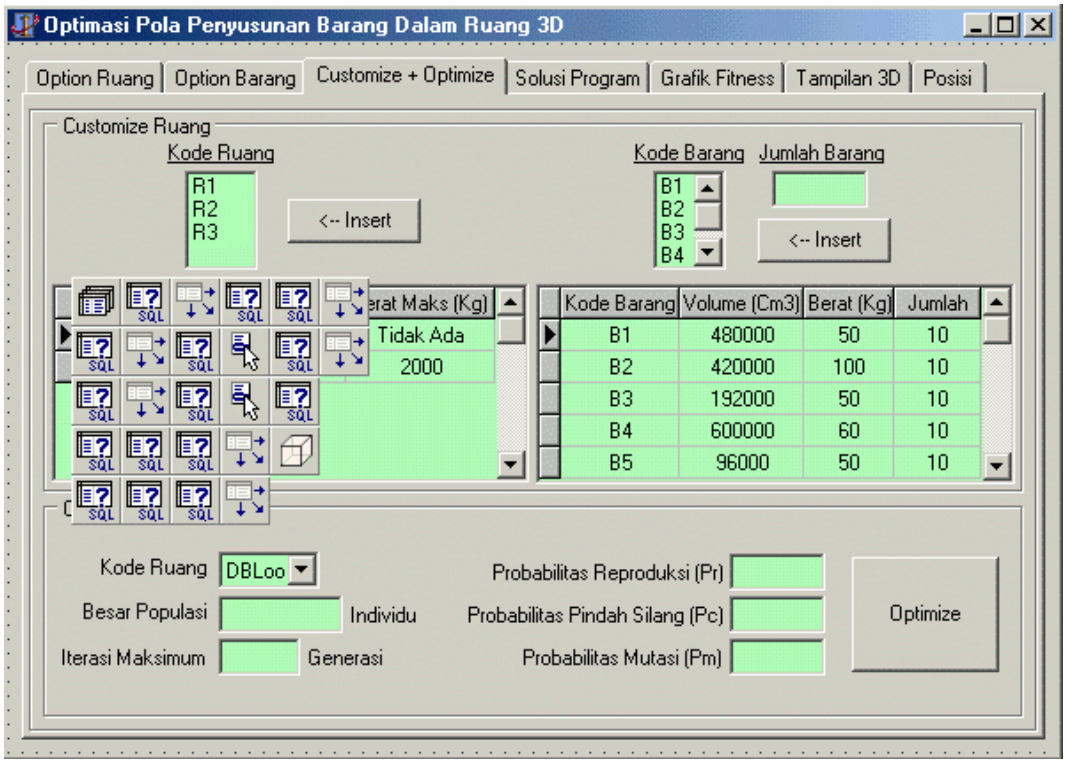
Barang dengan bentuk kotak atau balok dalam penyusunannya dapat memiliki enam buah posisi berbeda seperti pada gambar berikut :



Gambar 3.12. Tampilan Posisi Barang

- *Customize* dan *Optimize*

User bebas memilih ruang mana yang akan menjadi tempat optimasi dan mengisinya dengan barang-barang beserta jumlahnya. Selain itu, *user* harus menentukan salah satu ruang yang akan dioptimalkan pengisiannya dan mengisi parameter-parameter Algoritma Genetik yang meliputi “Besar Populasi”, “Iterasi Maksimum”, “Probabilitas Reproduksi”, “Probabilitas Pindah Silang”, dan “Probabilitas Mutasi”. Tampilan dari *Customize + Optimize* dapat dilihat pada gambar berikut ini :



Gambar 3.13. Tampilan *Customize* dan *Optimize*

3.4.2. Proses Optimasi

Pada saat *button* Optimize ditekan, maka program akan melakukan proses optimasi dengan langkah-langkah sebagai berikut :

3.4.2.1. Pembentukan Populasi Awal

Seluruh barang yang akan dioptimalkan pola penyusunannya dalam ruang yang dipilih dimasukkan kedalam variabel gen (tipe : *array of string[4]*) dimana masing-masing gen diisi oleh kode barang dan diikuti oleh gen berikutnya yang diisi posisi awal barang tersebut. Jadi panjang variabel gen sama dengan dua kali jumlah barang yang akan dioptimalkan.

```
{program pembentukan populasi awal}
procedure Populasi_Awal;
var i:integer;
begin
  main.QueryOptimasi.SQL.Clear;
  main.QueryOptimasi.SQL.Add('select   *   from   Campuran   where
     Kode_Ruang=:kr;');
  main.QueryOptimasi.Params[0].AsString:=main.dblookupcombobox1.Te
     xt;
```

```
  main.QueryOptimasi.ExecSQL;
  main.QueryOptimasi.Close;
  main.QueryOptimasi.Open;

  jumlah_gen:=0;
  loop:=0;
  temp:=0;
  main.queryOptimasi.First;
  while not main.queryOptimasi.Eof do
   begin
       jumlah_gen:=jumlah_gen+strtoint(main.queryoptimasi['Jumlah']
          );
      setlength(gen,jumlah_gen*2+4);
    for i:=1 to strtoint(main.queryoptimasi['Jumlah'])do
      begin
        Main.QueryBarangKembar.SQL.Clear;
        Main.QueryBarangKembar.SQL.Add('select * from Barang where
            Kode_Barang=:kb;');
        Main.QueryBarangKembar.Params[0].AsString:=main.
            QueryOptimasi['Kode_Barang'];
        Main.QueryBarangKembar.ExecSQL;
        Main.QueryBarangKembar.Close;
        Main.QueryBarangKembar.Open;
        gen[loop+temp+i]:=main.queryOptimasi['Kode_barang'];
        gen[loop+temp+i+1]:=main.queryBarangKembar['Posisi'];
        temp:=i;
       end;
     temp:=0;
     loop:=loop+((strtoint(main.queryoptimasi['Jumlah']))*2);
     main.queryOptimasi.Next;
   end;
Atur_Posisi {prosedur}
end;
```

3.4.2.2. Pengaturan Posisi Gen

Pada awal prosedur Atur_Posisi dilakukan pencarian FPB (Faktor Persekutuan Terbesar) dari seluruh ukuran barang yang ada beserta ukuran ruang tempat berlangsungnya proses optimasi. Pencarian FPB ini berfungsi untuk membuat skala perbandingan dari ukuran yang sebenarnya dengan tujuan mempercepat proses optimasi.

Misalnya : - Barang B1 : Panjang 150 Cm, Lebar 100 Cm, Tinggi 50 Cm
- Barang B2 : Panjang 80 Cm, Lebar 40 Cm, Tinggi 40 Cm
- Ruang R1 : Panjang 700 Cm, Lebar 400 Cm, Tinggi 300 Cm

Maka FPB akhir yang didapat adalah 10. Nilai 10 didapat dari membagi seluruh ukuran barang dan ruang yang ada dengan angka mulai dari 1 hingga ukuran barang terkecil. Jika pembagian tidak bersisa, maka nilai pembagi tersebut menjadi FPB.

Dengan FPB bernilai 10 maka perbandingan menjadi 1 Cm : 10 Cm, sehingga setiap ukuran dapat diperkecil dengan cara membagi dengan nilai FPB. Hasil yang didapat adalah :

- Barang B1 : Panjang 15 Cm, Lebar 10 Cm, Tinggi 5 Cm
- Barang B2 : Panjang 8 Cm, Lebar 4 Cm, Tinggi 4 Cm
- Ruang R1 : Panjang 70 Cm, Lebar 40 Cm, Tinggi 30 Cm

```
{program mencari nilai FPB}
{menghitung total panjang, lebar, tinggi barang ditambah panjang,
lebar, tinggi ruang}
total:=0;
main.QueryOptimasi.First;
while not main.QueryOptimasi.Eof do
  begin
    inc(total);
    ukuran_barang[total]:=main.QueryOptimasi['Panjang'];
    inc(total);
    ukuran_barang[total]:=main.QueryOptimasi['Lebar'];
    inc(total);
    ukuran_barang[total]:=main.QueryOptimasi['Tinggi'];
    main.QueryOptimasi.Next;
  end;

main.queryoptimasi.SQL.clear;
main.queryoptimasi.sql.add('Select    *    from    Ruang    where
   Kode_Ruang=:kr');
main.queryoptimasi.Params[0].asstring:=main.DBLookupComboBox1.Text
   ;
main.queryoptimasi.ExecSQL;
main.queryoptimasi.close;
main.queryoptimasi.open;
main.queryoptimasi.First;

inc(total);
ukuran_barang[total]:=strtoint(floattostr(strtofloat(main.QueryOpt
   imasi['Panjang'])*100));
inc(total);
ukuran_barang[total]:=strtoint(floattostr(strtofloat(main.QueryOpt
   imasi['Lebar'])*100));
inc(total);
ukuran_barang[total]:=strtoint(floattostr(strtofloat(main.QueryOpt
   imasi['Tinggi'])*100));

{pencarian FPB}
FPB:=1;
for i:=1 to ukuran do
begin
  for j:=1 to total do
    begin
       if (ukuran_barang[j] mod i=0) then
          begin
             statusFPB:=true;
```

```
         end
      else
         begin
            statusFPB:=false;
            break;
         end;
   end;
  if statusFPB then FPB:=i;
end;
```

3.4.2.3. Penempatan Gen Dalam Kromosom

Gen-gen yang telah di bentuk pada awal proses optimasi dimasukkan ke dalam variabel kromosom (tipe : *array of array of string[4]*) sebanyak individu dalam populasi yang dibentuk oleh *user*. Proses penempatan gen ini berlangsung terus menerus setiap muncul generasi baru sampai iterasi maksimum yang diinputkan oleh *user.*

Pada generasi pertama proses penempatan gen dilakukan dengan cara *randomize* untuk mengisi semua kromosom yang ada. Proses ini disebut dengan "Reproduksi Awal".

Mulai generasi kedua sampai iterasi maksimum teknik penempatan gen dilakukan berdasarkan hasil *random* angka antara 0 sampai 1 untuk mengetahui operator Agoritma Genetik apa yang akan dipakai sebagai landasan penempatan gen. Jadi jumlah probabilitas ketiga operator Algoritma Genetik yaitu Reproduksi (*Pr*), Pindah Silang (*Pc*), dan Mutasi (*Pm*) harus sama dengan satu.

Dengan menggunakan teknik *elitism*, yaitu mempertahankan individu yang memiliki *fitness* terbaik pada generasi tertentu untuk diikutsertakan kembali dalam proses algoritma genetik pada generasi berikutnya, diharapkan mampu untuk mendapatkan individu baru (*offspring*) dengan *fitness* yang lebih baik lagi melalui operator pindah silang.

- Progam Implementasi Operator Reproduksi (*Pr*)

```
for i:=1 to individu do
  begin
    jumlah_gen2:=jumlah_gen*2;
    for l:=1 to jumlah_gen2 do
        gen1[l]:=gen[l];
    temp:=0;

    {mengisi array kromosom dengan me-random seluruh gen yang
    ada}
    for j:=1 to jumlah_gen do
```

```
      begin
        repeat
          randomize;
          random_gen:=random(jumlah_gen2);
        until random_gen<>0;
         if (random_gen mod 2=0) then random_gen:=random_gen-1;
         kromosom[i,temp+j]:=gen1[random_gen];
         kromosom[i,temp+j+1]:=gen1[random_gen+1];
         temp:=j;

         {mengurangi 1 gen & posisinya dari array}
         if random_gen<jumlah_gen2-1 then
            begin
               for k:=random_gen to jumlah_gen2 do
                  begin
                 gen1[k]:=gen1[k+2];
               end;
           end;
           jumlah_gen2:=jumlah_gen2-2;
      end;
  end;
```

- Program Implementasi Operator Pindah Silang (*Pc*)

```
{menentukan batas kiri}
repeat
   randomize;
   random_gen:=random(jumlah_gen);
until (random_gen<>0) and (random_gen<jumlah_gen);
sisa_random:=jumlah_gen-random_gen;

{menentukan batas kanan}
repeat
   randomize;
   random_gen1:=random(sisa_random);
until (random_gen1<>0);
batas_kiri_cross:=inttostr(random_gen);
batas_kanan_cross:=inttostr(random_gen1);
random_gen:=random_gen*2-1;{batas kiri}
random_gen1:=random_gen1*2+2;{batas kanan}

for i:=1 to individu do
 begin
  jumlah_gen2:=jumlah_gen*2;

   {cari sisa gen2}
  if random_gen<>1 then
    begin
     for j:=1 to random_gen-1 do
       begin
        kromosom_sisa[i,j]:=kromosom[i,j];
       end;
     lanjut:=random_gen;
     if random_gen+random_gen1-1<jumlah_gen2-1 then
     begin
     for j:=random_gen+random_gen1 to jumlah_gen2 do
       begin
         kromosom_sisa[i,lanjut]:=kromosom[i,j];
         inc(lanjut);
```

```
        end;
      end;

      {seluruh sisa gen disimpan dlm kromosom_sisa}
      {isi kromosom[i,j]}
      loop:=1;
     for j:=random_gen to random_gen+random_gen1-1 do
       begin
        kromosom1[i,loop]:=kromosom[i,j];
        inc(loop);
       end;
     lanjut:=1;
     for j:=loop to jumlah_gen2 do
       begin
         kromosom1[i,j]:=kromosom_sisa[i,lanjut];
         inc(lanjut);
       end;
     for j:=1 to jumlah_gen2 do
       begin
          kromosom[i,j]:=kromosom1[i,j];
       end;
      end
     else
      begin
       lanjut:=random_gen;
       if random_gen+random_gen1-1<jumlah_gen2-1 then
         begin
             for j:=random_gen+random_gen1 to jumlah_gen2 do
             begin
                kromosom_sisa[i,lanjut]:=kromosom[i,j];
                inc(lanjut);
             end;
         end;

      {pembentukan kromosom baru}
      loop:=1;
       for j:=random_gen to random_gen+random_gen1-1 do
        begin
          kromosom1[i,loop]:=kromosom[i,j];
          inc(loop);
        end;
       lanjut:=1;
       for j:=loop to jumlah_gen2 do
         begin
           kromosom1[i,j]:=kromosom_sisa[i,lanjut];
           inc(lanjut);
         end;
       for j:=1 to jumlah_gen2 do
         begin
             kromosom[i,j]:=kromosom1[i,j];
         end;
      end;
   end;
```

- Program Implementasi Operator Mutasi (*Pm*)

```
{cari gen pertama}
jumlah_gen2:=jumlah_gen*2;
repeat
```

```
   randomize;
   random_gen:=random(jumlah_gen2);
until (random_gen<>0);
if random_gen mod 2=0 then
  begin
     permutasi1:=inttostr(random_gen div 2);
     random_gen:=random_gen-1;
  end
else permutasi1:=inttostr((random_gen+1)div 2);
{cari gen kedua}
repeat
   randomize;
   random_gen1:=random(jumlah_gen2);
until   (random_gen1<>0)   and   (random_gen1<>random_gen)   and
   (random_gen1<>random_gen+1);
if random_gen1 mod 2=0 then
  begin
    permutasi2:=inttostr(random_gen1 div 2);
    random_gen1:=random_gen1-1;
  end
else permutasi2:=inttostr((random_gen1+1) div 2);

{proses mutasi gen}
for i:=1 to individu do
  begin
     lanjut1:=kromosom[i,random_gen];
     lanjut2:=kromosom[i,random_gen+1];
     kromosom[i,random_gen]:=kromosom[i,random_gen1];
     kromosom[i,random_gen+1]:=kromosom[i,random_gen1+1];
     kromosom[i,random_gen1]:=lanjut1;
     kromosom[i,random_gen1+1]:=lanjut2;

     main.QueryBarangKembar.SQL.Clear;
     main.QueryBarangKembar.SQL.Add('select * from Barang where
        Kode_Barang=:kb;');
     main.QueryBarangKembar.Params[0].AsString:=kromosom[i,rand
        om_gen];
     main.QueryBarangKembar.ExecSQL;
     main.QueryBarangKembar.Close;
     main.QueryBarangKembar.Open;

     {penentuan rotasi posisi barang}
     if main.QueryBarangKembar['Rotasi']='Tanpa Rotasi' then
       begin
          if          kromosom[i,random_gen+1]='1'          then
             kromosom[i,random_gen+1]:='4';
          if          kromosom[i,random_gen+1]='2'          then
             kromosom[i,random_gen+1]:='5';
          if          kromosom[i,random_gen+1]='3'          then
             kromosom[i,random_gen+1]:='6';
          if          kromosom[i,random_gen+1]='4'          then
             kromosom[i,random_gen+1]:='1';
          if          kromosom[i,random_gen+1]='5'          then
             kromosom[i,random_gen+1]:='2';
          if          kromosom[i,random_gen+1]='6'          then
             kromosom[i,random_gen+1]:='3';
       end
     else
       begin
```

```
          repeat
             randomize;
             temp_posisi:=random(7);
          until temp_posisi<>0;
          kromosom[i,random_gen+1]:=inttostr(temp_posisi);
        end;
          main.QueryBarangKembar.SQL.Clear;
          main.QueryBarangKembar.SQL.Add('select  *  from  Barang
              where Kode_Barang=:kb;');
          main.QueryBarangKembar.Params[0].AsString:=kromosom[i,r
              andom_gen1];
          main.QueryBarangKembar.ExecSQL;
          main.QueryBarangKembar.Close;
          main.QueryBarangKembar.Open;
          if main.QueryBarangKembar['Rotasi']='Tanpa Rotasi' then
           begin
             if         kromosom[i,random_gen1+1]='1'         then
                kromosom[i,random_gen1+1]:='4';
             if         kromosom[i,random_gen1+1]='2'         then
                kromosom[i,random_gen1+1]:='5';
             if         kromosom[i,random_gen1+1]='3'         then
                kromosom[i,random_gen1+1]:='6';
             if         kromosom[i,random_gen1+1]='4'         then
                kromosom[i,random_gen1+1]:='1';
             if         kromosom[i,random_gen1+1]='5'         then
                kromosom[i,random_gen1+1]:='2';
             if         kromosom[i,random_gen1+1]='6'         then
                kromosom[i,random_gen1+1]:='3';
           end
         else
           begin
             repeat
               randomize;
               temp_posisi:=random(7);
             until temp_posisi<>0;
              kromosom[i,random_gen1+1]:=inttostr(temp_posisi);
           end;
       end;
    end;
```

3.4.2.4. Penyusunan Barang Dalam Ruang

Proses penyusunan barang dalam ruang dilakukan secara berulang-ulang sebanyak individu yang ada dalam setiap generasi. Teknik penyusunan barang yang dipakai adalah sebagai berikut :

- Barang disusun dalam ruang tiga dimensi yang koordinatnya diwakili oleh variabel posisi (tipe : *array of array of array of string [10]*).
- Sebelum proses penyusunan, ruang dikosongkan dengan membebaskan memori dari variabel posisi (*Setlength*(posisi,1,1,1)).
- Dilakukan Pemesanan memori untuk variabel posisi yang mewakili panjang, lebar, dan tinggi ruang (*Setlength*(posisi,l_ruang,t_ruang,p_ruang)).

- Mulai dari barang (gen) pertama sampai gen terakhir dalam variabel kromosom dilakukan langkah-langkah sebagai berikut :
  - Pengecekkan Beban Maksimal Ruang

    Jika dalam inputan awal ruang memiliki beban maksimal maka variabel status_ruang_max (tipe : *Boolean*) diisi *true* yang berarti dalam penempatan barang beban maksimal ruang diperhatikan , jika tidak memiliki beban maksimal maka variabel status_ruang_max diisi *false* yang berarti dalam penempatan barang beban maksimal ruang diabaikan.

    {pengecekkan beban maksimal ruang}

```
if status_ruang_max then
  begin
    total_berat:=total_berat+berat_brg;
    if total_berat<=ruang_max then
      begin
        total_berat:=total_berat-berat_brg;
        Posisi_Barang;{prosedur}
      end
    else
      begin
        masuk:=false;
        total_berat:=total_berat-berat_brg;
      end;
  end
else
  begin
    Posisi_Barang; {prosedur}
  end;
```

    Dalam prosedur Posisi_Barang terdapat proses pencarian tempat kosong untuk meletakkan barang dan pengisian koordinat ruang dengan kode barang untuk menandai bahwa koordinat ruang tersebut telah terisi barang.

  - Pencarian Tempat Kosong

    Saat ditemukan satu koordinat kosong dalam variabel posisi (misalnya posisi[1,1,1]=’’) maka akan dipanggil prosedur Tempat_Kosong yang mewakili posisi barang saat itu (terdapat 6 posisi barang) untuk mencari tempat kosong lainnya sejauh panjang, lebar, dan tinggi barang. Jika terpenuhi, maka variabel masuk (tipe : *boolean*) diisi *true* yang artinya proses boleh dilanjutkan ke tahap berikutnya. Prioritas penempatan

barang dalam ruang yang didahulukan adalah pertama sejajar sumbu X (lebar ruang), kemudian kedua sejajar sumbu Y (tinggi ruang), kemudian ketiga sejajar sumbu Z (panjang ruang).

Implementasi metode algoritma genetik dalam proses penyusunan barang yaitu dengan meletakkan secara alami barang-barang yang diwakili oleh kode barang dan posisinya pada gen-gen dalam satu kromosom mulai gen pertama sampai gen terakhir. Peletakkan secara alami diartikan bahwa proses peletakkan barang hanya memperhatikan bisa tidaknya suatu barang dengan posisinya pada gen masuk ke dalam ruang dengan *space* tertentu tanpa mengubah posisi barang. Jika *space* tertentu tersebut tidak dapat memenuhi panjang, lebar, dan tinggi barang, maka koordinat dalam ruang akan diubah sampai barang tersebut bisa masuk sepenuhnya ke dalam ruang kecuali koordinat telah mencapai panjang, lebar, dan tinggi maksimal ruang.

{program pencarian tempat kosong (posisi barang = 1)}

```
procedure Tempat_Kosong1;
var k,l,m:integer;
begin
for m:=z_pass to z_pass+p_brg do
  begin
    for l:=y_pass to y_pass+t_brg do
      begin
        for k:=x_pass to x_pass+l_brg do
           begin
               if           (posisi[k,l,m]='')           and
                  (x_pass+l_brg<=l_ruang+1)             and
                  (y_pass+t_brg<=t_ruang+1)             and
                  (z_pass+p_brg<=p_ruang+1) then
                begin
                 masuk:=true;
                end
              else
                 begin
                   masuk:=false;
                   exit;
                 end;
           end;
        end;
  end;
end;
```

- Pengisian Koordinat Ruang

Pengisian koordinat ruang dilakukan jika proses pencarian tempat kosong menghasilkan nilai variabel masuk=*true*. Koordinat dalam ruang yang telah ditemukan dalam proses pencarian tempat kosong sejauh panjang, lebar, tinggi barang ditandai dengan kode barang tersebut yang artinya bahwa barang lain tidak boleh menempati koordinat tersebut.

{program pengisian koordinat ruang (posisi barang = 1)}

```
if masuk then
 begin
  for z_koord:=z to z+p_brg-1 do
    begin
     for y_koord:=y to y+t_brg-1 do
       begin
         for x_koord:=x to x+l_brg-1 do
          begin
            posisi[x_koord,y_koord,z_koord]:=temp_kodebrg;
          end;
       end;
    end;
   solusi_x:=x;
   solusi_y:=y;
   solusi_z:=z;
   exit;
 end;
```

3.4.2.5. Perhitungan *Fitness*

Pada setiap akhir dari proses penyusunan barang ke dalam ruang akan dilakukan tiga teknik perhitungan *fitness* yaitu sebagai berikut :

1. Perhitungan jumlah satuan ruang yang terisi oleh barang

   Prinsip kerjanya yaitu semakin banyak jumlah satuan ruang yang terisi oleh barang maka semakin tinggi pula nilai *fitness*-nya, karena jumlah barang yang dapat dimasukkan ke dalam ruang akan semakin banyak pula. Perhitungan jumlah satuan ruang yang terisi oleh barang dibagi dalam empat tahap yaitu sebagai berikut :

   - Perhitungan seluruh ruang

     {program perhitungan fitness seluruh ruang}

```
terisi:=0;
for z:=1 to p_ruang+1 do
  begin
    for y:=1 to t_ruang+1 do
```

```
      begin
        for x:=1 to l_ruang+1 do
          begin
            if posisi[x,y,z]<>'' then
              begin
                inc(terisi);
              end;
          end;
      end;
  end;
```

- Perhitungan ¼ tinggi ruang

  {program perhitungan fitness ¼ tinggi ruang}

```
kosong1per4:=0;
for z:=1 to p_ruang do
  begin
    for y:=1 to (t_ruang div 4) do
      begin
        for x:=1 to l_ruang do
          begin
            if posisi[x,y,z]='' then inc(kosong1per4);
          end;
      end;
  end;
```

- Perhitungan ½tinggi ruang

  {program perhitungan fitness ½ tinggi ruang}

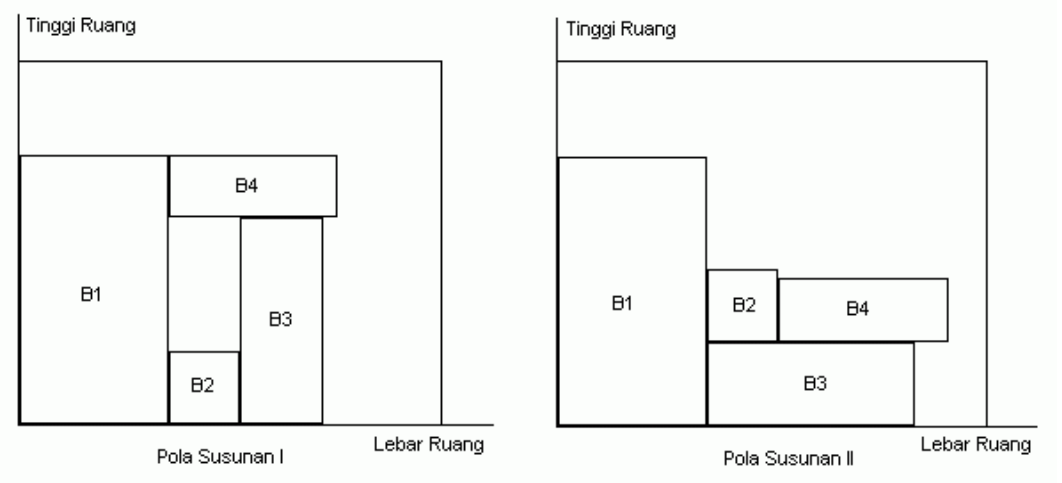
```
kosong1per2:=0;
for z:=1 to p_ruang do
  begin
    for y:=1 to (t_ruang div 2) do
      begin
        for x:=1 to l_ruang do
          begin
            if posisi[x,y,z]='' then inc(kosong1per2);
          end;
      end;
  end;
```

- Perhitungan ¾ tinggi ruang

  {program perhitungan fitness ¾ tinggi ruang}

```
kosong3per4:=0;
for z:=1 to p_ruang do
  begin
    for y:=1 to ((t_ruang*3) div 4) do
      begin
        for x:=1 to l_ruang do
          begin
            if posisi[x,y,z]='' then inc(kosong3per4);
          end;
      end;
  end;
```

Perhitungan *fitness* dalam ¼, ½, ¾ tinggi ruang dimaksudkan untuk mengantisipasi apabila terdapat *fitness* yang sama besar dari dua individu. Contoh hasil *fitness* yang sama besar dapat dilihat pada gambar berikut ini :



Gambar 3.14. Contoh Hasil *Fitness* Sama Besar

Pada Gambar 3.14 dapat dilihat bahwa *fitness* yang didapat dari Pola Susunan I dan Pola Susunan II adalah sama besar jika dipakai teknik menghitung jumlah satuan ruang yang terisi oleh barang, karena seluruh barang dapat dimasukkan ke dalam ruang. Tetapi jika dilihat secara tampilan tiga dimensi, maka Pola Susunan II lebih baik untuk diterapkan daripada Pola Susunan I karena tidak terdapat rongga kosong diantara tumpukan barang. Oleh karena itu diperlukan teknik perhitungan *fitness* dalam ¼, ½, ¾ tinggi ruang yang mencari jumlah satuan ruang terkecil yang tidak terisi oleh barang untuk dijadikan pola susunan dengan fitness terbaik.

2. Perhitungan jumlah barang yang seimbang (tidak roboh) di dalam ruang

Prinsip kerjanya yaitu semakin banyak barang yang menempati posisinya secara seimbang (titik berat barang berada diatas barang lain, kecuali barang yang terletak di lantai ruang), maka semakin tinggi pula *fitness* susunan barang dalam ruang tersebut, karena semakin sedikit jumlah barang yang roboh.

{program pengecekkan keseimbangan barang (posisi barang = 1)}

```
procedure Seimbang1;
begin
if y_pass<>1 then
  begin
    z_ttbrg:=p_brg div 2;
    x_ttbrg:=l_brg div 2;
    if   posisi_bay[x_pass+x_ttbrg,y_pass-1,z_pass+z_ttbrg]<>''
      then masuk:=true
    else masuk:=false;
  end
else masuk:=true;
end;
```

3. Perhitungan jumlah barang yang tidak rusak (utuh) di dalam ruang

Prinsip kerjanya yaitu semakin banyak barang yang tidak hancur dalam tumpukan (total berat barang yang berada diatas kurang dari atau sama dengan beban maksimal yang dapat ditanggung barang dibawah), maka semakin tinggi pula *fitness* yang dihasilkan oleh susunan barang dalam ruang tersebut.

{program pengecekkan beban maksimal barang (posisi barang = 1)}

```
procedure Cari_Beban1;
var l,m,o,p,q:integer;
    brgkode,sip,brgkode1:string;
begin
{mencari kode barang yang berada diatas}
  brgkode:='';
  brgkode1:='';
  m:=0;
  setlength(brgkd,1);
  for l:=y_pass+t_brg to t_ruang do
    begin
        stop:=false;
        for p:=z_pass to z_pass+p_brg-1 do
          begin
             for o:=x_pass to x_pass+l_brg-1 do
               begin
                  brgkode:=posisi_bay[o,l,p];
                  if (brgkode='') then stop:=true
                  else
                    begin
                      stop:=false;
                      break;
                    end;
               end;
             if stop=false then break;
          end;

        if stop then break
        else
          begin
            for p:=z_pass to z_pass+p_brg-1 do
              begin
```

```
                    for o:=x_pass to x_pass+l_brg-1 do
                      begin
                          if (brgkode1<>brgkode) and (brgkode<>'')
                          then
                            begin
                               inc(m);
                               setlength(brgkd,m+2);
                               brgkode1:=brgkode;
                               brgkd[m]:=brgkode;
                            end;
                      end;
                end;
            end;
    end;

{barang diatas lebih dari 1}
  if m>1 then
    begin
      beratgab:=0;
      for q:=1 to m do
        begin
          brgkode:=brgkd[q];
          sip:='';
          for o:=1 to length(brgkode) do
            begin
              if brgkode[o]<>'/' then
                 sip:=sip+brgkode[o]
              else break;
            end;
          main.QueryBarangHapus.SQL.Clear;
           main.QueryBarangHapus.SQL.Add('select * from Barang
           where Kode_Barang=:kb;');
          main.QueryBarangHapus.Params[0].AsString:=sip;
          main.QueryBarangHapus.ExecSQL;
          main.QueryBarangHapus.Close;
          main.QueryBarangHapus.Open;

           beratgab:=strtoint(main.QueryBarangHapus['Berat'])+b
           eratgab; {mencari total berat barang diatas}
          if beratgab<=bebanmaks then masuk:=true
          else
            begin
              masuk:=false;
              exit;
            end;
        end;
    end
  else

{barang diatas = 1}
  if m=1 then
    begin
       brgkode:=brgkd[1];
       sip:='';
       for o:=1 to length(brgkode) do
         begin
           if brgkode[o]<>'/' then
              sip:=sip+brgkode[o]
           else break;
```

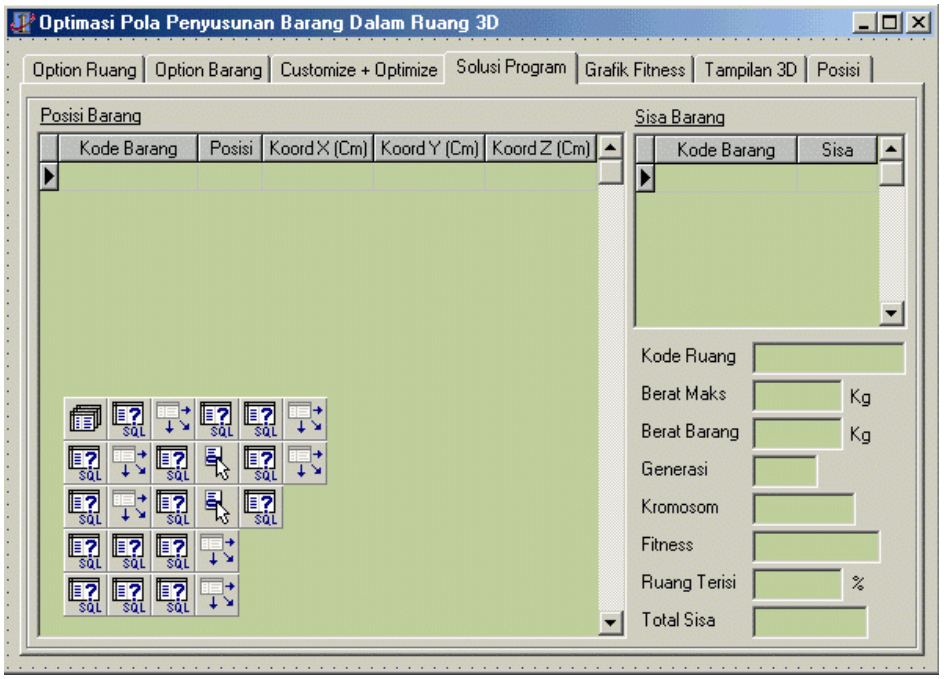
```
          end;
         main.QueryBarangHapus.SQL.Clear;
          main.QueryBarangHapus.SQL.Add('select  *  from  Barang
          where Kode_Barang=:kb;');
         main.QueryBarangHapus.Params[0].AsString:=sip;
         main.QueryBarangHapus.ExecSQL;
         main.QueryBarangHapus.Close;
         main.QueryBarangHapus.Open;
        if
        (strtoint(main.QueryBarangHapus['Berat'])<=bebanmaks)
        then masuk:=true
         else
           begin
             masuk:=false;
             exit;
           end;
    end
  else masuk:=true;
end;
```

Setelah semua teknik pencarian *fitness* dilakukan maka nilai-nilai *fitness* yang didapat digabungkan untuk mendapat nilai *fitness* akhir dari susunan barang. Semakin tinggi nilai *fitness* gabungan tersebut, maka semakin baik pula pola susunan barang dalam ruang tersebut.

3.4.3. Solusi Optimasi (*Output*)

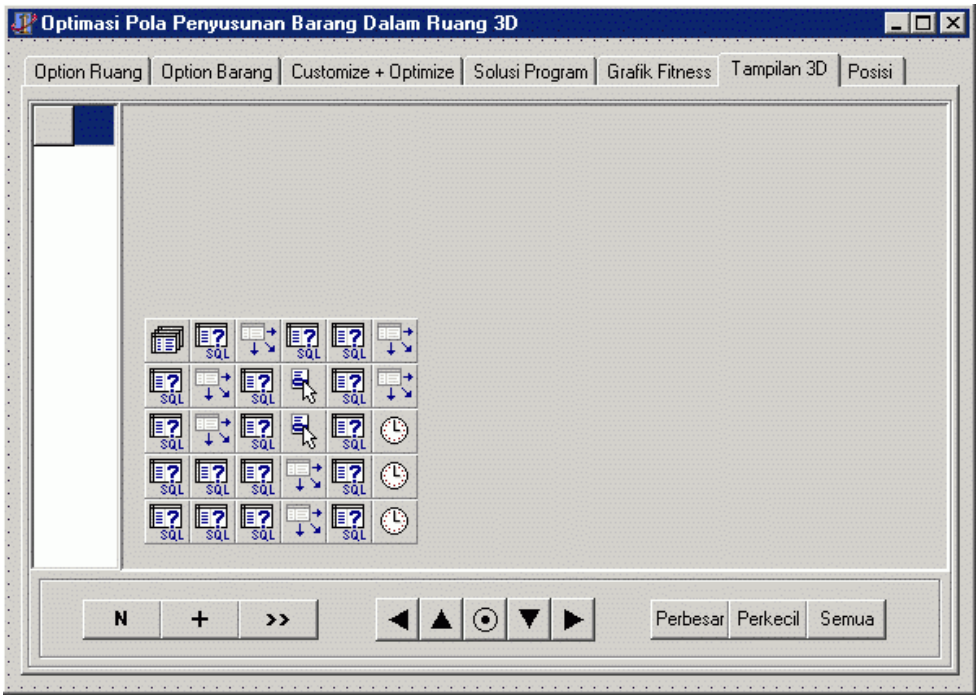
Solusi pola penyusunan barang terbaik hasil dari proses optimasi ditampilkan dalam dua bentuk yaitu :

- Solusi Program yang berisi tabel koordinat awal tempat meletakkan barang dalam ruang beserta dengan posisinya masing-masing. Disamping itu ditampilkan pula tabel sisa barang yang berisi barang apa saja yang tidak dapat masuk ke dalam ruang beserta jumlahnya. Tampilan dari Solusi Program dapat dilihat pada gambar berikut ini :



Gambar 3.15. Tampilan Solusi Program

- Tampilan animasi tiga dimensi yang berisi posisi-posisi barang dalam ruang dengan perbandingan tertentu dari keadaan sebenarnya. Tampilan animasi tiga dimensi dapat dilihat pada gambar berikut ini :



Gambar 3.16. Tampilan Animasi 3D