KECERDASAN KOMPUTASIONAL A

DOKUMENTASI PROGRAM

“Implementasi EP (Evolutionary Programming) untuk Pemilihan Calon Isteri Idaman”

Penyusun :

Utama Nur Ariputra

15/388508/PPA/04947

PROGRAM PASCASARJANA ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

2016

1. Pendahuluan

Evolutionary programming sebagai salah satu cabang evolutionary computation telah sangat berkembang dan dapat diimplementasikan ke dalam berbagai masalah. Pada tugas kecerdasan komputasional ini, evolutionary programming dikembangkan untuk menentukan tingkat kecocokan istri idaman para jomblowan yang mengharapkan ingin melangsungkan pernikahan tetapi di hadapkan pada pilihan calon istri yang lebih dari satu.

Pemilihan kasus dilatarbelakangi karena pada kehidupan sosial saat ini, banyak pria yang belum memiliki standar khusus dalam menentukan calon istri yang baik, yang kelak untuk anaknya. Setiap pria pasti mengharapkan kehadiran sosok istri yang tepat dalam membina dan mengarungi keluarga kelak sampai maut memisahkan dan sampai menghasilkan keturunan yang baik dan sukses dunia dan akhirat.

Dari penjelasan diatas, sehingga perlu dirancang sebuah metode pelatihan dengan menggunakan evolutionary programming dengan berdasarkan fitur yang diberikan, yakni:

-Kerohanian

-Kecantikan

-Kemapanan

-Keturunan

Dari keempat fitur tersebut kemudian akan direpresentasikan kedalam decision tree sehingga dapat diolah dengan urutan langkah algoritma evolutionary programming dan ditentukan apakah calon istri tersebut tersebut "diterima" atau "ditolak" sebagai calon istri idaman para pria.

2. Landasan Teori

- **Evolutionary Programming**

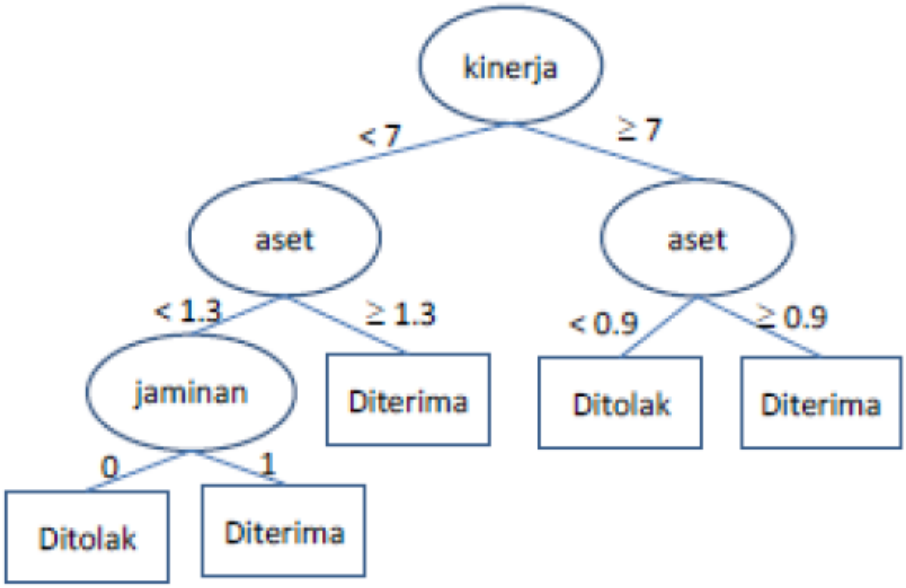
  Representasi yang digunakan dalam evolutionary programming biasanya disesuaikan dengan domain masalah. Salah satu representasi yang umum digunakan adalah vektor bernilai real dengan panjang tetap.

  Perbedaan utama antara evolutionary programming dan pendekatan sebelumnya adalah bahwa tidak ada pertukaran gen antara individu dalam populasi (crossover). Dengan demikian, hanya operator mutasi yang digunakan.

Untuk representasi vektor bernilai real, evolutionary programming sangat mirip dengan evolutionary strategy (ES) tanpa rekombinasi. Namun yang membedakan di antaranya adalah penjabaran nilai kromosom dilakukan dalam bentuk decision tree. Evolutionary Programming memiliki tujuan seperti GP untuk menghasilkan rangkaian program komputer tetapi prinsip kerjanya seperti ES.

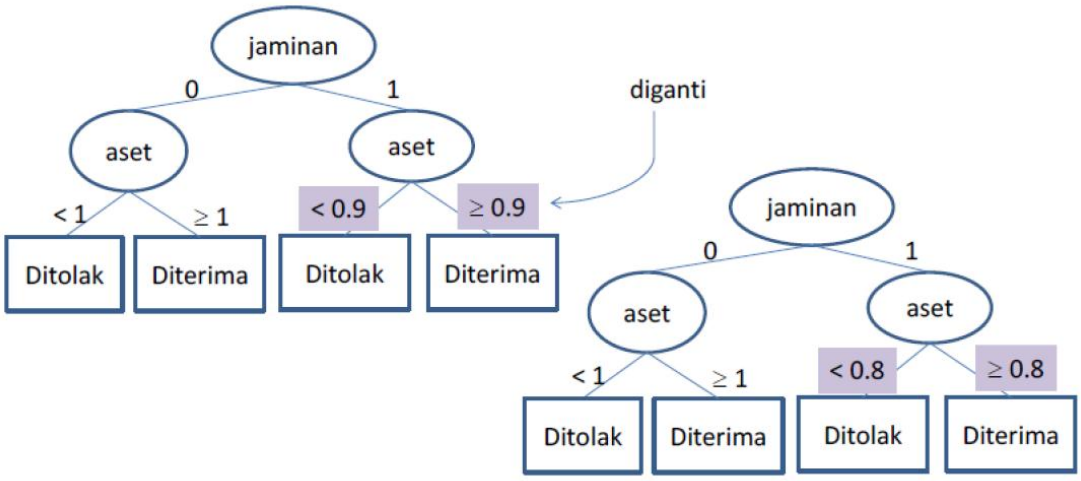
Representasi Kromosom

Seperti yang telah dijelaskan sebelumnya, evolutionary programming melakukan representasi kromosom dengan decision tree. Perbedaan evolutionary programming (EP) dengan genetic programming (GP) adalah nilai kromosom EP diletakkan pada edge tree, sedangkan pada GP diletakkan di dalam node.



- **Mutasi**

  Berbeda dari genetic programming yang melakukan pelatihan dengan menggunakan crossover dan mutasi, evolutionary programming hanya menggunakan mutasi saja. Proses mutasi dalam evolutionary programming adalah dengan mengganti nilai edge yang dirasa kurang akurat dengan nilai lain pada setiap iterasinya.



Evaluasi

Proses evaluasi pada evolutionary programming tidak jauh berbeda dengan algoritma evolutionary computation lainnya, yakni dengan menghitung nilai fitness dari setiap individu yang ada. Perhitungan nilai fitness adalah dengan menghitung derajat kebenaran dari setiap prediksi yang dilakukan. Setiap hasil yang sesuai prediksi akan diberi nilai atau skor sedangkan jika ternyata salah maka tidak diberi nilai. Skor tersebut kemudian dijumlah sehingga didapatkanlah nilai fitness pada individu.

No	Aset	Kinerja	Jaminan	Keputusan	Individu P1		Individu P2	
					Keputusan	Skor	Keputusan	Skor
1	1,5	6	0	0	1	0	1	0
2	0,7	6	1	0	1	0	0	1
3	2	7	0	1	1	1	1	1
4	1,6	5	1	1	1	1	1	1
5	0,9	8	0	1	1	1	0	0
6	0,8	8	1	1	0	0	0	0
		Fitness				3		3

3. Penjelasan Program

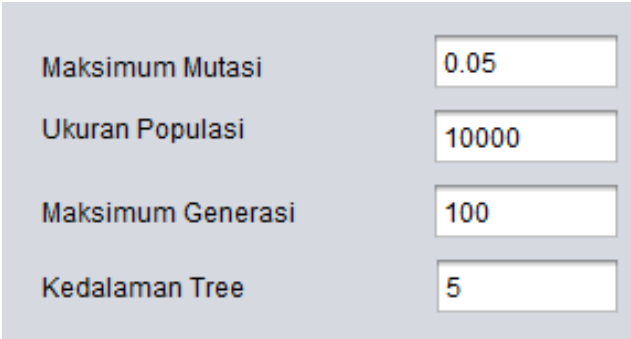
Program penentuan calon isteri idaman ini dibuat dengan bahasa pemrograman JAVA dengan menggunakan empat kelas utama, yakni kelas EvolutionaryProgramming, kelas Mutasi, Kelas PilihEdge dan kelas Start.

Kelas Evolutionary Programming.java terdiri dari kerangka urutan algoritma evolutionary programming mulai dari inisialisasi sampai pada tahap evaluasi.

Kelas Mutasi.java melakukan proses mutasi terhadap edge.

Kelas PilihEdge akan menentukan edge mana yang akan dipilih sesuai dengan perbandingan nilai fitur terhadap nilai edge.

Sedangkan kelas Start.java berisi Graphical User Interface (GUI) yang dapat memudahkan user dalam melakukan perubahan parameter. Berikut merupakan parameter yang digunakan sebagai inisialisasi pelatihan.



- Maksimum mutasi merupakan derajat jumlah mutasi yang dilakukan pada seluruh gen.
- Ukuran populasi adalah jumlah dari total populasi yang digunakan.
- Maksimum generasi adalah nilai maksimum dari iterasi yang dilakukan.
- Kedalaman tree adalah nilai maksimum dari kedalaman / level dari tree yang dibentuk.

Berikut merupakan penjelasan terkait kode fungsi utama yang dibentuk pada masing-masing kelas.

1. **Fungsi Mulai Pelatihan**

Fungsi ini terdapat di dalam Start.java. Fungsi ini akan menjadi handler awal ketika tombol “mulai operasi” ditekan.

Bilangan yang dimasukkan akan dikonversi ke double dan integer sesuai dengan parameter yang diinputkan. Fungsi kemudian akan melakukan pengambilan nilai parameter dan kemudian melemparkannya ke kelas Evolutionary Programming untuk diolah.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       if(lokasiFile.equals("")){
          JOptionPane.showMessageDialog(null, "Data belum diload");
       }
       else {
           mutasi =  Double.parseDouble(jTextField1.getText());
           popsize =  Integer.parseInt(jTextField3.getText());
           generasi =  Integer.parseInt(jTextField4.getText());
           tree =  Integer.parseInt(jTextField5.getText());

          try {
             System.out.print("Pelatihan sedang dimulai . . . \n");
             JOptionPane.showMessageDialog(null, "Pelatihan data sedang berlangsung, harap sabar
menunggu");
             Thread.sleep(2000);
          } catch (InterruptedException ex) {
             Logger.getLogger(Start.class.getName()).log(Level.SEVERE, null, ex);
          }
           EvolutionaryProgramming evo = new EvolutionaryProgramming(lokasiFile,mutasi, popsize,
tree);
           evo.evaluasi(generasi);
       }
    }
```

2. Fungsi Inisialisasi

Fungsi inisialisasi akan mengambil semua parameter yang diperlukan dalam pelatihan, termasuk file “problem.dat” yang menjadi data kasus dan nilai ekspektasinya dengan menggunakan BufferedReader(). Isi dari file kemudian akan di trim kemudian akan dipisahkan per baris sebagai data kasus.

Fungsi inisialisasi juga melakukan error handling apabila terdapat kesalahan pada tiap parameter yang dimasukkan.

```
void inisialisasi(String fname) {
   try {
     int i,j;
     String line;
     BufferedReader in =    new BufferedReader(new FileReader(fname));
     line = in.readLine();
     StringTokenizer tokens = new StringTokenizer(line);
     varnumber = Integer.parseInt(tokens.nextToken().trim());
     randomnumber = Integer.parseInt(tokens.nextToken().trim());
     minrandom =  Double.parseDouble(tokens.nextToken().trim());
     maxrandom =  Double.parseDouble(tokens.nextToken().trim());
     fitnesscases = Integer.parseInt(tokens.nextToken().trim());
     targets = new double[fitnesscases][varnumber+1];
     if (varnumber + randomnumber >= FSET_START )

  System.out.println("variable TERLALU BANYAK");
   for (i = 0; i < fitnesscases; i ++ ) {
       line = in.readLine();
       tokens = new StringTokenizer(line);
       for (j = 0; j <= varnumber; j++) {
         targets[i][j] = Double.parseDouble(tokens.nextToken().trim());
           }      }
     in.close();
   }
   catch(FileNotFoundException e) {
     System.out.println("ERROR: Maaf, data tidak tersedia");
     System.exit(0);
   }
   catch(Exception e ) {
     System.out.println("ERROR: Maaf, format data salah");
     System.exit(0);
   }
}
```

3. Fungsi Pemilihan Gen

Fungsi pemilihan gen akan menentukan jalur mana yang dipilih sebagai rutenya dengan berdasarkan pada nilai edge masing-masing. Misalnya pada sebuah node memiliki edge <0.3 dan >=0.3, kemudian ada nilai sebesar 0.5. Maka jalur edge yang diambil adalah >=0.3. Fungsi ini berlaku untuk setiap fitur yang tersedia.

```
public class PilihEdge {


  double run() {
    char primitive = program[PC++];
    if ( primitive < FSET_START )
      return(x[primitive]);
    switch ( primitive ) {
      case ROH : {
              double edge1 = run(), edge2 = run();
              if (edge1 <= edge2) return( edge1 );
              else return (edge2);
            }
      case CAN : {
              double edge1 = run(), edge2 = run();
              if (edge1 <= edge2) return( edge1 );
              else return (edge2);
            }
      case MAP : {
              double edge1 = run(), edge2 = run();
              if (edge1 <= edge2) return( edge1 );
              else return (edge2);
            }
      case KET : {
              double edge1 = run(), edge2 = run();
              if (edge1 <= edge2) return( edge1 );
              else return (edge2);
            }
      }
    return( 0.0 );
  }
}
```

4. Fungsi Hitung Nilai Fitness

Fungsi hitung nilai fitness akan melakukan kalkulasi terhadap nilai fitness pada tiap individu generasi.

Fungsi diawali dengan mengambil hasil pelatihan dan target yang hendak dicapai berupa multidimensional array. Nilai fitness disini merupakan jumlahan dari selisih nilai yang didapatkan dengan nilai target yang diharapkan yang diperoleh dari dalam file “problem.dat”

```
double fitness_function( char [] Prog ) {
  int i = 0, len;
  double result, fit = 0.0;
  len = traverse( Prog, 0 );
  for (i = 0; i < fitnesscases; i ++ ) {
    for (int j = 0; j < varnumber; j ++ )
       x[j] = targets[i][j];
    program = Prog;
    PC = 0;
    result = run();
    fit += Math.abs( result - targets[i][varnumber]);
    }
  return(-fit );
}
```

5. Fungsi Mutasi

Fungsi mutasi akan melakukan mutasi terhadap nilai edge pada path yang dipilih. Nilai yang digunakan untuk merubahnya adalah random sesuai dengan parameter awal yang telah user masukkan. Mutasi juga akan dilakukan sebanyak jumlah derajat mutasi yang diberikan oleh user.

Nilai yang di-return pada fungsi ini adalah nilai array char dari parent dengan edge baru.

```
public class Mutasi {
 char [] mutasi( char [] parent, double pmut ) {
    int len = EvolutionaryProgramming.traverse( parent, 0 ), i;
    int mutsite;
    char [] parentcopy = new char [len];

    System.arraycopy( parent, 0, parentcopy, 0, len );
    for (i = 0; i < len; i ++ ) {
      if ( rd.nextDouble() < pmut ) {
      mutsite =  i;
      if ( parentcopy[mutsite] < FSET_START )
        parentcopy[mutsite] = (char) rd.nextInt(varnumber+randomnumber);
      else
        switch(parentcopy[mutsite]) {
            case ROH:
            case CAN:
            case MAP:
            case KET:
          parentcopy[mutsite] =
            (char) (rd.nextInt(FSET_END - FSET_START + 1)
                  + FSET_START);
        }
      }
    }
    return( parentcopy );
  }
}
```

6. Fungsi Evaluasi

Fungsi evaluasi akan mengambil offspring baru lalu menentukan individu mana yang akan digunakan di dalam generasi selanjutnya berdasarkan nilai fitness yang dimiliki oleh himpunan parent dan offspring. Fungsi evaluasi juga akan menentukan kapan sebuah operasi dapat berhenti dan mencetak hasilnya.

```
void evaluasi(int genr) {
   int gen = 0, indivs, offspring, parent1, parent2, parent;
   double newfit;
   char []newind;
   print_parms();
   stats( fitness, pop, 0 );
   for ( gen = 1; gen < genr; gen ++ ) {
    if (  fbestpop > -1e-5 ) {
    //System.out.print("PROBLEM SOLVED\n");
    System.exit( 0 );
    }
    for ( indivs = 0; indivs < POPSIZE; indivs ++ ) {
    parent = tournament( fitness, TSIZE );
    Mutasi mut = new Mutasi();
    newind = mut.mutasi(pop[parent], PMUT_PER_NODE );
    newfit = fitness_function( newind );
    offspring = negative_tournament( fitness, TSIZE );
    pop[offspring] = newind;
    fitness[offspring] = newfit;
    }
    stats( fitness, pop, gen );
   }
```

4. *Hasil dan Kesimpulan*

Program penentuan calon isteri idaman ini merupakan salah satu contoh penerapan evolutionary programming dalam membantu penyelesaian masalah sederhana di kehidupan sehari-hari.

Hasil yang diperoleh dari pelatihan keempat fitur menggunakan program ini beragam sesuai dengan parameter mutasi, besaran populasi, kedalaman tree dan maksimum generasi yang ditentukan.

Meskipun program ini sudah dapat membantu menetapkan siapa saja yang dapat diterima sebagai calon menantu dalam bentuk representasi kromosom berupa decision tree, tapi tidak menutup kemungkinan jumlah fitur atau kriteria yang dimasukkan dalam pelatihan dapat ditambah untuk dapat meningkatkan akurasi dari hasil yang diperoleh.