BAB IV
IMPLEMENTASI

4.1 Sumber Data

Sumber data (order) yang digunakan pada aplikasi ini didapat dari sebuah perusahaan *body repair* "RISKY" Jl. LA. Sucipto Malang.

4.2 Lingkungan Implementasi

Proses implementasi merupakan tahapan penerapan rancangan optimasi ini pada bahasa pemrograman yang dapat dimengerti oleh komputer. Lingkungan implementasi yang akan dijelaskan pada bab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.2.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem optimasi waktu menggunakan algoritma genetika ini adalah sebagai berikut :

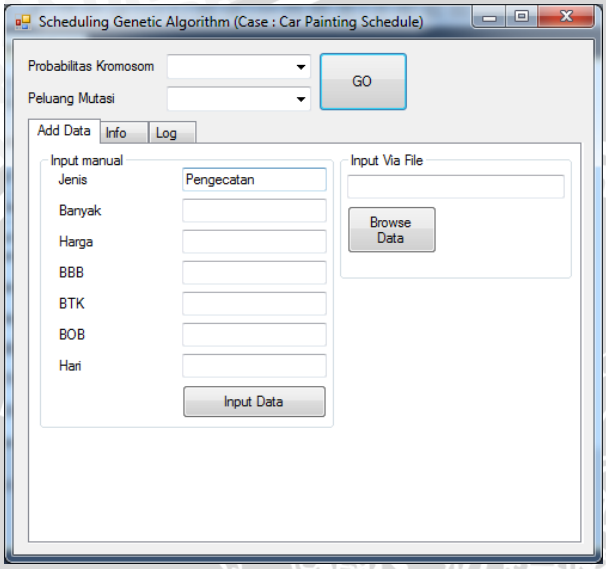
1. Processor Intel Pentium M 2,00 GB
2. RAM 2,00 GB
3. Harddisk dengan kapasitas 75 GB
4. Monitor 14.1'
5. Keyboard
6. Mouse

4.2.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pembuatan sistem aplikasi optimasi waktu menggunakan algoritma genetika ini adalah Microsoft Visual C# sebagai software development dalam implementasi rancangan sistem.

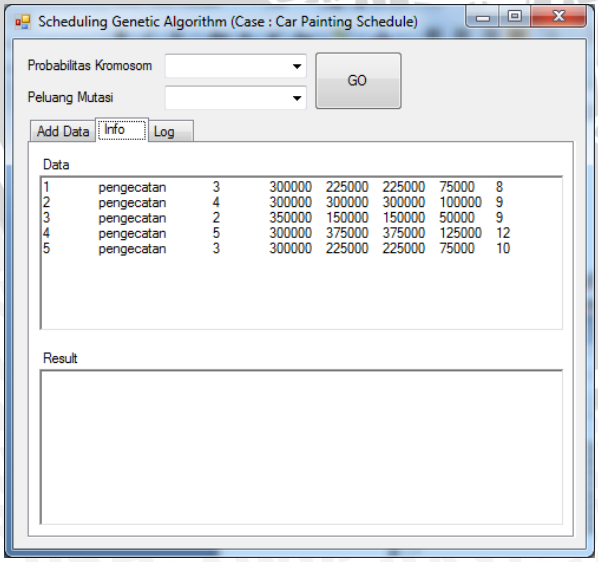
4.3 Implementasi Optimasi Waktu *Body Repair* Menggunakan Algoritma Genetika

Tampilan form utama aplikasi optimasi *body repair* menggunakan Algoritma Genetik dapat dilihat pada pada gambar 4.1.



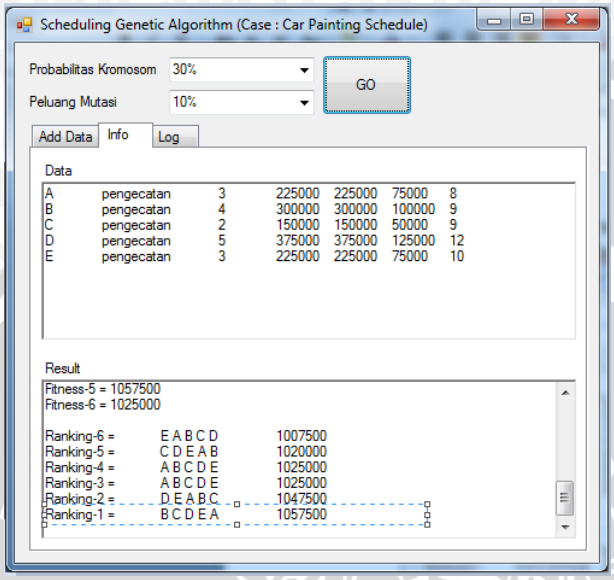
Gambar 4.1 Tampilan utama aplikasi

Untuk tampilan data aplikasi optimasi *body repair* menggunakan Algoritma Genetik dapat dilihat pada pada gambar 4.2.



Gambar 4.2 Tampilan Data

Untuk tampilan hasil terbaik dari proses genetik aplikasi optimasi menggunakan Algoritma Genetik dapat dilihat pada pada gambar 4.3



Gambar 4.3 Tampilan Hasil Terbaik Proses Genetik

4.4 Implementasi Program Optimasi Menggunakan Algoritma Genetika

Berdasarkan perancangan pada bab 3, maka pada bab ini akan dibahas mengenai implementasi dari perancangan tersebut. Secara garis besar proses terbagi menjadi 5 bagian utama, yaitu representasi kromosom, proses *crossover*, proses mutasi, proses menghitung fungsi *fitness* dan proses seleksi rangking.

4.4.1 Representasi Kromosom

Representasi kromosom dilakukan berdasarkan nama order. Jumlah gen yang dihasilkan sesuai dengan jumlah order yang akan diproses.

```
1         //representasi kromosom
2         int[][] kromosom = new int[data.Count][];
3         for (int a = 0; a < data.Count; a++)
4         {
5            kromosom[a] = new int[data.Count];
6            for (int b = 0; b < data.Count; b++)
7            {
8               kromosom[a][b] = temp1[b];
9               temp2[b] = temp1[b];
10           }
11           temp1[0] = temp2[data.Count - 1];
12           for (int b = 1; b < data.Count; b++)
13           {
14              temp1[b] = temp2[b - 1];
15           }
16        }
17
18        richTextBox2.Text += "\n";
19        for (int a = 0; a < data.Count; a++)
20        {
21           richTextBox2.Text += "Kromosom-" + a + " = ";
22           for (int b = 0; b < data.Count; b++)
23           {
24              richTextBox2.Text += kromosom[a][b];
25           }
26           richTextBox2.Text += "\n";
27        }
```

Source code 4.1 Representasi Kromosom

4.4.2 Crossover

Proses *crossover* dalam skripsi ini menggunakan metode *Position Based Crossover*. Langkah pertama adalah dengan membangkitkan nilai *random* antara 0 sampai 1 untuk setiap kromosom. Kromosom yang mempunyai nilai < Pc maka kromosom tersebut yang mengalami *crossover*. Kemudian memasangkan kromosom dan memilih secara acak posisi gen pada *parent 1*. Menyalin gen yang sudah terpilih kedalam *protochild.* Menghapus gen-gen pada *parent 2* yang sama dengan *proto-child* dan menempatkan gen *parent* 2 yang tersisa secara berurutan ke *proto-child. Protochild* terakhir merupakan krormosom anak *(offspring).*

```
1   //crossover
2           double[] rvKromosom = new double[data.Count];
3           List<int> coKromosom = new List<int>();
4           double Pc = 0, Pm = 0;
5
6           if (comboBox1.SelectedIndex == 0)
7           {
8               Pc = 0.1;
9           }
10          else if (comboBox1.SelectedIndex == 1)
11          {
12              Pc = 0.3;
13          }
14          else if (comboBox1.SelectedIndex == 2)
15          {
16              Pc = 0.5;
17          }
18          else if (comboBox1.SelectedIndex == 3)
19          {
20              Pc = 0.7;
21          }
```

```
22          else if (comboBox1.SelectedIndex == 4)
23          {
24              Pc = 0.9;
25          }
26
27          if (comboBox2.SelectedIndex == 0)
28          {
29              Pm = 0.1;
30          }
31          else if (comboBox2.SelectedIndex == 1)
32          {
33              Pm = 0.3;
34          }
35          else if (comboBox2.SelectedIndex == 2)
36          {
37              Pm = 0.5;
38          }
39          else if (comboBox2.SelectedIndex == 3)
40          {
41              Pm = 0.7;
42          }
43          else if (comboBox2.SelectedIndex == 4)
44          {
45              Pm = 0.9;
46          }
47
48          for (int a = 0; a < data.Count; a++)
49          {
50              rvKromosom[a] = r.NextDouble();
51              if (rvKromosom[a] < Pc)
52              {
```

```
53              coKromosom.Add(a);
54          }
55      }
56
57      for (int a = 0; a < coKromosom.Count; a++)
58      {
59          richTextBox2.Text += coKromosom[a] + "\n";
60      }
61
62      List<int> kromosom1 = new List<int>();
63      List<int> kromosom2 = new List<int>();
64
65      for (int a = 0; a < coKromosom.Count; a++)
66      {
67          for (int b = 0; b < coKromosom.Count; b++)
68          {
69              if ((coKromosom[a] != coKromosom[b]))
70              {
71                  kromosom1.Add(coKromosom[a]);
72                  kromosom2.Add(coKromosom[b]);
73              }
74          }
75      }
76
77      for (int a = 0; a < kromosom1.Count; a++)
78      {
79          for (int b = 0; b < kromosom1.Count; b++)
80          {
81              if (a == b)
82              {
83                  continue;
```

```
84                  }
85                  else if ((kromosom1[a] == kromosom2[b]) &&
86  (kromosom2[a] == kromosom1[b]))
87                  {
88                      kromosom1.RemoveAt(b);
89                      kromosom2.RemoveAt(b);
90                  }
91              }
92          }
93
94          for (int a = 0; a < kromosom1.Count; a++)
95          {
96              richTextBox2.Text += kromosom1[a] + "\t" + kromosom2[a] +
97  "\n";
98          }
99
100         int[][] offspring = new int[kromosom1.Count][];
101         for (int a = 0; a < kromosom1.Count; a++)
102         {
103             offspring[a] = new int[data.Count];
104             List<int> rest = new List<int>();
105             for (int b = 0; b < kromosom[kromosom1[a]].Length; b++)
106             {
107                 if (b % 2 == 0)
108                 {
109                     offspring[a][b] = kromosom[kromosom1[a]][b];
110                     rest.Add(kromosom[kromosom1[a]][b]);
111                 }
112                 else
113                 {
114                     offspring[a][b] = 99;
```

```
115             }
116         }
117
118         for (int b = 0; b < kromosom[kromosom1[a]].Length; b++)
119         {
120             if (!rest.Contains(kromosom[kromosom2[a]][b]))
121             {
122                 rest.Add(kromosom[kromosom2[a]][b]);
123             }
124         }
125
126         if (kromosom[kromosom1[a]].Length % 2 == 0)
127         {
128             for (int b = 0; b < kromosom[kromosom1[a]].Length / 2; b++)
129             {
130                 rest.RemoveAt(0);
131             }
132         }
133         else
134         {
135             for (int b = 0; b < (kromosom[kromosom1[a]].Length / 2) + 1;
136 b++)
137             {
138                 rest.RemoveAt(0);
139             }
140         }
141
142         for (int b = 0; b < kromosom[kromosom1[a]].Length; b++)
143         {
144             if (offspring[a][b] == 99)
145             {
```

```
146                 offspring[a][b] = rest[0];
147                 rest.RemoveAt(0);
148             }
149         }
150     }
151
152     for (int a = 0; a < kromosom1.Count; a++)
153     {
154         richTextBox2.Text += "Offspring-" + a + " = ";
155         for (int b = 0; b < kromosom[kromosom1[a]].Length; b++)
156         {
157             richTextBox2.Text += offspring[a][b];
158         }
159         richTextBox2.Text += "\n";
160     }
161
162     int[][] population = new int[kromosom.Length +
163 offspring.Length][];
164     for (int a = 0; a < population.Length; a++)
165     {
166         population[a] = new int[data.Count];
167         if (a < kromosom.Length)
168         {
169             for (int b = 0; b < data.Count; b++)
170             {
171                 population[a][b] = kromosom[a][b];
172             }
173         }
174         else
175         {
176             for (int b = 0; b < kromosom[kromosom1[0]].Length; b++)
```

```
177             {
178                 population[a][b] = offspring[a - kromosom.Length][b];
179             }
180         }
181     }
182
183     for (int a = 0; a < population.Length; a++)
184     {
185         richTextBox2.Text += "Population-" + a + " = ";
186         for (int b = 0; b < data.Count; b++)
187         {
188             richTextBox2.Text += population[a][b];
            }
            richTextBox2.Text += "\n";
        }
```

Source code 4.2 Proses *Crossover*

4.4.3 Mutasi

Proses mutasi menggunakan metode *Reciprocal Exchange Mutation*. Mutasi dilakukan berdasarkan pada probabilitas mutasi (Pm) yang telah ditentukan. Langkah pertama yaitu membangkitkan nilai *random* antara 0 sampai 1 untuk setiap kromosom. Kromosom yang mempunyai nilai < Pm maka kromosom tersebut yang mengalami mutasi. Memilih 2 posisi gen secara acak. Kemudian saling menukarkan posisi gen yang terpilih.

```
1   //mutasi
2           double[] rpmKromosom = new double[population.Length];
3           for (int a = 0; a < population.Length; a++)
4           {
5               rpmKromosom[a] = r.NextDouble();
6               richTextBox2.Text += "rpmKromosom" + a + " = " +
7   rpmKromosom[a] + "\n";
8               try
9               {
10                  if (rpmKromosom[a] < Pm)
11              {
12                  int tmp = 0, r1 = 0, r2 = 0;
13                  do
14                  {
15                      r1 = population[a][r.Next(0, data.Count)];
16                      r2 = population[a][r.Next(0, data.Count)];
17                  }
18                  while (r1 == r2);
19                  tmp = population[a][r1];
20                  population[a][r1] = population[a][r2];
21                  population[a][r2] = tmp;
22              }
23              }
24              catch (Exception err)
25              {
26                  continue;
27              }
28          }
29
30              for (int a = 0; a < population.Length; a++)
31              {
```

```
32              richTextBox2.Text += "Mutated Population-" + a + " = ";
33              for (int b = 0; b < data.Count; b++)
34              {
35                  richTextBox2.Text += population[a][b];
36              }
37              richTextBox2.Text += "\n";
38          }
```

Source code 4.3 Proses Mutasi

4.4.4 Fungsi *Fitness*

Proses pertama yang dilakukan dalam menghitung *fitness* yaitu menghitung profit bruto yaitu banyak panel * harga – (bbb+btk+bob), menghitung total profit bruto, mengitung total hari pengerjaan dan kemudian mengecek cashback maka didapatkan profit netto yang ditentukan sebagai *fitness* akhir.

```
1   //fitness
2           int[] profitBruto = new int[data.Count];
3           for (int a = 0; a < data.Count; a++)
4           {
5               profitBruto[a] = data[a].banyak * data[a].harga - (data[a].bbb +
6   data[a].btk + data[a].bob);
7               richTextBox2.Text += "Profit Bruto-" + a + " = " + profitBruto[a]
8   + "\n";
9           }
10
11          int[][,] panel = new int[population.Length][,];
12          for (int a = 0; a < population.Length; a++)
13          {
14              panel[a] = new int[5, data.Count];
```

```
15              for (int b = 0; b < 5; b++)
16              {
17                  for (int c = 0; c < data.Count; c++)
18                  {
19                      try
20                      {
21                          panel[a][b, c] = data[population[a][c] - 1].banyak;
22                      }
23                      catch (Exception err)
24                      {
25                          continue;
26                      }
27                  }
28              }
29          }
30
31          for (int a = 0; a < population.Length; a++)
32          {
33              richTextBox2.Text += "Panel Value-" + a + " : \n";
34              for (int b = 0; b < 5; b++)
35              {
36                  for (int c = 0; c < 5; c++)
37                  {
38                      richTextBox2.Text += panel[a][b, c] + " ";
39                  }
40                  richTextBox2.Text += "\n";
41              }
42              richTextBox2.Text += "\n";
43          }
44
45          int[][] hitungHari = new int[population.Length][];
```

```
46          int[] maxDay = new int[population.Length];
47          int[][] calcDay = new int[population.Length][];
48          int[][] stopDay = new int[population.Length][];
49          //solution3
50          for (int a = 0; a < population.Length; a++)
51          {
52              hitungHari[a] = new int[data.Count];
53              calcDay[a] = new int[3];
54              stopDay[a] = new int[data.Count];
55              List<int> todolist = new List<int>();
56
57              for (int b = 0; b < data.Count; b++)
58              {
59                  todolist.Add(population[a][b]);
60              }
61
62              richTextBox2.Text += "\nto do list : ";
63              for (int b = 0; b < data.Count; b++)
64              {
65                  richTextBox2.Text += todolist[b] + " ";
66              }
67              richTextBox2.Text += "\n";
```

Source code 4.4 Proses Hitung Fitness

4.4.5 Seleksi Rangking

Proses seleksi digunakan yaitu seleksi Rangking. Seleksi dilakukan dengan tujuan untuk memilih individu yang ikut dalam proses reproduksi selanjutnya. Kromosom diurutkan berdasakan nilai *fitness*. Kromosom dengan nilai terbaik akan ditentukan sebagai hasil teroptimasi.

```
1
2           List<ranking> rank = new List<ranking>();
3           for (int a = 0; a < population.Length; a++)
4           {
5              string temps = "";
6              for (int b = 0; b < data.Count; b++)
7              {
8                 temps +=  Convert.ToChar(64 + population[a][b]) + " ";
9              }
10             rnk.nama = temps;
11             rnk.fitness = fitness[a];
12             rank.Add(rnk);
13          }
14
15          rank.Sort
16                (
17                   delegate(ranking p1, ranking p2)
18                   {
19                      return p1.fitness.CompareTo(p2.fitness);
20                   }
21                );
22
23          richTextBox3.Text += "\n";
24          for (int a = 0; a < population.Length; a++)
25          {
26             richTextBox3.Text += "Ranking-" + (population.Length - a) + " =
27 \t" + rank[a].nama + "\t" + rank[a].fitness + "\n";
28          }
29       }
```

Source code 4.5 Proses Pengurutan Kromosom