BAB V
IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi sistem berdasarkan perancangan yang telah diuraikan pada Bab sebelumnya. Bab implementasi membahas implementasi basis data, implementasi sistem, dan implementasi antarmuka sistem.

5.1 Implementasi Basis Data

Subbab ini menjelaskan mengenai implementasi basis data yang digunakan pada sistem penerapan algoritma *evolution strategies* untuk optimasi distribusi barang dua tahap. Sistem penerapan algoritma *evolution strategies* untuk optimasi distribusi barang dua tahap menggunakan sebuah basis data dengan nama "es" dengan enam tabel yang terdiri dari :

1. Tabel Parameter ES digunakan untuk menyimpan nilai parameter algoritma *evolution strategies.*

Tabel 5.1 Implementasi Tabel Parameter ES

Nama tabel : parameter_es			
No.	**Nama Kolom**	**Tipe Data**	**Keterangan**
1	id_par	Integer	Id parameter ES digunakan sebagai primary key
2	Miu	Integer	Ukuran populasi
3	Lambda	Integer	Ukuran *offspring*
4	par_a	Float	Nilai parameter *a*
5	sigma_min	Integer	Batas bawah *strategy parameter*
6	sigma_max	Integer	Batas atas *strategy parameter*
7	n_generasi	Integer	Jumlah generasi
8	r_segmen_1	Integer	Rasio mutasi segmen 1
9	r_segmen_2	Integer	Rasio mutasi segmen 2
10	r_dua_segmen	Integer	Rasio mutasi segmen 1 dan 2

2. Tabel Produsen, menyimpan data produsen dan kapasitas persediaan produsen.

Tabel 5.2 Implementasi Tabel Produsen

Nama tabel : produsen			
No.	**Nama Kolom**	**Tipe Data**	**Keterangan**
1	id_produsen	Varchar	Id produsen digunakan sebagai primary key
2	Kapasitas	Integer	Kapasitas persediaan yang dimiliki produsen

3. Tabel Agen, menyimpan data agen dan kebutuhan agen.

Tabel 5.3 Implementasi Tabel Agen

Nama tabel : agen			
No.	**Nama Kolom**	**Tipe Data**	**Keterangan**
1	id_agen	Varchar	Id agen digunakan sebagai primary key
2	Kapasitas	Integer	Jumlah permintaan agen

4. Tabel Sub Agen, menyimpan data sub agen dan kebutuhan sub agen.

Tabel 5.4 Implementasi Tabel Sub Agen

Nama tabel : subagen			
No.	**Nama Kolom**	**Tipe Data**	**Keterangan**
1	id_subagen	Varchar	Id subagen digunakan sebagai primary key
2	Kapasitas	Integer	Jumlah permintaan sub agen

5. Tabel Biaya Distribusi Tahap 1, menyimpan biaya distribusi dari produsen ke agen.

Tabel 5.5 Implementasi Tabel Biaya Distribusi Tahap 1

Nama tabel : biaya_t1			
No.	**Nama Kolom**	**Tipe Data**	**Keterangan**
1	id_biaya_t1	Varchar	Id biaya tahap 1 digunakan sebagai primary key
2	Produsen	Varchar	Id produsen
3	Agen	Varchar	Id agen
4	biaya	Integer	Biaya distribusi tahap 1, yaitu dari produsen ke agen.

6. Tabel Biaya Distribusi Tahap 2, menyimpan biaya distribusi dari agen ke sub agen.

Tabel 5.6 Implementasi Tabel Biaya Distribusi Tahap 2

Nama tabel : biaya_t2			
No.	**Nama Kolom**	**Tipe Data**	**Keterangan**
1	id_biaya_t2	Varchar	Id biaya tahap 2 digunakan sebagai primary key
2	Agen	Varchar	Id agen
3	Subagen	Varchar	Id sub agen
4	Biaya	Integer	Biaya distribusi tahap 2, yaitu dari agen ke sub agen.

5.2 Implementasi Sistem

Berdasarkan perancangan yang telah diuraikan pada Bab IV dalam subbab ini membahas mengenai implementasi sistem penerapan algoritma *evolution strategies* untuk optimasi distribusi barang dua tahap ke dalam kode program.

5.2.1 Implementasi Input Data

Data input yang diperlukan dalam proses optimasi distribusi barang dua tahap berupa data input parameter kasus dan parameter algoritma *evolution strategies*. Implementasi proses input data dapat dilihat pada *Source Code* 5.1.

```
function Add_produsen(){
  include("config.php");
  $id = $_POST['id_produsen'];
  $kap = $_POST['kapasitas'];
  $sql =
            "INSERT INTO produsen(id_produsen, kapasitas)
            VALUES('$id','$kap');";
  mysql_query($sql, $dbConn);
  $sql = "SELECT count(*) as total from agen;";
  $result = mysql_query($sql, $dbConn);
  $data = mysql_fetch_assoc($result);
  $x = $data['total'];
  for ($i = 1; $i <= $x; $i++) {
    $agen = 'agen'.$i;
    $id_biaya_t1 = $id."_".$agen;
    $sql = "INSERT INTO biaya_t1(
                  id_biaya_t1, produsen, agen, biaya)
            VALUES(
                  '$id_biaya_t1', '$id', '$agen', 0);";
      mysql_query($sql, $dbConn);
  }
}
function Add_Agen(){
  include("config.php");
  $id = $_POST['id_agen'];
  $kap = $_POST['kapasitas'];
  $sql = "INSERT INTO agen(id_agen, kapasitas)
            VALUES('$id','$kap');";
  mysql_query($sql, $dbConn);
  $sql = "SELECT count(*) as total from produsen;";
  $result = mysql_query($sql, $dbConn);
  $data = mysql_fetch_assoc($result);
  $x = $data['total'];
  for ($i = 1; $i <= $x; $i++) {
```

```
35      $produsen = 'produsen'.$i;
36      $id_biaya_t1 = $produsen."_".$id;
37      $sql = "INSERT INTO biaya_t1(
38                      id_biaya_t1, produsen, agen, biaya)
39              VALUES(
40                      '$id_biaya_t1', '$produsen', '$id', 0);";
41       mysql_query($sql, $dbConn);
42    }
43    $sql = "SELECT count(*) as total from subagen;";
44    $result = mysql_query($sql, $dbConn);
45    $data = mysql_fetch_assoc($result);
46    $x = $data['total'];
47    for ($i = 1; $i <= $x; $i++) {
48      $subagen = 'sub'.$i;
49      $id_biaya_t2 = $id."_".$subagen;
50      $sql = "INSERT INTO biaya_t2(
51                      id_biaya_t2, agen, subagen, biaya)
52              VALUES(
53                      '$id_biaya_t2', '$id', '$subagen', 0);";
54        mysql_query($sql, $dbConn);
55    }
56  }
57  function Add_SubAgen(){
58    include("config.php");
59    $id = $_POST['id_subagen'];
60    $kap = $_POST['kapasitas'];
61    $sql = "INSERT INTO subagen(id_subagen, kapasitas)
62              VALUES('$id','$kap');";
63    mysql_query($sql, $dbConn);
64    $sql = "SELECT count(*) as total from agen;";
65    $result = mysql_query($sql, $dbConn);
66    $data = mysql_fetch_assoc($result);
67    $x = $data['total'];
68    for ($i = 1; $i <= $x; $i++) {
69      $agen = 'agen'.$i;
70        $id_biaya_t2 = $agen."_".$id;
71      $sql =
72            "INSERT INTO biaya_t2(
73                      id_biaya_t2, agen, subagen, biaya)
```

```
74              VALUES(
75                      '$id_biaya_t2', '$agen', '$id', 0);";
76          mysql_query($sql, $dbConn);
77      }
78  }
79  function Edit_Parameter(){
80      include("config.php");
81      $id = 0;
82      $miu = $_POST["miu"];
83      $lambda = $_POST["lambda"];
84      $par_a = $_POST["par_a"];
85      $sigma_min = $_POST["sigma_min"];
86      $sigma_max = $_POST["sigma_max"];
87      $n_generasi = $_POST["n_generasi"];
88      $r_segmen_1 = $_POST["r_segmen_1"];
89      $r_segmen_2 = $_POST["r_segmen_2"];
90      $r_dua_segmen = $_POST["r_dua_segmen"];
91      $sum_segmen = $r_segmen_1 + $r_segmen_2 + $r_dua_segmen;
92      if ($sum_segmen == 100) {
93          $sql =
94                "UPDATE parameter_es SET
95                miu='$miu',lambda='$lambda',par_a='$par_a',
96                sigma_min='$sigma_min',sigma_max='$sigma_max',
97                n_generasi='$n_generasi',
98                r_segmen_1='$r_segmen_1',
99                r_segmen_2='$r_segmen_2',
100               r_dua_segmen='$r_dua_segmen'
101               WHERE id_par='$id'";
102         mysql_query($sql, $dbConn);
103         $error_messages = "&err=Changes saved&alr=success";
104     }
105     else {
106         $error_messages    =    "&err=<strong>Warning!</strong>
107 Cannot   save   query   data.<br>Jumlah   <u>Perbandingan   %
108 Segmen</u> harus 100&alr=danger";
109     }
```

Source Code 5.1 Implementasi Input Data

Implementasi input data yang dijelaskan *Source Code* 5.1. Baris 1-22 merupakan fungsi untuk menambahkan data produsen. Baris 3-8 merupakan proses menambahkan id dan kapasitas produsen ke tabel produsen dalam *database.* Baris 9-21 adalah proses membentuk matriks biaya distribusi tahap 1 berdasarkan jumlah produsen yang diinputkan. Baris tersebut juga melakukan proses penyimpanan biaya produksi tahap 1 pada tabel biaya_t1. Baris 23-56 adalah fungsi menambahkan data agen. Baris 25-29 adalah proses menambahkan id dan jumlah permintaan agen ke tabel agen dalam *database.* Baris 30-42 adalah proses menambahkan biaya distribusi tahap 1 ke tabel biaya_t1. Sedangkan baris 43-55 adalah proses membentuk matriks biaya tahap 2 berdasarkan jumlah agen. Fungsi menambahkan data sub agen ditunjukkan pada baris 57-78. Baris 59-63 adalah proses untuk menambahkan id dan jumlah permintaan sub agen ke tabel subagen. Baris 64-77 adalah proses untuk menambahkan biaya distribusi tahap 2 ke tabel biaya_t2. Baris 79-102 adalah fungsi untuk mengedit parameter ES yang sebelumnya telah tersimpan pada tabel parameter_es di *database.*

5.2.2 Implementasi *Generate* Populasi

Implementasi *generate* populasi terdiri dari dua proses utama, yaitu proses penentuan posisi gen dan pembangkitan populasi awal. Implementasi penentuan posisi gen ditunjukkan pada *Source Code* 5.2.

```
1  $jml_produsen = hit_bok("produsen");
2  $jml_agen = hit_bok("agen");
3  $jml_subagen = hit_bok("subagen");
4  $kolom = $jml_agen;
5  $baris = $jml_produsen;
6  $Representasi_T1=0;
7  for ($i = 0; $i <= $baris; $i++) {
8      for ($j = 0; $j <= $kolom; $j++) {
9          if(($i == 0) AND ($j == 0)){
10             echo "<td></td>";}
11         else if(($i == 0) AND ($j != 0)){
12             echo "<td>Agen$j</td>";}
13         else if(($i != 0) AND ($j == 0)){
14             echo "<td>Produsen$i</td>";}
15         else {
```

```
16                  $rep_show = $Representasi_T1+1;
17                  $str = strval($rep_show);
18                  $RT_1[$str]['produsen'] = $i;
19                  $RT_1[$str]['agen'] = $j;
20                  echo "<td>$rep_show</td>";
21                  $Representasi_T1++;
22              }
23          }
24  }
25  $kolom = $jml_subagen;
26  $baris = $jml_agen;
27  $Representasi_T2=0;
28  for ($i = 0; $i <= $baris; $i++) {
29      for ($j = 0; $j <= $kolom; $j++) {
30          if(($i == 0) AND ($j == 0)){
31              echo "<td></td>";}
32          else if(($i == 0) AND ($j != 0)){
33              echo "<td>SubAgen$j</td>";}
34          else if(($i != 0) AND ($j == 0)){
35              echo "<td>Agen$i</td>";}
36          else {
37              $rep_show = $Representasi_T2+1;
38              $str = strval($rep_show);
39              $RT_2[$str]['agen'] = $i;
40              $RT_2[$str]['subagen'] = $j;
41              $Representasi_T2++;
42          }
43      }
44  }
```

Source Code 5.2 Implementasi Penentuan Posisi Gen

Individu yang dibentuk terdiri dari dua segmen. Segmen pertama merepresentasikan distribusi tahap 1, yaitu distribusi dari produsen ke agen. Segmen kedua merupakan representasi distribusi tahap 2, yaitu distribusi dari agen ke sub agen. Baris 1-3 merupakan proses mengambil jumlah produsen, agen, dan sub agen. Posisi gen pada kromosom segmen pertama ditentukan oleh hasil perkalian jumlah produsen dan agen ditunjukkan pada baris kode 5-25. Posisi gen pada kromosom segmen kedua ditentukan oleh hasil perkalian jumlah agen dan sub

agen ditunjukkan pada baris kode 27-44. Pembentukan individu selanjutnya dilakukan dengan mengacak bilangan permutasi pada setiap segmen. Populasi awal yang dibangkitkan sesuai dengan nilai miu (μ) yang dimasukkan oleh *user*. Pembangkitan individu yang menjadi populasi awal ditunjukkan pada *Source Code* 5.3.

```
1   $posisi_T1 = array();
2   $posisi_T2 = array();
3   for ($row = 0; $row < ($Representasi_T1); $row++) {
4           $show = $row+1;
5           array_push($posisi_T1,$show);
6         }
7         for ($row = 0; $row < ($Representasi_T2); $row++) {
8         $show = $row+1;
9         array_push($posisi_T2,$show);
10        }
11  for ($r = 0; $r < $miu; $r++) {
12        $P_T1[$r] = $posisi_T1;
13        shuffle($P_T1[$r]);
14        $P_T2[$r] = $posisi_T2;
15        shuffle($P_T2[$r]);
16        }
```

Source Code 5.3 Implementasi Pembangkitan Populasi

Baris 1-3 dalam *Source Code* 5.3 merupakan inisialisasi array untuk kromosom segmen 1 dan segmen2. Baris 3-6 adalah proses untuk mengacak angka permutasi pada array kromosom segmen 1. Baris 7-10 adalah proses untuk mengacak angka permutasi pada array kromosom segmen 2. Baris 11-16 adalah proses untuk membangkitkan individu sebanyak jumlah *miu* secara *random.*

5.2.3 Implementasi Menghitung Fitness

Nilai *fitness* diperoleh dari hasil perhitungan yang melibatkan data kapasitas produsen, kebutuhan agen, kebutuhan sub agen. dan data biaya distribusi. Implementasi perhitungan nilai *fitness* ditunjukkan pada *Source Code* 5.4.

```
1   function Hitung_Fitness($t1x,$t2x){
2         $Fit = $t1x + $t2x;
3         $Fit = round($Fit, 0);
```

```
4         $Fit_Digits = strlen((string) $Fit);
5         $Pembagi = 1;
6         for ($i = 1; $i < $Fit_Digits; $i++) {
7               $Pembagi *= 10;
8         }
9         $Fit = $Pembagi / $Fit;
10        return $Fit;
11  }
```

Source Code 5.4 Implementasi Perhitungan *Fitness*

Baris 1-11 pada *Source code* 5.4 merupakan fungsi menghitung nilai *fitness*. Baris 9 menunjukkan bahwa nilai *fitness* diperoleh dengan menggunakan rumus minimasi sesuai pada Persamaan 2.9. dimana *constanta* yang diinisialisasikan dengan variabel $Pembagi dibagi dengan total biaya distribusi dua tahap yang diperoleh berdasarkan Persamaan 2.1. Implementasi perhitungan total biaya distribusi ditunjukkan pada *Source Code* 5.5.

```
1   Function Show_Rep_fitness_T1($x,$ke,$Arr_P,$jml_produsen,
2             $jml_agen,$produsen,$agen,$RT_1,$B_T1) {
3         $k = $ke;
4         $thp = 1;
5         $total_harga = 0;
6         $temp_P = $Arr_P;
7         foreach ($temp_P as $value){
8               $str = strval($value);
9               $jalur1 = $RT_1[$str]['produsen'];
10              $jalur2 = $RT_1[$str]['agen'];
11              $ns = 'produsen'.$jalur1;
12              $na = 'agen'.$jalur2;
13              $ck = $produsen_temp[$ns] - $agen_temp[$na];
14              $harga_show = $agen_temp[$na];
15              $produsen_temp[$ns] = $ck;
16              $agen_temp[$na] -= $agen_temp[$na];
17              if ($ck < 0) {
18                    $harga_show = $harga_show + $ck;
19                    $agen_temp[$na] -= $ck;
20                    $produsen_temp[$ns] -= $ck;
21              }
```

```
22              $g_produsen = 'produsen'.$jalur1;
23              $g_agen = 'agen'.$jalur2;
24              $b = $B_T1[$g_produsen][$g_agen];
25              $harga_show = $harga_show * $b;
26              $total_harga += $harga_show;
27        return $total_harga;
28  }
29  Function Show_Rep_fitness_T2($x,$ke,$Arr_P,$jml_agen,
30        $jml_subagen,$agen,$subagen,$RT_2,$B_T2) {
31        $k = $ke;
32        $thp = 2;
33        $total_harga = 0;
34        $temp_P = $Arr_P;
35        foreach ($temp_P as $value){
36              $str = strval($value);
37              $jalur1 = $RT_2[$str]['agen'];
38              $jalur2 = $RT_2[$str]['subagen'];
39              $ns = 'agen'.$jalur1;
40              $na = 'sub'.$jalur2;
41              $ck = $agen_temp[$ns] - $subagen_temp[$na];
42              $harga_show = $subagen_temp[$na];
43              $agen_temp[$ns] = $ck;
44              $subagen_temp[$na] -= $subagen_temp[$na];
45              if ($ck < 0) {
46                    $harga_show = $harga_show + $ck; //Harga
47                    $subagen_temp[$na] -= $ck;
48                    $agen_temp[$ns] -= $ck;
49                    }
50              $g_agen = 'agen'.$jalur1;
51              $g_subagen = 'sub'.$jalur2;
52              $b = $B_T2[$g_agen][$g_subagen];
53              $harga_show = $harga_show * $b;
54              $total_harga += $harga_show;
55        return $total_harga;
56  }
```

Source Code 5.5 Implementasi Total Biaya Distribusi Dua Tahap

Baris 1-32 pada *Source code* 5.5 merupakan proses perhitungan biaya distribusi tahap 1. Baris 33-65 adalah proses untuk menghitung biaya distribusi

tahap 2. Sesuai dengan Persamaan 2.1. total biaya distribusi dua tahap diperoleh dari menjumlahkan biaya distribusi barang tahap 1 dan tahap 2. Biaya distribusi barang pada setiap tahapan diperoleh dengan mengalikan biaya distribusi tiap unit barang dengan jumlah barang yang didistribusikan pada setiap jalur distribusi.

5.2.4 Implementasi Reproduksi

Pada siklus ES(μ+λ) *offspring* dihasilkan dari proses mutasi. Metode mutasi yang digunakan adalah *exchange mutation.* Mutasi dilakukan pada segmen 1, segmen 2, serta segmen 1 dan 2. Implementasi proses reproduksi ditunjukkan pada *Source Code* 5.6.

```
$jml_offspring = $miu * $lambda;
function acak_offspring($sgmn,$T1_C_x,$T2_C_x,
      $Representasi_T1,$Representasi_T2,$Sigma_P_x) {
      $array_T_temp = 0;
      $xi = ceil($Sigma_P_x);
      if($sgmn >= ($jml_offspring_segmen1 +
         $jml_offspring_segmen2)) {
      //$v = "Dua Segmen";
      for ($i = 0; $i < $xi; $i++) {
         $random_t1 = array_rand($T1_C_x,2);
         $random_t2 = array_rand($T2_C_x,2);
            $ck1 = $random_t1[0];
            $ck2 = $random_t1[1];
            $array_T_temp = $T1_C_x[$ck1];
            $T1_C_x[$ck1] = $T1_C_x[$ck2];
            $T1_C_x[$ck2] = $array_T_temp;
            $ck1 = $random_t2[0];
            $ck2 = $random_t2[1];
            $array_T_temp = $T2_C_x[$ck1];
            $T2_C_x[$ck1] = $T2_C_x[$ck2];
            $T2_C_x[$ck2] = $array_T_temp;
            }
      }
      elseif ($sgmn >= $jml_offspring_segmen1) {
      //$v = "Segmen 2";
      for ($i = 0; $i < $xi; $i++) {
            $random_t2 = array_rand($T2_C_x,2);
```

```
28                $ck1 = $random_t2[0];
29                $ck2 = $random_t2[1];
30                $array_T_temp = $T2_C_x[$ck1];
31                $T2_C_x[$ck1] = $T2_C_x[$ck2];
32                $T2_C_x[$ck2] = $array_T_temp;
33                }
34        }else {
35        //$v = "Segmen 1";
36        for ($i = 0; $i < $xi; $i++) {
37                $random_t1 = array_rand($T1_C_x,2);
38                $ck1 = $random_t1[0];
39                $ck2 = $random_t1[1];
40                $array_T_temp = $T1_C_x[$ck1];
41                $T1_C_x[$ck1] = $T1_C_x[$ck2];
42                $T1_C_x[$ck2] = $array_T_temp;
43                }
44        }
45    return array($T1_C_x,$T2_C_x);
46 }
47 $c = 0;
48 for ($r = 0; $r < $miu; $r++) {
49    for ($n = 0; $n < $lambda; $n++) {
50       $C_T1[$c] = $P_T1[$r];
51       $C_T2[$c] = $P_T2[$r];
52       $acak_C = acak_offspring($c,$C_T1[$c],$C_T2[$c],
53                $Representasi_T1,$Representasi_T2,$Sigma_P[$r]);
54       $C_T1[$c] = $acak_C[0];
55       $C_T2[$c] = $acak_C[1];
56       $c++;
57    }
58 }
```

Source Code 5.6 Implementasi Proses Mutasi

Proses mutasi diawali dengan menentukan segmen mutasi sesuai dengan input nilai perbandingan mutasi segmen. Mutasi dilakukan dengan memilih dua gen secara acak kemudian menukar nilai pada kedua gen terpilih. Baris 1 pada *Source code* 5.6 merupakan proses inisialisasi ukuran *offspring* yang dihasilkan adalah sebanyak nilai *miu* dikalikan dengan nilai *lamda.* Baris 3-50 adalah fungsi yang

melakukan proses penentuan segmen yang akan dimutasi sesuai dengan perbandingan segmen mutasi yang diinputkan oleh *user*. Baris 8-26 adalah proses mutasi pada segmen 1 dan 2. Baris 27-37 adalah proses mutasi pada segmen 2. Baris 38-48 adalah proses mutasi pada segmen 1. Baris 53-62 adalah proses yang menghasilkan *offspring* sebanyak nilai *miu* dikalikan dengan nilai *lamda*.

Mekanisme *self adaption* digunakan untuk merubah nilai *strategy parameter* dari *offspring* berdasarkan nilai *fitness* yang dihasilkan. Proses *Self adaption* sesuai dengan aturan 1/5, dimana nilai *strategy parameter* akan dinaikkan atau diturunkan sesuai dengan persamaan 2.7. Implementasi proses *self adaption* ditunjukkan pada *Source Code* 5.7.

```
1  $c = 0;
2  for ($r = 0; $r < $miu; $r++) {
3        $temp_jumlah[$r] = 0;
4        for ($n = 0; $n < $lambda; $n++) {
5        if ($Fit_C[$c] > $Fit_P[$r]) {
6              $temp_jumlah[$r]++;}
7              $c++;
8        }
9              $banding1 = $temp_jumlah[$r] / $lambda;
10             $banding2 = 0.2;
11       if ($banding1 > $banding2) {
12             $hasil_sigma_c = 1.0 / $par_a;}
13       else if ($banding1 < $banding2){
14             $hasil_sigma_c = 1.0 * $par_a;}
15       else {
16             $hasil_sigma_c = 1.0;}
17       $Pengkali_Sigma_C[$r] = $hasil_sigma_c;
18 }
19 $c = 0;
20 for ($r = 0; $r < $miu; $r++) {
21       for ($n = 0; $n < $lambda; $n++) {
22       $Sigma_C[$c] = $Sigma_P[$r] * $Pengkali_Sigma_C[$r];
23       $Sigma_C[$c] = round($Sigma_C[$c], 4);
24       $c++;}
25 }
```

Source Code 5.7 Implementasi Proses *Self Adaption*

Baris 2-8 merupakan proses membandingkan nilai *fitness parent* dan *offspring*. Baris 11-16 merupakan proses percabangan untuk menentukan *sigma* (*strategy parameters*) dinaikkan atau diturunkan sesuai dengan persamaan 2.7. Baris 20-24 merupakan proses membangkitkan nilai *sigma offspring*.

5.2.5 Implementasi Seleksi

Metode seleksi yang digunakan adalah *elitism selection* yang melibatkan *parent* dan *offspring*. Hasil dari proses seleksi ini adalah individu baru yang akan bereproduksi pada generasi berikutnya. *Source Code* 5.8 menjelaskan implementasi proses seleksi.

```
1  for ($r = 0; $r < $miu; $r++) {
2      $A_T1[$r] = $P_T1[$r];
3      $A_T2[$r] = $P_T2[$r];
4      $Fit_A[$r] = $Fit_P[$r];
5      $Sigma_A[$r] = $Sigma_P[$r];
6      $Biaya_Total_A[$r] = $Biaya_Total_P[$r];
7  }
8  for ($r = $miu; $r < ($miu + $jml_offspring); $r++) {
9        $c = $r - $miu;
10       $A_T1[$r] = $C_T1[$c];
11       $A_T2[$r] = $C_T2[$c];
12       $Fit_A[$r] = $Fit_C[$c];
13       $Sigma_A[$r] = $Sigma_C[$c];
14       $Biaya_Total_A[$r] = $Biaya_Total_C[$c];
15 }
16 array_multisort($Fit_A, SORT_DESC, $Sigma_A,
17                 SORT_DESC, $Biaya_Total_A,
18                 $A_T1, $A_T2);
```

Source Code 5.8 Implementasi Proses Seleksi

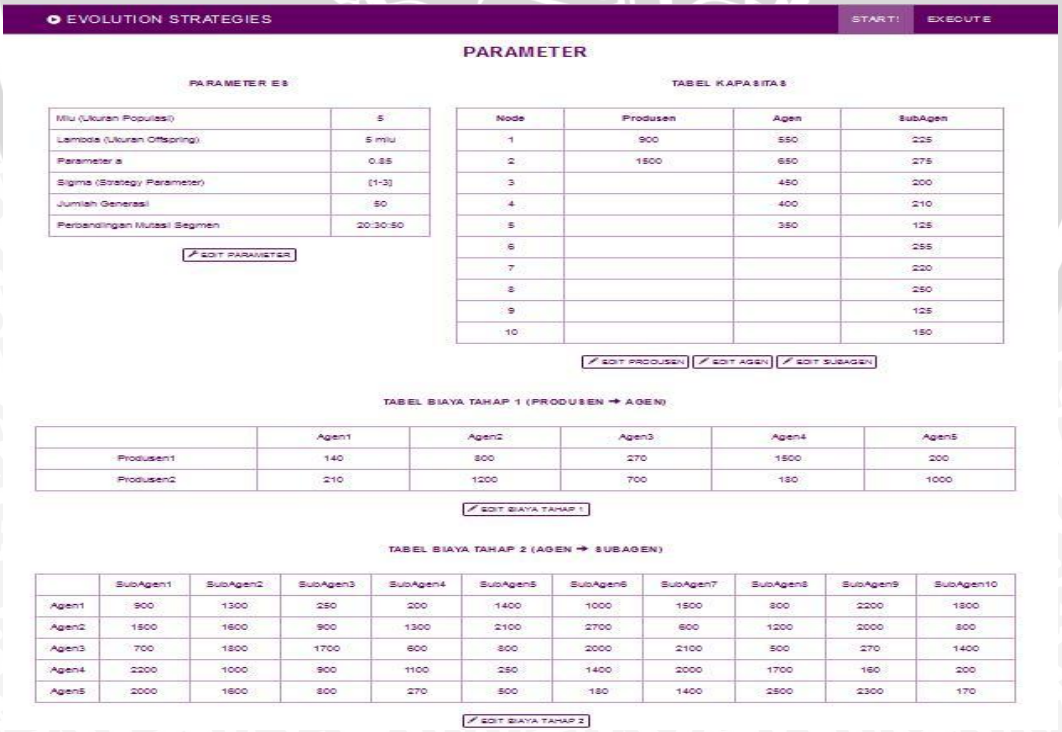
Baris 1-15 adalah proses evaluasi, yaitu mengumpulkan kromosom *parent* dan *offspring*. Baris 16-18 adalah proses mengurutkan kromosom *parent* dan *offspring* secara *descending*. Setelah pengurutan kromosom ini selanjutnya dipilih individu dengan nilai *fitness* terbaik sebanyak ukuran populasi awal untuk dijadikan *parent* pada proses reproduksi generasi berikutnya.

5.3 Implementasi Antarmuka Sistem

Berdasarkan perancangan antarmuka sistem pada Bab sebelumnya, antarmuka sistem penerapan algoritma *evolution strategies* untuk optimasi distribusi barang dua tahap terdiri dari halaman utama, halaman input parameter, dan halaman proses perhitungan. Gambar 5.1 merupakan implementasi antarmuka halaman utama sistem.

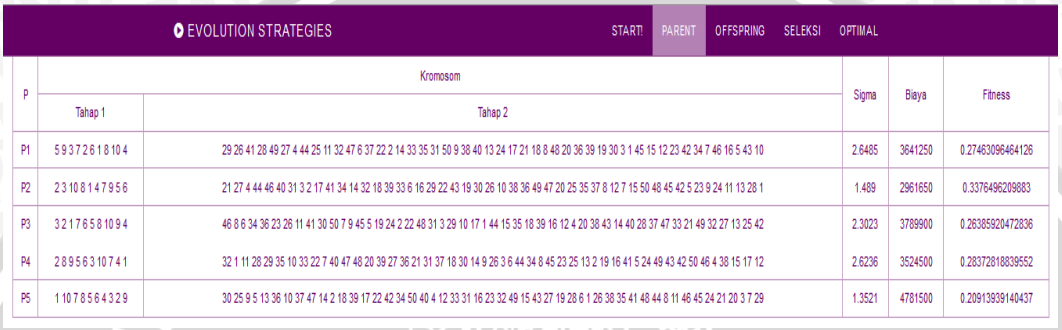
Gambar 5.1 Implementasi Antarmuka Halaman Utama Sistem

Halaman utama dari sistem memuat judul sistem dan tombol *scroll* yang berfungsi sebagai tombol *start* untuk melakukan proses input parameter. Gambar 5.2 merupakan implementasi antarmuka halaman input parameter.



Gambar 5.2 Implementasi Antarmuka Halaman Input Parameter

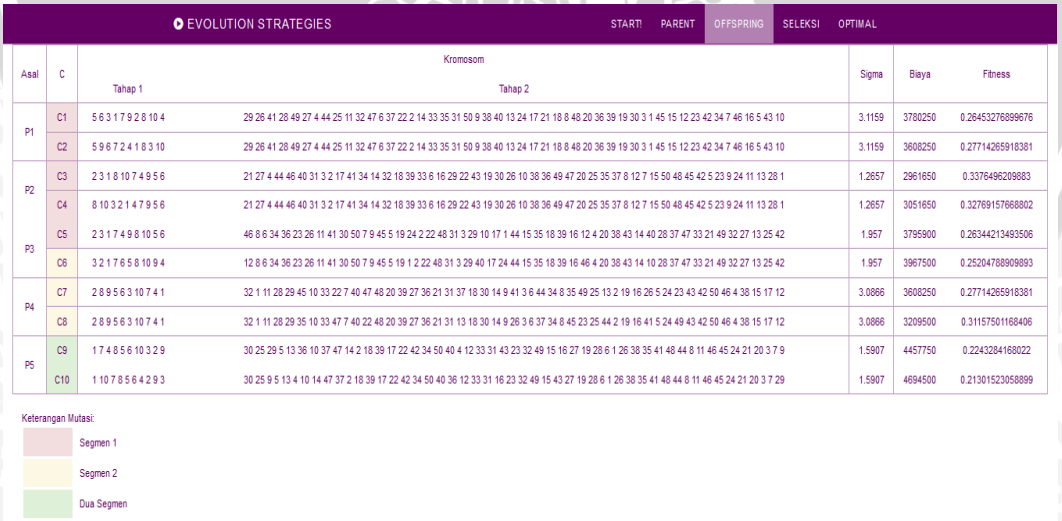
Halaman input parameter berisi tabel parameter algoritma *evolution strategies*, tabel kapasitas produsen, permintaan agen dan sub agen, serta tabel biaya distribusi tahap 1 dan biaya distribusi tahap 2. Proses perhitungan menggunakan algoritma *evolution strategies* dibagi menjadi beberapa tahapan, yaitu inisialisasi populasi awal, reproduksi, seleksi, dan pembangkitan populasi baru. Pada tab "*parent"* pada halaman proses perhitungan menampilkan inisialisasi populasi awal ditunjukkan pada Gambar 5.3.



P	Kromosom		Sigma	Biaya	Fitness
	Tahap 1	Tahap 2			
P1	5 9 3 7 2 6 1 8 10 4	29 26 41 28 49 27 4 44 25 11 32 47 6 37 22 2 14 33 35 31 50 9 38 40 13 24 17 21 18 8 48 20 36 39 19 30 3 1 45 15 12 23 42 34 7 46 16 5 43 10	2.6485	3641250	0.27463096464126
P2	2 3 10 8 1 4 7 9 5 6	21 27 4 44 46 40 31 3 2 17 41 34 14 32 18 39 33 6 16 29 22 43 19 30 26 10 38 36 49 47 20 25 35 37 8 12 7 15 50 48 45 42 5 23 9 24 11 13 28 1	1.489	2961650	0.3376496209883
P3	3 2 1 7 6 5 8 10 9 4	46 8 6 34 36 23 26 11 41 30 50 7 9 45 5 19 24 2 22 48 31 3 29 10 17 1 44 15 35 18 39 16 12 4 20 38 43 14 40 28 37 47 33 21 49 32 27 13 25 42	2.3023	3789900	0.26385920472836
P4	2 8 9 5 6 3 10 7 4 1	32 1 11 28 29 35 10 33 22 7 40 47 48 20 39 27 36 21 31 37 18 30 14 9 26 3 6 44 34 8 45 23 25 13 2 19 16 41 5 24 49 43 42 50 46 4 38 15 17 12	2.6236	3524500	0.28372818839552
P5	1 10 7 8 5 6 4 3 2 9	30 25 9 5 13 36 10 37 47 14 2 18 39 17 22 42 34 50 40 4 12 33 31 16 23 32 49 15 43 27 19 28 6 1 26 38 35 41 48 44 8 11 46 45 24 21 20 3 7 29	1.3521	4781500	0.20913939140437

Gambar 5.3 Implementasi Inisialisasi Populasi Awal

Individu pada populasi awal dibangkitkan secara *random* sebanyak input *miu* yang berikan oleh *user*. Individu dibangkitkan dalam kromosom dua segmen disertai dengan nilai *strategy parameter*, total biaya distribusi dua tahap, dan nilai *fitness*. Halaman proses perhitungan juga menampilkan hasil reproduksi pada tab "*offspring*" seperti pada Gambar 5.4.



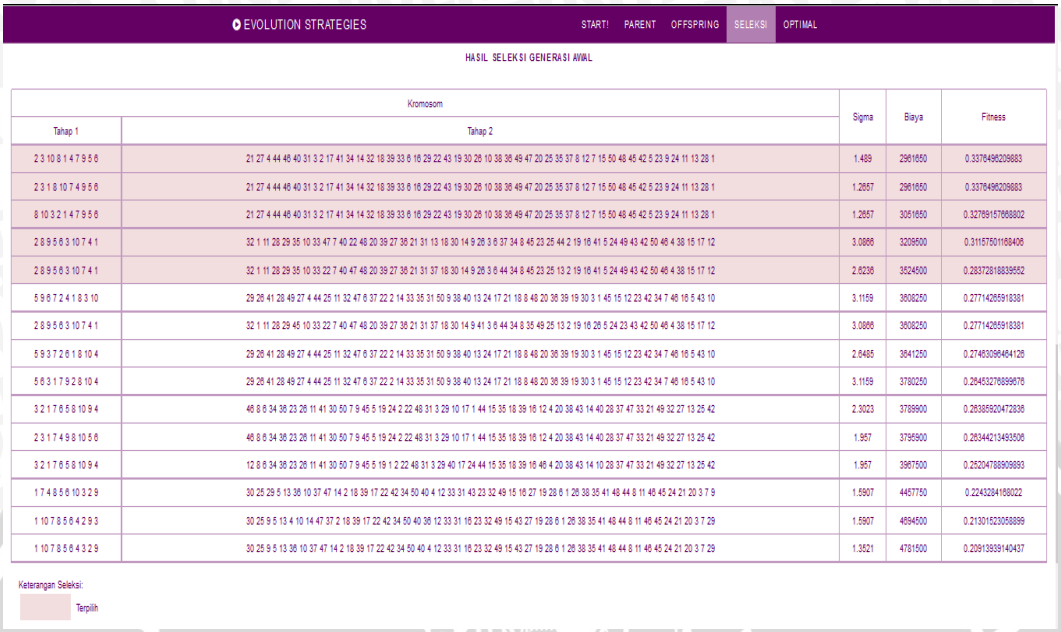
Asal	C	Kromosom		Sigma	Biaya	Fitness
		Tahap 1	Tahap 2			
P1	C1	5 6 3 1 7 9 2 8 10 4	29 26 41 28 49 27 4 44 25 11 32 47 6 37 22 2 14 33 35 31 50 9 38 40 13 24 17 21 18 8 48 20 36 39 19 30 3 1 45 15 12 23 42 34 7 46 16 5 43 10	3.1159	3780250	0.26453276899676
	C2	5 9 6 7 2 4 1 8 3 10	29 26 41 28 49 27 4 44 25 11 32 47 6 37 22 2 14 33 35 31 50 9 38 40 13 24 17 21 18 8 48 20 36 39 19 30 3 1 45 15 12 23 42 34 7 46 16 5 43 10	3.1159	3608250	0.27714265918381
P2	C3	2 3 1 8 10 7 4 9 5 6	21 27 4 44 46 40 31 3 2 17 41 34 14 32 18 39 33 6 16 29 22 43 19 30 26 10 38 36 49 47 20 25 35 37 8 12 7 15 50 48 45 42 5 23 9 24 11 13 28 1	1.2657	2961650	0.3376496209883
	C4	8 10 3 2 1 4 7 9 5 6	21 27 4 44 46 40 31 3 2 17 41 34 14 32 18 39 33 6 16 29 22 43 19 30 26 10 38 36 49 47 20 25 35 37 8 12 7 15 50 48 45 42 5 23 9 24 11 13 28 1	1.2657	3051650	0.32769157668802
P3	C5	2 3 1 7 4 9 8 10 5 6	46 8 6 34 36 23 26 11 41 30 50 7 9 45 5 19 24 2 22 48 31 3 29 10 17 1 44 15 35 18 39 16 12 4 20 38 43 14 40 28 37 47 33 21 49 32 27 13 25 42	1.957	3795900	0.26344213493506
	C6	3 2 1 7 6 5 8 10 9 4	12 8 6 34 36 23 26 11 41 30 50 7 9 45 5 19 1 2 22 48 31 3 29 40 17 24 44 15 35 18 39 16 46 4 20 38 43 14 10 28 37 47 33 21 49 32 27 13 25 42	1.957	3967500	0.25204788909893
P4	C7	2 8 9 5 6 3 10 7 4 1	32 1 11 28 29 45 10 33 22 7 40 47 48 20 39 27 36 21 31 37 18 30 14 9 41 3 6 44 34 8 35 49 25 13 2 19 16 26 5 24 23 43 42 50 46 4 38 15 17 12	3.0866	3608250	0.27714265918381
	C8	2 8 9 5 6 3 10 7 4 1	32 1 11 28 29 35 10 33 47 7 40 22 48 20 39 27 36 21 31 13 18 30 14 9 26 3 6 37 34 8 45 23 25 44 2 19 16 41 5 24 49 43 42 50 46 4 38 15 17 12	3.0866	3209500	0.31157501168406
P5	C9	1 7 4 8 5 6 10 3 2 9	30 25 29 5 13 36 10 37 47 14 2 18 39 17 22 42 34 50 40 4 12 33 31 43 23 32 49 15 16 27 19 28 6 1 26 38 35 41 48 44 8 11 46 45 24 21 20 3 7 9	1.5907	4457750	0.2243284168022
	C10	1 10 7 8 5 6 4 2 9 3	30 25 9 5 13 4 10 14 47 37 2 18 39 17 22 42 34 50 40 36 12 33 31 16 23 32 49 15 43 27 19 28 6 1 26 38 35 41 48 44 8 11 46 45 24 21 20 3 7 29	1.5907	4694500	0.21301523058899

Keterangan Mutasi:
Segmen 1
Segmen 2
Dua Segmen

Gambar 5.4 Implementasi Proses Reproduksi

Offspring dihasilkan dengan proses mutasi. Mutasi dilakukan pada segmen 1, segmen 2, maupun segmen 1 dan 2 sesuai dengan perbandingan mutasi yang

dimasukkan oleh *user*. Tab "seleksi" menampilkan hasil seleksi seperti yang ditunjukkan pada Gambar 5.5.

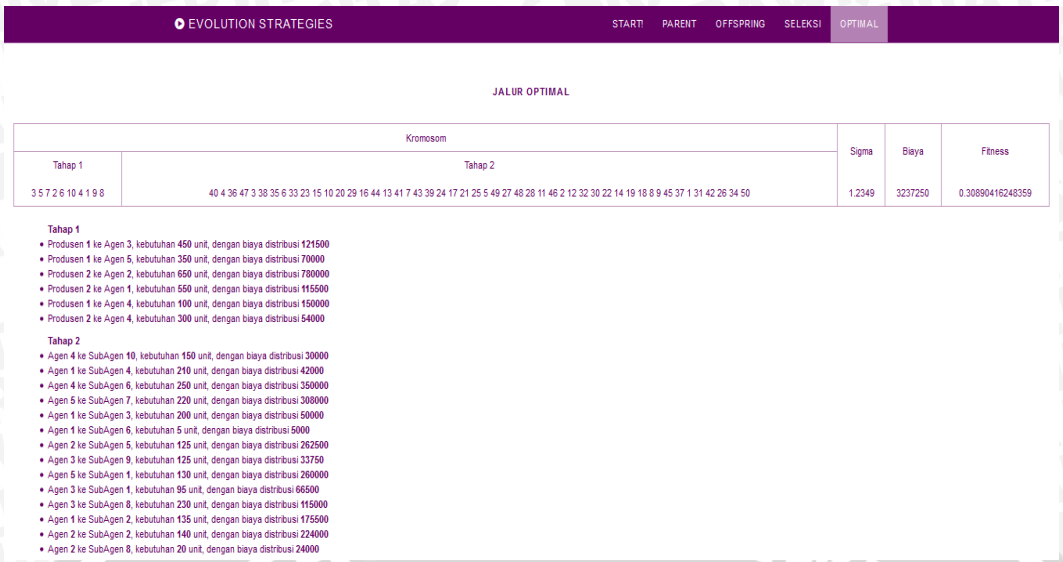


Gambar 5.5 Implementasi Proses Seleksi

Pada tab "seleksi" ditampilkan pengurutan *parent* dan *offspring* secara *descending* berdasarkan nilai *fitness*. Dari hasil pengurutan tersebut dipilih sebanyak *miu* individu yang akan digunakan dalam reproduksi pada generasi berikutnya. Hasil seleksi pada generasi ke-n ditunjukkan pada Gambar 5.6.

Gambar 5.6 Implementasi Hasil Seleksi Populasi Generasi Ke-n

Hasil seleksi generasi ke-n yang ditampilkan adalah individu terbaik dari setiap generasi. Pada halaman proses perhitungan juga menampilkan hasil akhir yang berupa jalur optimal distribusi barang dua tahap. Jalur optimal distribusi dua tahap ditampilkan pada tab "optimal" seperti yang ditunjukkan pada Gambar 5.7.



Gambar 5.7 Implementasi Output Jalur Distribusi Optimal

Output dari sistem ini adalah jalur distribusi barang dua tahap yang optimal. Sistem akan menampilkan kromosom terpilih, nilai *strategy parameter,* total biaya distribusi dua tahap, nilai *fitness*, dan penjabaran jalur distribusinya.