BAB 5 IMPLEMENTASI

Bab ini akan memaparkan hasil dari implementasi sistem sesuai dengan perancangan yang telah dibuat sebelumnya mengenai implementasi optimasi penjadwalan bimbingan skripsi menggunakan algoritme genetika.

5.1 Implementasi Algoritme Genetika

Dalam mengimplementasikan algoritme genetika dalam penjadwalan bimbingan skripsi yang pertama dilakukan yaitu inisialisasi kromosmom. Sistem ini terdiri dari 5 proses utama yaitu proses inisialisasi kromosom, proses *crossover*, proses evaluasi *fitness* dan proses seleksi.

5.1.1 Implementasi Proses Kromosom

Inisialisasi kromosom adalah utama dalam mengimplementasikan algoritme genetika. inisialisasi kromosom bertujuan agar dapat membangkitkan sejumlah kromosom pada setiap populasi. Program ini menghasilkan kromosom yang bernilai bilangan integer secara acak. Implementasi inisialisasi kromosom ditunjukan pada Kode Program 5.1.

```
1   public void inisialisasiKromosom() throws IOException,
2   BiffException {
3         this.jadwalP0 = new
4         int[this.pop_size][this.pnjg_kromosom];
5         for (int i = 0; i < pop_size; i++) {
6             for (int j = 0; j < pnjg_kromosom; j++) {
7             this.jadwalP0[i][j]=(int)(batasb awah+(Math.random() *
8             (batasatas-batasbawah)));
9             }
10         }
11       }
```

Kode Program 5.1 Implementasi Proses Kromosom

Penjelasan Kode Program 5.1 mengenai proses kromosom adalah sebagai berikut:

1. Baris 3-4 adalah proses inisialisasi dari variabel jadwalP0.
2. Baris 5-8 adalah proses inisinialisasi nilai kromosom secara acak pada seluruh individu jadwal bimbingan sesuai dengan *popsize* dan panjang kromsom.

5.1.2 Implementasi Proses *Extended intermediate Crossover*

Proses ini adalah penyilangan antara dua induk yang di pilih secara acak akan menghasilkan generasi yang baru. *Crossover* pada sistem ini menggunakan *extended intermediate crossover.* Kode Program 5.2 adalah kode Implementasi *crossover*.

```
1   public int[][] Crossover(int[][] Individu, int u) {
2     Random random = new Random();
3     int Pop_Size = Individu.length;
4     int Hasil;
5     double[][] alfa = new double[3][panjang_kromosom];
6     int[][] HasilCrossover = new
7     int[Individu.length][panjang_kromosom];
8     double BatasBawahRand = 0, BatasAtasRand = 1;
9     for (int j = 0; j < panjang_kromosom; j++) {
10        alfa[0][j] = (random.nextDouble() * (BatasAtasRand));
11    }
12    BatasBawahRand = 0;
13    BatasAtasRand = Pop_Size - 1;
14    int IndexP1 = (int) (random.nextInt((int) (BatasAtasRand -
15    BatasBawahRand + 1)) + BatasBawahRand);
16    int IndexP2 = (int) (random.nextInt((int) (BatasAtasRand -
17    BatasBawahRand + 1)) + BatasBawahRand);
18    while (IndexP1 == IndexP2) {
19        IndexP2 = (int) (random.nextInt((int) (BatasAtasRand -
20        BatasBawahRand + 1)) + BatasBawahRand);
21    }
22    for (int i = 0; i < 2; i++) {
23        for (int j = 0; j < panjang_kromosom; j++) {
24            HasilCrossover[1][j] = (int) (Individu[IndexP2][j] +
25           (alfa[0][j] * (Individu[IndexP1][j] - Individu[IndexP2][j])));
26            HasilCrossover[0][j] = (int) (Individu[IndexP1][j] +
27            (alfa[0][j] * (Individu[IndexP2][j] - Individu[IndexP1][j])));
28            if (HasilCrossover[0][j] > 250) {
29                HasilCrossover[0][j] = (int)(batasb awah+(Math.random() *
30               (batasatas-batasbawah)));
31            }
32            if (HasilCrossover[1][j] > 250) {
33                HasilCrossover[1][j] = (int)(batasb awah+(Math.random() *
34                (batasatas-batasbawah)));
35            }
36        }
      }
      return HasilCrossover;
```

Kode Program 5.2 Kode Proses *Extended intermediate Crossover*

Penjelasan Kode Program 5.2 mengenai proses *crossover* adalah sebagai berikut:

1. Baris 2-7 inisialisasi parameter untuk method *crossover*().
2. Baris 8-10 menghitung nilai *alfa*.
3. Baris 11-19 menentukan nilai Index*P1* dan Index*P2*, dimana jika nilai sama maka akan di *random* ulang dengan perulangan.
4. Baris 20-35 menentukan nilai hasil dari perhitungan *intermediate extended crossover*.
5. Baris 36 mengembalikan nilai dari Hasil*Crossover*.

5.1.3 Implementasi Proses *Random mutation*

Pada proses ini gen yang terbentuk akan diubah dan menjadi suatu kromosom yang baru dimana akan menghasilkan 1 *offspring*. Metode pada penelitian ini menggunakan *random mutation*. Impementasi mutasi ditunjukkan pada kode program 5.3.

```
1  public int[][] RandomMutasi(int[][] Individu) {
2      Random random = new Random();
3      int panjangChromosome = Individu[0].length;
4      int Pop_Size = Individu.length;
5      int[][] Hasil = new int[1][ panjangChromosome];
6      int byk_anak_crossover = 0;
7      int p = (int) (mr * pop_size);
8      int BatasBawahRand = 1, BatasAtasRand =
9      panjangChromosome;
10     int PosRandom = random.nextInt(BatasAtasRand -
11     BatasBawahRand + 1) + BatasBawahRand;
12     double[][] P1 = new double[1][ panjangChromosome];
13     BatasBawahRand = 0;
14     BatasAtasRand = Pop_Size - 1;
15     int i_old2 = 0;
16     int m_old2 = individuGabunganMhs[i_old2].length;
17     int tempp2 = individuGabunganMhs.length;
18     int out2[] = new int[tempp2];
19     int min[] = new int[panjang_kromosom];
20     int max[] = new int[panjang_kromosom];
21     int byk_anak_mutasi = 0;
22     if (mr * pop_size >= (0.5 + p)) {
23         byk_anak_mutasi = (int) Math.ceil(mr * pop_size);
24     } else {
25         byk_anak_mutasi = (int) (mr * pop_size);
26     }
27     int u = 0;
28     for (int j = 0; j < panjang_kromosom; j++) {
```

```
29          min[j] = 250;
30          for (int i = 0; i < individuGabunganMhs.length -
31          byk_anak_mutasi; i++) {
32              if (individuGabunganMhs[i][j] < min[j]) {
33                  min[j] = individuGabunganMhs[i][j];
34              }
35          }
36      }
37      for (int j = 0; j < panjang_kromosom; j++) {
38          max[j] = 1;
39          for (int i = 0; i < individuGabunganMhs.length -
40          byk_anak_mutasi; i++) {
41              if (individuGabunganMhs[i][j] > max[j]) {
42                  max[j] = individuGabunganMhs[i][j];
43              }
44          }
45      }
46      int IndP1 = random.nextInt(BatasAtasRand -
47      BatasBawahRand + 1) + BatasBawahRand;
48      for (int i = 0; i < panjangChromosome; i++) {
49          P1[0][i] = Individu[IndP1][i];
50          Hasil[0][i] = Individu[IndP1][i];
51      }
52      double[][] alfa = new double[3][panjang_kromosom];
53      int[][] HasilCrossover = new
54      int[Individu.length][panjang_kromosom];
55      for (int j = 0; j < panjang_kromosom; j++) {
56          alfa[0][j] = Math.random();
57      }
58      Hasil[0][ PosRandom- 1] = (int)
59      (P1[0][ PosRandom -1]+((int)
60      (alfa[0][ PosRandom - 1]*(max[PosRandom - 1]
61      - min[PosRandom - 1])))));
62      return Hasil;
63      }
```

Kode Program 5.3 Kode Proses *Random mutation*

Penjelasan Kode Program 5.3 mengenai proses mutasi adalah sebagai berikut:

1. Baris 2-9 inisialisasi parameter.
2. Baris 10 memilih individu *parent* secara acak.
3. Baris 12-21 inisialisasi parameter.
4. Baris 22-36 menghitung nilai terkecil dari setiap populasi.

5. Baris 37-45 menghitung nilai terbesar dari setiap populasi .
6. Baris 46 menentukan nilai index*P1* dengan *random*.
7. Baris 48-50 menempatkan posisi indexP1
8. Baris 52-57 menentukan nilai *alfa*.
9. Baris 58-61 menghitung proses mutasi sesuai dengan rumus *random mutation. .*
10. Baris 62 mengembalikan nilai Hasil.

5.1.4 Implementasi Proses Evaluasi *Fitness*

Pada proses ini merupakan proses untuk menghitung nilai *fitness* dari masing-masing individu dengan melakukan pengecekan *constraint* penjadwalan. Terdapat 4 *constraint* yang harus di lakukan, kemudian akan dihitung nilai *fitness*nya. Implementasi evaluasi *fitness* ditunjukan pada Kode Program 5.4.

```
1   public void evaluasi() {
2     this.fitness = new double[this.individuGabunganMhs.length];
3     this.pinalti = new double[this.individuGabunganMhs.length];
4     this.pinalti1=new double[this.individuGabunganMhs.length];
5     this.pinalti2=new double[this.individuGabunganMhs.length];
6     this.pinalti3=new double[this.individuGabunganMhs.length];
7     this.pinalti4=new double[this.individuGabunganMhs.length];
8     this.pinalti1d=new double[this.individuGabunganMhs.length];
9     this.pinalti2d=new double[this.individuGabunganMhs.length];
10    this.pinalti3d=new double[this.individuGabunganMhs.length];
11    this.pinalti4d=new double[this.individuGabunganMhs.length];
12    this.sum = new double[this.individuGabunganMhs.length];;
13    this.sum1 = new double[this.individuGabunganMhs.length];
14    this.sum2 = new double[this.individuGabunganMhs.length];
15    this.sum3 = new double[this.individuGabunganMhs.length];
16    this.sum4 = new double[this.individuGabunganMhs.length];
17    int a = 0;
18    con1 = 1;
19    con2 = 1;
20    con3 = 1;
21    con4 = 0.5;
22    for (int i = 0; i < individuGabunganMhs.length; i++) {
23        if (a == panjang_kromosom) {
24            a = 0;
25        }
26        cekConstraint1(i, a);
27        cekConstraint2(i, a);
28        cekConstraint3(i, a);
29        cekConstraint4(i, a);
30        a++;
```

```
31      }
32      for (int i = 0; i < individuGabunganMhs.length; i++) {
33          fitness[i] = sum1[i] + sum2[i] + sum3[i] + sum4[i];
34      }
35  }
```

Kode Program 5.4 Implementasi Proses Evaluasi *Fitness*

Penjelasan Kode Program 5.4 mengenai implementasi proses evaluasi *fitness* adalah sebagai berikut:

1. Baris 2-16 inisialisasi *array*.
2. Baris 22-30 pengulangan proses evaluasi sebanyak individuGabunganMhs.
3. Baris 32-33 adalah pengecekan *constraint* 1 sampai 4 apakah terdapat pinalti atau tidak. Jika iya maka pinalti akan bertambah 1 atau 0,5.
4. Baris 32-33 adalah perhitungan nilai *fitness*.

5.1.4.1 Implementasi Proses Method Cek*Constraint* 1

Pada proses ini bertujuan untuk mengecek apakah ada jadwal bimbingan yang ditetapkan ternyata terdapat kesamaan dengan jadwal dosen pembimbing 1. Jika ada maka akan diberikan bobot pinalti. Implementasi Cek *Constraint* 1 ditunjukan pada Kode Program 5.5.

```
1   private void checkConstraint1(int ke, int a) {
2       boolean status;
3       int x = 0, j = 0, c = 0;
4       while (x < panjang_kromosom) {
5           int temp = individuGabunganMhs[ke][(x)];
6           for (j = 0; j < arDosen.length; j++) {
7               if (arJadwal[x].getKodeD1() == arDosen[j].getId()) {
8                   for (int p = 0; p < 23; p++) {
9                       if (arDosen[j].getId() == 1) {
10                          cari3 = DosenPem.get(0).getSlotMK1().get(p);
11                      } else if (arDosen[j].getId() == 2) {
12                          cari3 = DosenPem.get(0).getSlotMK2().get(p);
13                      } else if (arDosen[j].getId() == 3) {
14                          cari3 = DosenPem.get(0).getSlotMK3().get(p);
15                      } else if (arDosen[j].getId() == 4) {
16                          cari3 = DosenPem.get(0).getSlotMK4().get(p);
17                      } else if (arDosen[j].getId() == 5) {
18                          cari3 = DosenPem.get(0).getSlotMK5().get(p);
19                      } else if (arDosen[j].getId() == 6) {
20                          cari3 = DosenPem.get(0).getSlotMK6().get(p);
21                      } else if (arDosen[j].getId() == 7) {
22                          cari3 = DosenPem.get(0).getSlotMK7().get(p);
23                      } else if (arDosen[j].getId() == 8) {
```

```
                        cari3 = DosenPem.get(0).getSlotMK8().get(p);
                    } else if (arDosen[j].getId() == 9) {
                        cari3 = DosenPem.get(0).getSlotMK9().get(p);
                    } else if (arDosen[j].getId() == 10) {
                        cari3 = DosenPem.get(0).getSlotMK10().get(p);
                    } else if (arDosen[j].getId() == 11) {
                        cari3 = DosenPem.get(0).getSlotMK11().get(p);
                    } else if (arDosen[j].getId() == 12) {
                        cari3 = DosenPem.get(0).getSlotMK12().get(p);
                    } else if (arDosen[j].getId() == 13) {
                        cari3 = DosenPem.get(0).getSlotMK13().get(p);
                    } else if (arDosen[j].getId() == 14) {
                        cari3 = DosenPem.get(0).getSlotMK15().get(p);
                    } else if (arDosen[j].getId() == 15) {
                        cari3 = DosenPem.get(0).getSlotMK15().get(p);
                    } else if (arDosen[j].getId() == 16) {
                        cari3 = DosenPem.get(0).getSlotMK16().get(p);
                    } else if (arDosen[j].getId() == 17) {
                        cari3 = DosenPem.get(0).getSlotMK17().get(p);
                    } else if (arDosen[j].getId() == 18) {
                        cari3 = DosenPem.get(0).getSlotMK18().get(p);
                    } else if (arDosen[j].getId() == 19) {
                        cari3 = DosenPem.get(0).getSlotMK19().get(p);
                    } else if (arDosen[j].getId() == 20) {
                        cari3 = DosenPem.get(0).getSlotMK20().get(p);
                    } else if (arDosen[j].getId() == 21) {
                        cari3 = DosenPem.get(0).getSlotMK21().get(p);
                    } else if (arDosen[j].getId() == 22) {
                        cari3 = DosenPem.get(0).getSlotMK22().get(p);
                    } else if (arDosen[j].getId() == 23) {
                        cari3 = DosenPem.get(0).getSlotMK23().get(p);
                    } else if (arDosen[j].getId() == 24) {
                        cari3 = DosenPem.get(0).getSlotMK24().get(p);
                    } else if (arDosen[j].getId() == 25) {
                        cari3 = DosenPem.get(0).getSlotMK25().get(p);
                    } else if (arDosen[j].getId() == 26) {
                        cari3 = DosenPem.get(0).getSlotMK26().get(p);
                    } else if (arDosen[j].getId() == 27) {
                        cari3 = DosenPem.get(0).getSlotMK27().get(p);
                    } else if (arDosen[j].getId() == 28) {
                        cari3 = DosenPem.get(0).getSlotMK28().get(p);
                    } else if (arDosen[j].getId() == 29) {
                        cari3 = DosenPem.get(0).getSlotMK29().get(p);
                    } else if (arDosen[j].getId() == 30) {
```

```
68                              cari3 = DosenPem.get(0).getSlotMK30().get(p);
69                          } else if (arDosen[j].getId() == 31) {
70                              cari3 = DosenPem.get(0).getSlotMK31().get(p);
71                          } else if (arDosen[j].getId() == 32) {
72                              cari3 = DosenPem.get(0).getSlotMK32().get(p);
73                          } else if (arDosen[j].getId() == 33) {
74                              cari3 = DosenPem.get(0).getSlotMK33().get(p);
75                          } else if (arDosen[j].getId() == 34) {
76                              cari3 = DosenPem.get(0).getSlotMK34().get(p);
77                          } else if (arDosen[j].getId() == 35) {
78                              cari3 = DosenPem.get(0).getSlotMK35().get(p);
79                          } else if (arDosen[j].getId() == 36) {
80                              cari3 = DosenPem.get(0).getSlotMK36().get(p);
81                          } else if (arDosen[j].getId() == 37) {
82                              cari3 = DosenPem.get(0).getSlotMK37().get(p);
83                          } else if (arDosen[j].getId() == 38) {
84                              cari3 = DosenPem.get(0).getSlotMK38().get(p);
85                          } else if (arDosen[j].getId() == 39) {
86                              cari3 = DosenPem.get(0).getSlotMK39().get(p);
87                          } else if (arDosen[j].getId() == 40) {
88                              cari3 = DosenPem.get(0).getSlotMK40().get(p);
89                          } else if (arDosen[j].getId() == 41) {
90                              cari3 = DosenPem.get(0).getSlotMK41().get(p);
91                          } else if (arDosen[j].getId() == 42) {
92                              cari3 = DosenPem.get(0).getSlotMK42().get(p);
93                          } else if (arDosen[j].getId() == 43) {
94                              cari3 = DosenPem.get(0).getSlotMK43().get(p);
95                          } else if (arDosen[j].getId() == 44) {
96                              cari3 = DosenPem.get(0).getSlotMK45().get(p);
97                          } else if (arDosen[j].getId() == 45) {
98                              cari3 = DosenPem.get(0).getSlotMK45().get(p);
99                          } else if (arDosen[j].getId() == 46) {
100                             cari3 = DosenPem.get(0).getSlotMK46().get(p);
101                         } else if (arDosen[j].getId() == 47) {
102                             cari3 = DosenPem.get(0).getSlotMK47().get(p);
103                         } else if (arDosen[j].getId() == 48) {
104                             cari3 = DosenPem.get(0).getSlotMK48().get(p);
105                         } else if (arDosen[j].getId() == 49) {
106                             cari3 = DosenPem.get(0).getSlotMK49().get(p);
107                         } else if (arDosen[j].getId() == 50) {
108                             cari3 = DosenPem.get(0).getSlotMK50().get(p);
109                         } else if (arDosen[j].getId() == 51) {
110                             cari3 = DosenPem.get(0).getSlotMK51().get(p);
111                         } else if (arDosen[j].getId() == 52) {
```

```
112                             cari3 = DosenPem.get(0).getSlotMK52().get(p);
113                         } else if (arDosen[j].getId() == 53) {
114                             cari3 = DosenPem.get(0).getSlotMK53().get(p);
115                         } else if (arDosen[j].getId() == 54) {
116                             cari3 = DosenPem.get(0).getSlotMK54().get(p);
117                         } else if (arDosen[j].getId() == 55) {
118                             cari3 = DosenPem.get(0).getSlotMK55().get(p);
119                         } else if (arDosen[j].getId() == 56) {
120                             cari3 = DosenPem.get(0).getSlotMK56().get(p);
121                         } else if (arDosen[j].getId() == 57) {
122                             cari3 = DosenPem.get(0).getSlotMK57().get(p);
123                         } else if (arDosen[j].getId() == 58) {
124                             cari3 = DosenPem.get(0).getSlotMK58().get(p);
125                         } else if (arDosen[j].getId() == 59) {
126                             cari3 = DosenPem.get(0).getSlotMK59().get(p);
127                         } else if (arDosen[j].getId() == 53) {
128                             cari3 = DosenPem.get(0).getSlotMK53().get(p);
129                         } else if (arDosen[j].getId() == 54) {
130                             cari3 = DosenPem.get(0).getSlotMK54().get(p);
131                         } else if (arDosen[j].getId() == 55) {
132                             cari3 = DosenPem.get(0).getSlotMK55().get(p);
133                         } else if (arDosen[j].getId() == 56) {
134                             cari3 = DosenPem.get(0).getSlotMK56().get(p);
135                         } else if (arDosen[j].getId() == 57) {
136                             cari3 = DosenPem.get(0).getSlotMK57().get(p);
137                         } else if (arDosen[j].getId() == 58) {
138                             cari3 = DosenPem.get(0).getSlotMK58().get(p);
139                         } else if (arDosen[j].getId() == 59) {
140                             cari3 = DosenPem.get(0).getSlotMK59().get(p);
141                         } else if (arDosen[j].getId() == 60) {
142                             cari3 = DosenPem.get(0).getSlotMK60().get(p);
143                         } else if (arDosen[j].getId() == 61) {
144                             cari3 = DosenPem.get(0).getSlotMK61().get(p);
145                         } else if (arDosen[j].getId() == 62) {
146                             cari3 = DosenPem.get(0).getSlotMK62().get(p);
147                         } else if (arDosen[j].getId() == 63) {
148                             cari3 = DosenPem.get(0).getSlotMK63().get(p);
149                         } else if (arDosen[j].getId() == 64) {
150                             cari3 = DosenPem.get(0).getSlotMK64().get(p);
151                         } else if (arDosen[j].getId() == 65) {
152                             cari3 = DosenPem.get(0).getSlotMK65().get(p);
153                         } else if (arDosen[j].getId() == 66) {
154                             cari3 = DosenPem.get(0).getSlotMK66().get(p);
155                         } else if (arDosen[j].getId() == 67) {
```

```
156                             cari3 = DosenPem.get(0).getSlotMK67().get(p);
157                         }
158                         if (temp == cari3) {
159                             status = true;
160                             pinalti1[ke] += 1;
161                             pinalti1d[ke] += 1;
162                         }
163                     }
164                 }
165             }
166             x++;
167         }
168         pinalti1a += pinalti1d[ke];
169         sum1[ke] = (con1 / (pinalti1[ke] + 1));
170     }
171
```

Kode Program 5.5 Implementasi Proses Method Cek*Constraint* 1

Penjelasan untuk Kode Program 5.5 mengenai proses method cek*constraint* 1 dijelaskan sebagai berikut:

1. Baris 3-4 inisialisasi parameter.
2. Baris 5-158 pengulangan sebanyak *x (*panjang kromosom), *j* (panjang jadwal dosen) dan *p* (jumlah kolom dari jadwal dosen).
3. Baris 7 jika nilai dari jadwal bimbingan sama dengan jadwal dari dosen pembimbing 1.
4. Baris 11-158 penempatan nilai cari 3 dari kode dosen yang memiliki nilai yang sama dengan nilai 1-67.
5. Baris 160-162 jika nilai temp sama dengan nilai cari3 maka terjadi pinalti1[ke] dan nilai pinalti akan bertambah.
6. Baris 168-169 perhitungan nilai sum1[ke].

5.1.4.2 Implementasi Proses Method Cek*Constraint* 2

Pada proses ini bertujuan untuk mengecek apakah ada jadwal bimbingan yang ditetapkan ternyata terdapat kesamaan dengan jadwal dosen pembimbing 2. Jika ada maka akan diberikan bobot pinalti. Implementasi Cek *Constraint* 2 ditunjukan pada Kode Program 5.6.

```
1   private void check*Constraint2*(int ke, int a) {
2       boolean status;
3       int x = 0, j = 0, c = 0;
4       int o = pop_size * panjang_kromosom;
5       while (x < panjang_kromosom) {
6           int temp = individuGabunganMhs[ke][(x)];
7           for (j = 0; j < arDosen.length; j++) {
```

```
8                 if (arJadwal[x].getKodeD2() == arDosen[j].getId()) {
9                     for (int p = 0; p < 23; p++) {
10                        if (arDosen[j].getId() == 1) {
11                            cari3 = DosenPem.get(0).getSlotMK1().get(p);
12                        } else if (arDosen[j].getId() == 2) {
13                            cari3 = DosenPem.get(0).getSlotMK2().get(p);
14                        } else if (arDosen[j].getId() == 3) {
15                            cari3 = DosenPem.get(0).getSlotMK3().get(p);
16                        } else if (arDosen[j].getId() == 4) {
17                            cari3 = DosenPem.get(0).getSlotMK4().get(p);
18                        } else if (arDosen[j].getId() == 5) {
19                            cari3 = DosenPem.get(0).getSlotMK5().get(p);
20                        } else if (arDosen[j].getId() == 6) {
21                            cari3 = DosenPem.get(0).getSlotMK6().get(p);
22                        } else if (arDosen[j].getId() == 7) {
23                            cari3 = DosenPem.get(0).getSlotMK7().get(p);
24                        } else if (arDosen[j].getId() == 8) {
25                            cari3 = DosenPem.get(0).getSlotMK8().get(p);
26                        } else if (arDosen[j].getId() == 9) {
27                            cari3 = DosenPem.get(0).getSlotMK9().get(p);
28                        } else if (arDosen[j].getId() == 10) {
29                            cari3 = DosenPem.get(0).getSlotMK10().get(p);
30                        } else if (arDosen[j].getId() == 11) {
31                            cari3 = DosenPem.get(0).getSlotMK11().get(p);
32                        } else if (arDosen[j].getId() == 12) {
33                            cari3 = DosenPem.get(0).getSlotMK12().get(p);
34                        } else if (arDosen[j].getId() == 13) {
35                            cari3 = DosenPem.get(0).getSlotMK13().get(p);
36                        } else if (arDosen[j].getId() == 14) {
37                            cari3 = DosenPem.get(0).getSlotMK15().get(p);
38                        } else if (arDosen[j].getId() == 15) {
39                            cari3 = DosenPem.get(0).getSlotMK15().get(p);
40                        } else if (arDosen[j].getId() == 16) {
41                            cari3 = DosenPem.get(0).getSlotMK16().get(p);
42                        } else if (arDosen[j].getId() == 17) {
43                            cari3 = DosenPem.get(0).getSlotMK17().get(p);
44                        } else if (arDosen[j].getId() == 18) {
45                            cari3 = DosenPem.get(0).getSlotMK18().get(p);
46                        } else if (arDosen[j].getId() == 19) {
47                            cari3 = DosenPem.get(0).getSlotMK19().get(p);
48                        } else if (arDosen[j].getId() == 20) {
49                            cari3 = DosenPem.get(0).getSlotMK20().get(p);
50                        } else if (arDosen[j].getId() == 21) {
51                            cari3 = DosenPem.get(0).getSlotMK21().get(p);
```

```
52                          } else if (arDosen[j].getId() == 22) {
53                              cari3 = DosenPem.get(0).getSlotMK22().get(p);
54                          } else if (arDosen[j].getId() == 23) {
55                              cari3 = DosenPem.get(0).getSlotMK23().get(p);
56                          } else if (arDosen[j].getId() == 24) {
57                              cari3 = DosenPem.get(0).getSlotMK24().get(p);
58                          } else if (arDosen[j].getId() == 25) {
59                              cari3 = DosenPem.get(0).getSlotMK25().get(p);
60                          } else if (arDosen[j].getId() == 26) {
61                              cari3 = DosenPem.get(0).getSlotMK26().get(p);
62                          } else if (arDosen[j].getId() == 27) {
63                              cari3 = DosenPem.get(0).getSlotMK27().get(p);
64                          } else if (arDosen[j].getId() == 28) {
65                              cari3 = DosenPem.get(0).getSlotMK28().get(p);
66                          } else if (arDosen[j].getId() == 29) {
67                              cari3 = DosenPem.get(0).getSlotMK29().get(p);
68                          } else if (arDosen[j].getId() == 30) {
69                              cari3 = DosenPem.get(0).getSlotMK30().get(p);
70                          } else if (arDosen[j].getId() == 31) {
71                              cari3 = DosenPem.get(0).getSlotMK31().get(p);
72                          } else if (arDosen[j].getId() == 32) {
73                              cari3 = DosenPem.get(0).getSlotMK32().get(p);
74                          } else if (arDosen[j].getId() == 33) {
75                              cari3 = DosenPem.get(0).getSlotMK33().get(p);
76                          } else if (arDosen[j].getId() == 34) {
77                              cari3 = DosenPem.get(0).getSlotMK34().get(p);
78                          } else if (arDosen[j].getId() == 35) {
79                              cari3 = DosenPem.get(0).getSlotMK35().get(p);
80                          } else if (arDosen[j].getId() == 36) {
81                              cari3 = DosenPem.get(0).getSlotMK36().get(p);
82                          } else if (arDosen[j].getId() == 37) {
83                              cari3 = DosenPem.get(0).getSlotMK37().get(p);
84                          } else if (arDosen[j].getId() == 38) {
85                              cari3 = DosenPem.get(0).getSlotMK38().get(p);
86                          } else if (arDosen[j].getId() == 39) {
87                              cari3 = DosenPem.get(0).getSlotMK39().get(p);
88                          } else if (arDosen[j].getId() == 40) {
89                              cari3 = DosenPem.get(0).getSlotMK40().get(p);
90                          } else if (arDosen[j].getId() == 41) {
91                              cari3 = DosenPem.get(0).getSlotMK41().get(p);
92                          } else if (arDosen[j].getId() == 42) {
93                              cari3 = DosenPem.get(0).getSlotMK42().get(p);
94                          } else if (arDosen[j].getId() == 43) {
95                              cari3 = DosenPem.get(0).getSlotMK43().get(p);
```

```
96                          } else if (arDosen[j].getId() == 44) {
97                              cari3 = DosenPem.get(0).getSlotMK45().get(p);
98                          } else if (arDosen[j].getId() == 45) {
99                              cari3 = DosenPem.get(0).getSlotMK45().get(p);
100                         } else if (arDosen[j].getId() == 46) {
101                             cari3 = DosenPem.get(0).getSlotMK46().get(p);
102                         } else if (arDosen[j].getId() == 47) {
103                             cari3 = DosenPem.get(0).getSlotMK47().get(p);
104                         } else if (arDosen[j].getId() == 48) {
105                             cari3 = DosenPem.get(0).getSlotMK48().get(p);
106                         } else if (arDosen[j].getId() == 49) {
107                             cari3 = DosenPem.get(0).getSlotMK49().get(p);
108                         } else if (arDosen[j].getId() == 50) {
109                             cari3 = DosenPem.get(0).getSlotMK50().get(p);
110                         } else if (arDosen[j].getId() == 51) {
111                             cari3 = DosenPem.get(0).getSlotMK51().get(p);
112                         } else if (arDosen[j].getId() == 52) {
113                             cari3 = DosenPem.get(0).getSlotMK52().get(p);
114                         } else if (arDosen[j].getId() == 53) {
115                             cari3 = DosenPem.get(0).getSlotMK53().get(p);
116                         } else if (arDosen[j].getId() == 54) {
117                             cari3 = DosenPem.get(0).getSlotMK54().get(p);
118                         } else if (arDosen[j].getId() == 55) {
119                             cari3 = DosenPem.get(0).getSlotMK55().get(p);
120                         } else if (arDosen[j].getId() == 56) {
121                             cari3 = DosenPem.get(0).getSlotMK56().get(p);
122                         } else if (arDosen[j].getId() == 57) {
123                             cari3 = DosenPem.get(0).getSlotMK57().get(p);
124                         } else if (arDosen[j].getId() == 58) {
125                             cari3 = DosenPem.get(0).getSlotMK58().get(p);
126                         } else if (arDosen[j].getId() == 59) {
127                             cari3 = DosenPem.get(0).getSlotMK59().get(p);
128                         } else if (arDosen[j].getId() == 53) {
129                             cari3 = DosenPem.get(0).getSlotMK53().get(p);
130                         } else if (arDosen[j].getId() == 54) {
131                             cari3 = DosenPem.get(0).getSlotMK54().get(p);
132                         } else if (arDosen[j].getId() == 55) {
133                             cari3 = DosenPem.get(0).getSlotMK55().get(p);
134                         } else if (arDosen[j].getId() == 56) {
135                             cari3 = DosenPem.get(0).getSlotMK56().get(p);
136                         } else if (arDosen[j].getId() == 57) {
137                             cari3 = DosenPem.get(0).getSlotMK57().get(p);
138                         } else if (arDosen[j].getId() == 58) {
139                             cari3 = DosenPem.get(0).getSlotMK58().get(p);
```

```
140                         } else if (arDosen[j].getId() == 59) {
141                             cari3 = DosenPem.get(0).getSlotMK59().get(p);
142                         } else if (arDosen[j].getId() == 60) {
143                             cari3 = DosenPem.get(0).getSlotMK60().get(p);
144                         } else if (arDosen[j].getId() == 61) {
145                             cari3 = DosenPem.get(0).getSlotMK61().get(p);
146                         } else if (arDosen[j].getId() == 62) {
147                             cari3 = DosenPem.get(0).getSlotMK62().get(p);
148                         } else if (arDosen[j].getId() == 63) {
149                             cari3 = DosenPem.get(0).getSlotMK63().get(p);
150                         } else if (arDosen[j].getId() == 64) {
151                             cari3 = DosenPem.get(0).getSlotMK64().get(p);
152                         } else if (arDosen[j].getId() == 65) {
153                             cari3 = DosenPem.get(0).getSlotMK65().get(p);
154                         } else if (arDosen[j].getId() == 66) {
155                             cari3 = DosenPem.get(0).getSlotMK66().get(p);
156                         } else if (arDosen[j].getId() == 67) {
157                             cari3 = DosenPem.get(0).getSlotMK67().get(p);
158                         }
159                         if (temp == cari3) {
160                             status = true;
161                             pinalti2[ke] += 1;
162                             pinalti2d[ke] += 1;
163                         }
164                     }
165                 }
166             }
167             x++;
168         }
169         pinalti2a += pinalti2d[ke];
170         sum2[ke] = (con2 / (pinalti2[ke] + 1));
171     }
172
```

Kode Program 5.6 Implementasi Proses Method Cek*Constraint* 2

Penjelasan untuk Kode Program 5.6 dalam proses method cek*constraint* 2 dijelaskan sebagai berikut:

1. Baris 3-4 inisialisasi parameter.
2. Baris 5-158 pengulangan sebanyak *x(*panjang kromosom), *j*(panjang jadwal dosen) dan *p*(jumlah kolom dari jadwal dosen).
3. Baris 8 jika nilai dari jadwal bimbingan sama dengan jadwal dari dosen pembimbing 1.
4. Baris 11-158 penempatan nilai cari 3 dari kode dosen yang memiliki nilai yang sama dengan nilai 1-67.

5. Baris 160-162 jika nilai temp sama dengan nilai cari3 maka terjadi pinalti2[ke] dan nilai pinalti akan bertambah.
6. Baris 168-170 perhitungan nilai sum2[ke].

5.1.4.3 Implementasi Proses Method Cek*Constraint* 3

Pada proses ini bertujuan untuk mengecek apakah ada jadwal bimbingan yang ditetapkan ternyata terdapat kesamaan dengan jadwal perkuliahan mahasiswa. Jika ada maka akan diberikan bobot pinalti. Implementasi Method Cek*Constraint* 3 ditunjukan pada Kode Program 5.7.

```
1   private void cekConstraint3(int ke, int a) {
2       this.cari4 = new int[500][12];
3       boolean status;
4         pinalti[ke] = 0;
5         fitness[ke] = 0;
6       int x = 0, j = 0, c = 0;
7         int o = pop_size * panjang_kromosom;
8       while (x < panjang_kromosom) {
9           int temp = individuGabunganMhs[ke][(x)];
10          j = x;
11          for (int p = 0; p < 12; p++) {
12              cari4[j][p] = temp4cek[j][p];
13              if (temp == cari4[j][p]) {
14                  status = true;
15                  pinalti3[ke] += 1;
16                  pinalti3d[ke] += 1;
17              }
18          }
19          x++;
20      }
21      pinalti3a += pinalti3d[ke];
22      sum3[ke] += (con3 / (pinalti3[ke] + 1));
23  }
```

Kode Program 5.7 Implementasi Proses Method Cek*Constraint* 3

Penjelasan untuk Kode Program 5.7 dalam proses method cek*constraint* 3 dijelaskan sebagai berikut:

1. Baris 2-7 inisialisasi parameter.
2. Baris 8-20 pengulangan sebanyak *x*(panjang kromosom), *j*(panjang jadwal dosen) dan *p*(jumlah kolom dari jadwal dosen).
3. Baris 9 deklarasi nilai temp.
4. Baris 12 deklarasi *array* cari4.
5. Baris 13-16 jika nilai temp sama dengan nilai cari4 maka terjadi pinalti[ke] dan nilai pinalti akan bertambah.

6. Baris 21-22 perhitungan nilai sum3[ke].

5.1.4.4 Implementasi Proses Method Cek*Constraint* 4

Pada proses ini bertujuan untuk mengecek apakah ada jadwal bimbingan yang ditetapkan ternyata terdapat kesamaan dengan jadwal perkuliahan mahasiswa. Jika ada maka akan diberikan bobot pinalti. Implementasi Method Cek*Constraint* 3 ditunjukan pada Kode Program 5.8.

```
1   private void cekConstraint4(int ke, int a) {
2       boolean status;
3       int x = 0, j = 0, c = 0, l = 0, k = 1, m = 0;
4       int v = 0;
5       for (c = 0; c < panjang_kromosom; c++) {
6           while (x < panjang_kromosom) {
7               int temp2 = individuGabunganMhs[ke][(x)];
8               int temp = individuGabunganMhs[ke][c];
9               if (temp == temp2) {
10                  if (x != c) {
11                      if(arJadwal[c].getKodeD1()==
12                      arJadwal[x].getKodeD1()||arJadwal[c].getKodeD2()
13                      == arJadwal[x].getKodeD2()) {
14                          pinalti4[ke] += 0.5;
15                      }
16                  }
17              }
18              x++;
19          }
20          x = 0;
21      }
22      sum4[ke] += (con4 / (pinalti4[ke] + 1));
23
```

Kode Program 5.8 Implementasi Proses Method Cek*Constraint* 4

Penjelasan Kode Program 5.8 mengenai proses method cek*constraint* 4 dijelaskan sebagai berikut:

1. Baris 2-4 inisialisasi parameter.
2. Baris 6-21 pengulangan sebanyak variabel *m*(panjang kromosom), *c*(individuGabunganMhs) dan *x*(jumlah kolom dari jadwal dosen).
3. Baris 7-8 pengisian nilai temp dan temp*2*.
4. Baris 9 jika nilai temp dan temp*2* ternyata sama yang mana pencarian jadwal bimbingan ternyata ada yang sama.
5. Baris 10 dimana nilai ke dan nilai variabel *c* tidak boleh sama yang berposisi *array* yang dicari dengan mencari tidak boleh sama.

6. Baris 11-14 jika jadwal bimbingan yang dicari pada kode dosen 1 dan 2 ternyata sama dengan jadwal bimbingan yang mencari. Jika sama maka akan terjadi pinalti.
7. Baris 22 perhitungan nilai sum4.

5.1.5 Implementasi Proses Seleksi

Implementasi seleksi merupakan pengurutan seluruh individu dari nilai *fitness* yang didapat dari yang terbesar ke terkecil, lalu hanya ambil sebesar *popsize* berdasarkan individu yang memiliki nilai *fitness* terbesar untuk dijadikan dilanjutnya ke generasi selanjutnya. Implementasi Seleksi ditunjukan pada Kode Program 5.9.

```
1   public BestGeneration seleksi(int Ke) {
2     double[] fitnessSorted = new double[fitness.length];
3     int[][] jadwalP02 = new int[pop_size][panjang_kromosom];
4     int[] nomorIndexSorted = new int[fitness.length];
5     double tempor;
6     int indTempor = 0;
7     int y = 0;
8     for (int i = 0; i < fitness.length; i++) {
9         fitnessSorted[i] = fitness[i];
10        nomorIndexSorted[i] = i;
11    }
12    for (int i = 0; i < fitness.length; i++) {
13        for (int j = (int) (i + 1); j < fitness.length; j++) {
14            if (fitnessSorted[i] < fitnessSorted[j]) {
15                tempor = fitnessSorted[j];
16                indTempor = nomorIndexSorted[j];
17                fitnessSorted[j] = fitnessSorted[i];
18                nomorIndexSorted[j] = nomorIndexSorted[i];
19                fitnessSorted[i] = tempor;
20                nomorIndexSorted[i] = indTempor;
21            }
22        }
23    }
24    int a = 0;
25    for (int i = 0; i < pop_size; i++) {
26        for (int j = 0; j < panjang_kromosom; j++) {
27            jadwalP0[i][j] = individuGabunganMhs[indexSorted[i]][j];
28        }
29    }
30    for (int i = 0; i < 1; i++) {
31        for (int j = 0; j < panjang_kromosom; j++) {
32            terter = indexSorted[i];
33        }
```

```
34      }
35      arBest[Ke] = new BestGeneration(Ke, fitnessSorted[0],
36      jadwalP0[0]);        return arBest[Ke];
37  }
```

Kode Program 5.9 Implementasi Proses Seleksi

Penjelasan Kode Program 5.9 mengenai proses seleksi dijelaskan sebagai berikut:

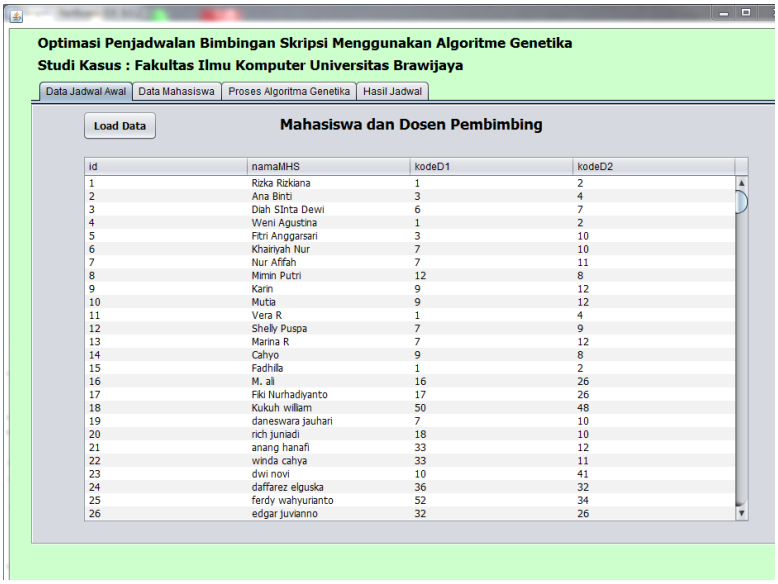
1. Baris 2-7 inisialisasi parameter.
2. Baris 8-11 pengulangan sebanyak panjang *fitness* yang akan digunakan untuk mengisi nilai *array* dari variabel *fitness*Sorted awal.
3. Baris 12-23 pengulangan sebanyak panjang *fitness* yang akan digunakan untuk proses pengurutan nilai dari tertinggi ke terendah, indeks individu juga disusun berdasarkan urutan nilai *fitness*.
4. Baris 24-27 pengulangan sebanyak *popsize* untuk mengambil individu dengan nilai *fitness* tertinggi.
5. Baris 30-33 untuk menempatkan nilai variabel terter pada *popsize* yang memiliki nilai *fitness* tertinggi.
6. Baris 38 mengembalikan nilai arBest.

5.2 Implementasi Antarmuka

Implementasi antarmuka penjadwalan bimbingan terdiri dari 4 halaman yang terdir dari jadwal data, data mahasiswa dan dosen, proses algoritme genetika dan hasil jadwal.

5.2.1 Halaman Data Jadwal Awal

Implementasi halaman data jadwal adalah halaman pertama dari sistem. Halaman ini berfungsi untuk memuat data jadwal bimbingan dari *database* sebagai *input* sistem. Implementasi halaman data jadwal ditunjukan pada Gambar 5.1.

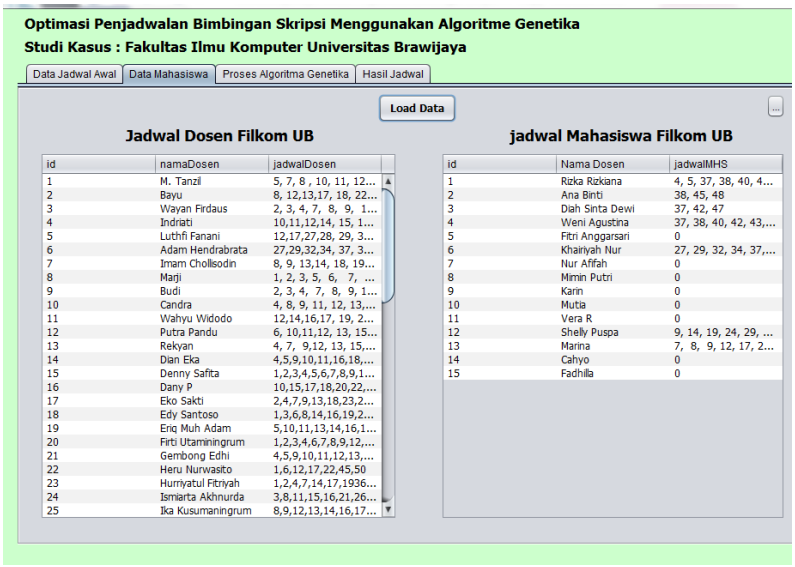


Gambar 5.1 Halaman Data Jadwal Awal

Pada Gambar 5.1 merupakan tampilan dari halaman data jadwal awal bimbingan berdasarkan nama mahasiswa, dosen pembimbing pertama dan dosen pembimbing kedua. Proses data jadwal diawali dengan menekan tombol *load* data untuk bisa mengakses data tersebut dari *database* MYSQL. Setelah pengaksesan berhasil data akan disimpan dalam bentuk *array* yang kemudian sistem akan menampilkan hasilnya kedalam bentuk tabel.

5.2.2 Halaman Data Mahasiswa

Halaman data mahasiswa ini berfungsi untuk menampilkan data dosen dan mahasiswa kedalam tabel sesuai dengan *database* MYSQL yang telah di buat yang ada di *localhost*. Halaman data mahasiswa dan dosen ditunjukan pada Gambar 5.2.

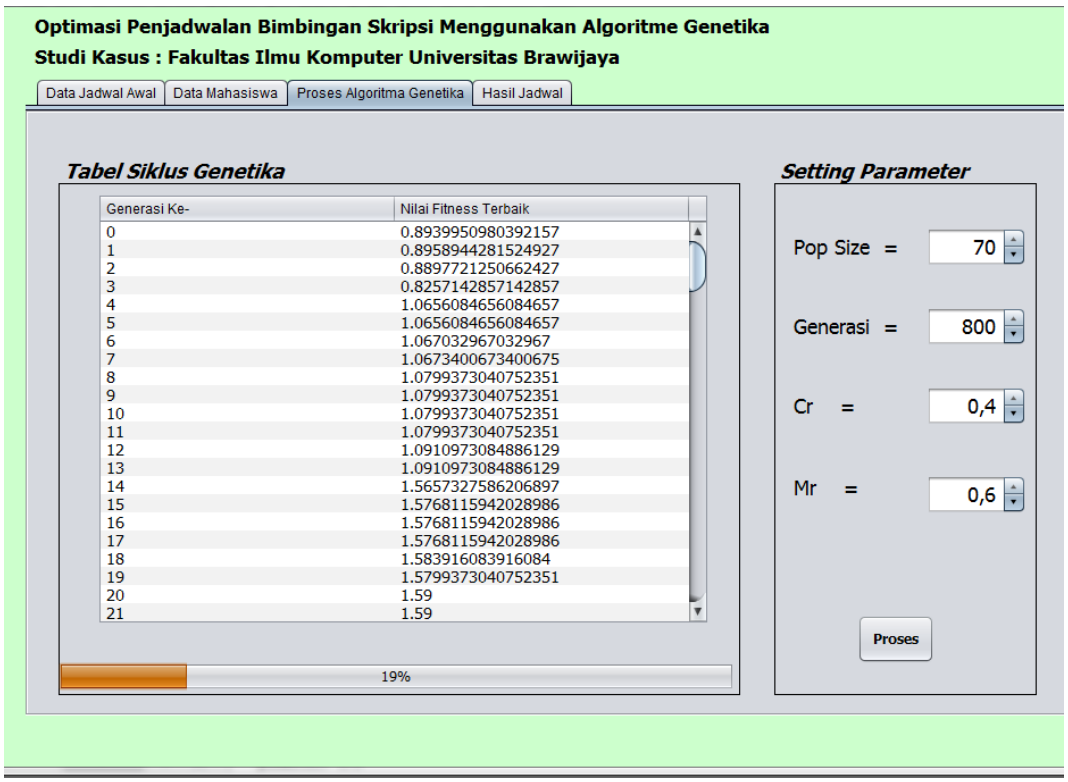


Gambar 5.2 Halaman Data Mahasiswa

Gambar 5.2 hanya menampilkan hasil implementasi dari halaman data mahasiswa dan dosen. Pada proses pengguna dapat menekan tombol *load* data yang berfungsi untuk bisa mengakses data mahasiswa dan dosen di *database* MYSQL dan kemudian menampilkannya kehalaman tersebut kedalam bentuk tabel. Sebelum proses menampilkan data ke tabel, data disimpan dahulu kedalam *array* dua dimensi.

5.2.3 Halaman Proses Algoritme Genetika

Pada halaman pengguna akan diminta untuk memasukan beberapa nilai parameter algoritme genetika seperti *popsize*, *cr*,*mr*, dan jumlah generasi yang akan ingin dihasilkan lalu menampilkan hasilnya berupa nilai dari *fitness* terbaik pada setiap generasi.

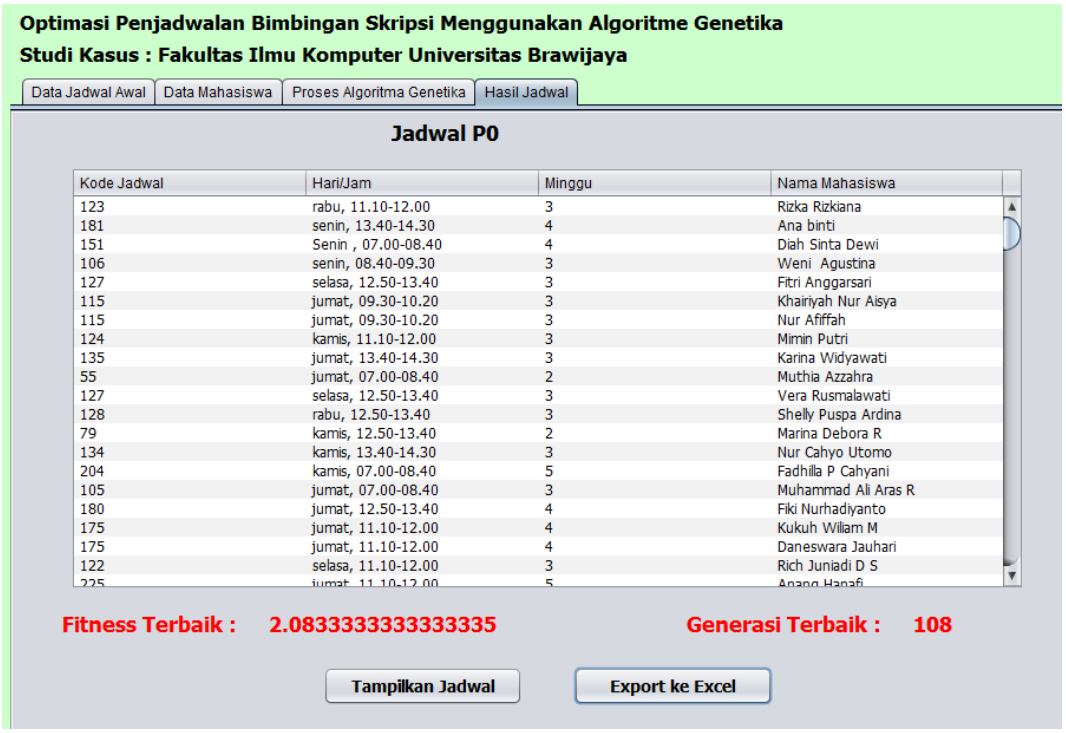


Gambar 5.3 Halaman Proses Algoritme Genetika

Pada halaman ini berdasarkan Gambar 5.3 pengguna harus memasukan beberapa parameter yang digunakan oleh algoritme genetika. *Input*an tersebut akan disajikan dalam bentuk *spinner*. Batas bawah parameter bernilai 1. Sedangkan batas bawah *cr* dan *mr* adalah 0,1 dan batas atas adalah 0,9. Jika *button* proses di tekan maka akan menampilkan nilai *fitness* terbaik dari setiap generasi di dalam Tabel.

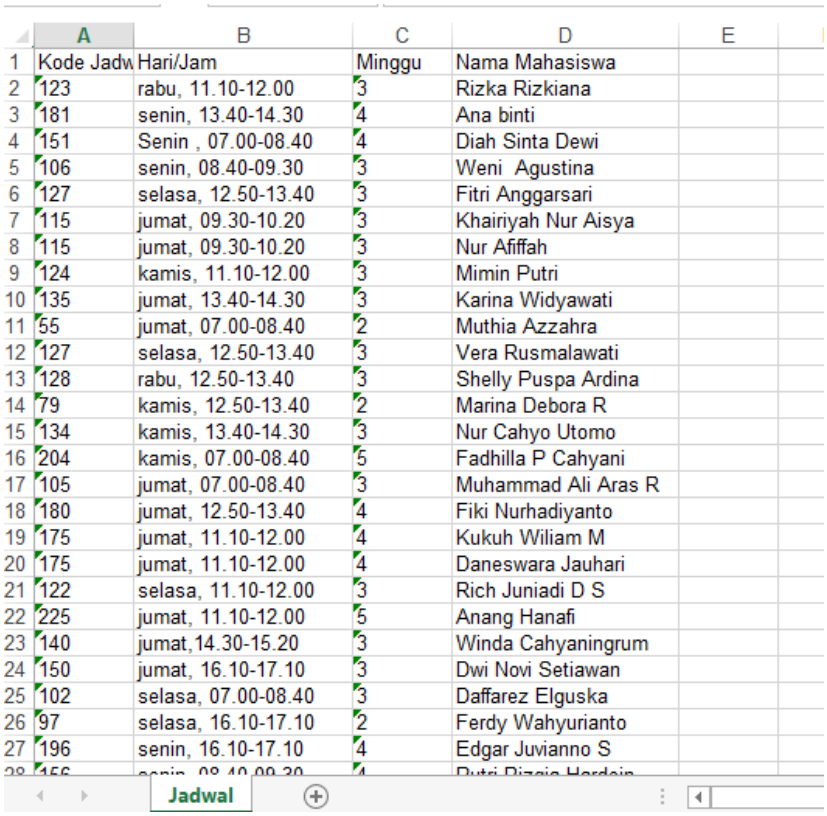
5.2.4 Halaman Hasil Jadwal P0

Individu yang terbaik akan di ubah menjadi jadwal bimbingan berdasarkan kode bimbingan yang diperoleh oleh sistem. Implementasi halaman hasil jadwal ditunjukan pada Gambar 5.4 dan hasil yang sudah di-*export* ke format .xls ditunjukan pada Gambar 5.5.



Gambar 5.4 Halaman Hasil Jadwal

Pada Gambar 5.4 diatas yaitu halaman proses algoritme genetika, pengguna dapat menekan tombol tampilkan jadwal untuk menampilkan jadwal bimbingan yang sudah di seleksi menggunakan *elitism* dan merupakan hasil terbaik yang didapatkan oleh sistem. Pada halaman ini terdapat empat kolom yang berisi kode jadwal bimbingan yaitu individu yang terbaik dihasilkan oleh sistem, kolom jam atau hari yang merupakan jam dan hari untuk mahasiswa tersebut melakukan bimbingan berdasarkan dari kode, kolom minggu ditentukan berdasarkan kode jadwal dimana P0 terdapat lima minggu dan dalam satu minggu terdapat 50 slot jadwal yang harus diisi, kolom nama mahasiswa yaitu *list* nama beberapa mahasiswa yang mengambil skripsi. Dibawah tabel jadwal terdapat nilai *fitness* terbaik dan generasi terbaik yang merupakan hasil dari solusi optimal yang dihasilkan oleh sistem dalam hal ini nilai *fitness* terbaik yaitu 2,0833 dan generasi terbaik yaitu ke-108. Pengguna dapat menekan tombol *Export* ke Excel yang berfungsi untuk meng-*export* hasil jadwal yang diperoleh kedalam bentuk excel. File tersebut sesuai dengan jadwal yang telah di hasilkan oleh sistem dan akan disimpan kedalam folder skripsi/*output*/jadwal.xls. Pada sistem ini setiap kali pengguna menekan tombol *export* hasil *output* akan selalu di perbarui. Pengguna yang ingin melakukan perubahan parameter dapat menjalankan kembali aplikasi tersebut.



	A	B	C	D	E
1	Kode Jadw	Hari/Jam	Minggu	Nama Mahasiswa	
2	123	rabu, 11.10-12.00	3	Rizka Rizkiana	
3	181	senin, 13.40-14.30	4	Ana binti	
4	151	Senin , 07.00-08.40	4	Diah Sinta Dewi	
5	106	senin, 08.40-09.30	3	Weni Agustina	
6	127	selasa, 12.50-13.40	3	Fitri Anggarsari	
7	115	jumat, 09.30-10.20	3	Khairiyah Nur Aisya	
8	115	jumat, 09.30-10.20	3	Nur Afiffah	
9	124	kamis, 11.10-12.00	3	Mimin Putri	
10	135	jumat, 13.40-14.30	3	Karina Widyawati	
11	55	jumat, 07.00-08.40	2	Muthia Azzahra	
12	127	selasa, 12.50-13.40	3	Vera Rusmalawati	
13	128	rabu, 12.50-13.40	3	Shelly Puspa Ardina	
14	79	kamis, 12.50-13.40	2	Marina Debora R	
15	134	kamis, 13.40-14.30	3	Nur Cahyo Utomo	
16	204	kamis, 07.00-08.40	5	Fadhilla P Cahyani	
17	105	jumat, 07.00-08.40	3	Muhammad Ali Aras R	
18	180	jumat, 12.50-13.40	4	Fiki Nurhadiyanto	
19	175	jumat, 11.10-12.00	4	Kukuh Wiliam M	
20	175	jumat, 11.10-12.00	4	Daneswara Jauhari	
21	122	selasa, 11.10-12.00	3	Rich Juniadi D S	
22	225	jumat, 11.10-12.00	5	Anang Hanafi	
23	140	jumat,14.30-15.20	3	Winda Cahyaningrum	
24	150	jumat, 16.10-17.10	3	Dwi Novi Setiawan	
25	102	selasa, 07.00-08.40	3	Daffarez Elguska	
26	97	selasa, 16.10-17.10	2	Ferdy Wahyurianto	
27	196	senin, 16.10-17.10	4	Edgar Juvianno S	

Jadwal

Gambar 5.5 Implementasi Halaman Hasil Jadwal pada Excel