KECERDASAN KOMPUTASIONAL A

DOKUMENTASI PROGRAM

"Implementasi Evolutionary Programming untuk Pemilihan Calon Menantu Idaman"

Penyusun :

Nurcahya Pradana T.P.

15/388492/PPA/04931

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 buatan ini, evolutionary programming dikembangkan untuk menentukan tingkat kecocokan menantu idaman orang tua. Pemilihan kasus dilatarbelakangi karena pada kehidupan sosial saat ini, banyak orang tua yang belum memiliki standar khusus dalam menentukan calon menantu yang baik untuk anaknya. Setiap orang tua pasti mengharapkan kehadiran sosok menantu yang tepat di tengah-tengah keluarga mereka sehingga perlu dirancang sebuah metode pelatihan dengan menggunakan evolutionary programming dengan berdasarkan parameter yang diberikan, yakni:

-Kerohanian

-Ketampanan

-Pekerjaan

-Keturunan

Dari keempat parameter tersebut kemudian akan direpresentasikan kedalam decision tree sehingga dapat ditentukan apakah orang tersebut "diterima" atau "ditolak" sebagai menantu.

2. Landasan Teori

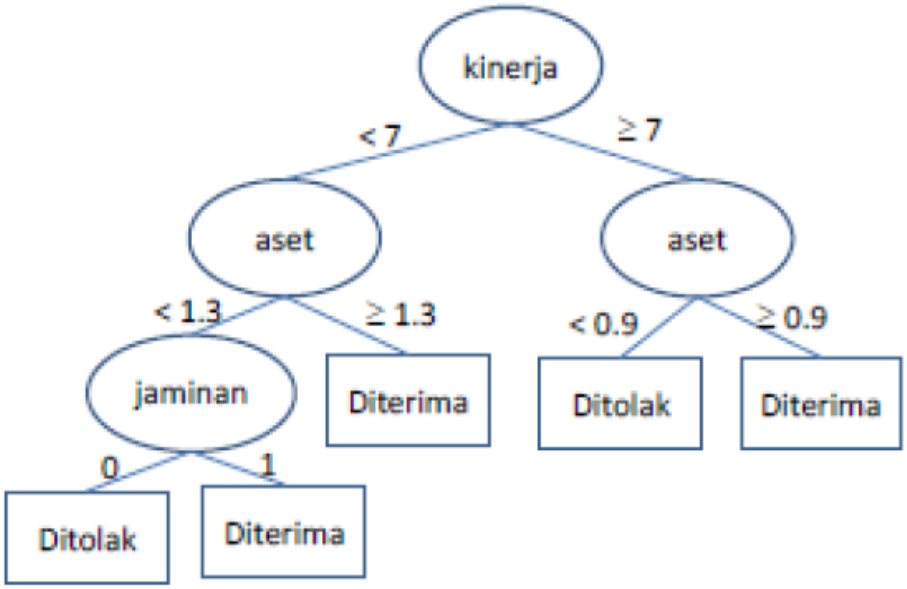
- **Evolutionary Programming**

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.

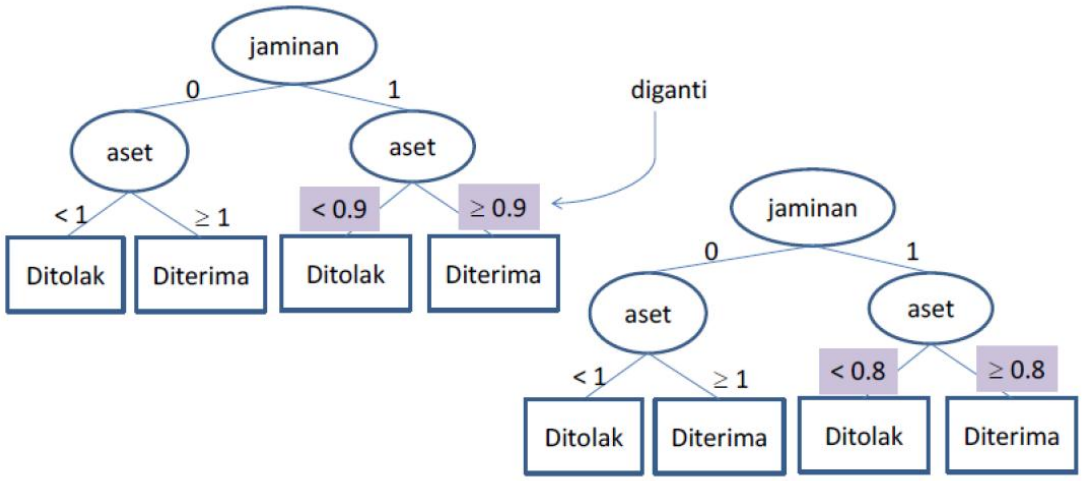
- **Representasi Kromosom**

  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.



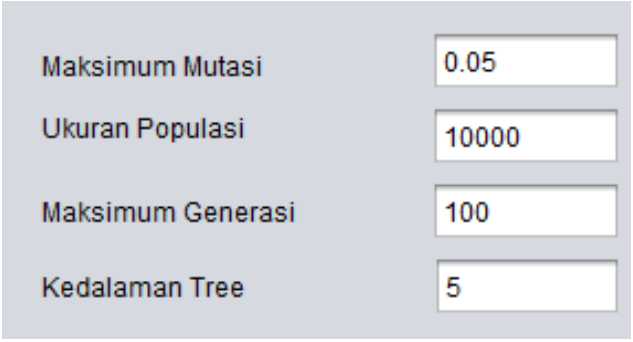
- **Evaluasi**

  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 menantu idaman ini dibuat dengan bahasa pemrograman JAVA dengan menggunakan tiga kelas utama, yakni kelas EvolutionaryProgramming, kelas Mutasi, dan kelas Start. Kelas EvolutionaryProgramming.java terdiri dari kerangka urutan algoritma evolutionary programming mulai dari inisialisasi sampai pada tahap evaluasi. Kelas Mutasi.java melakukan proses mutasi terhadap 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. Fungsi akan melakukan pengambilan nilai parameter dan kemudian melemparkannya ke kelas EvolutionaryProgramming 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 dimulai . . . \n");
            JOptionPane.showMessageDialog(null, "Pelatihan data sedang berlangsung, harap tunggu");
            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. 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("Terlalu banyak variabel");
  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: Data tidak tersedia");
     System.exit(0);
   }
   catch(Exception e ) {
     System.out.println("ERROR: 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.

```
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 TAM : {
            double edge1 = run(), edge2 = run();
            if (edge1 <= edge2) return( edge1 );
            else return (edge2);
          }
    case PEK : {
            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. Nilai fitness disini merupakan jumlahan dari selisih nilai yang didapatkan dengan nilai 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 TAM:
            case PEK:
            case KET:
          parentcopy[mutsite] =
            (char) (rd.nextInt(FSET_END - FSET_START + 1)
                  + FSET_START);
        }
      }
    }
    return( parentcopy );
  }
}
```

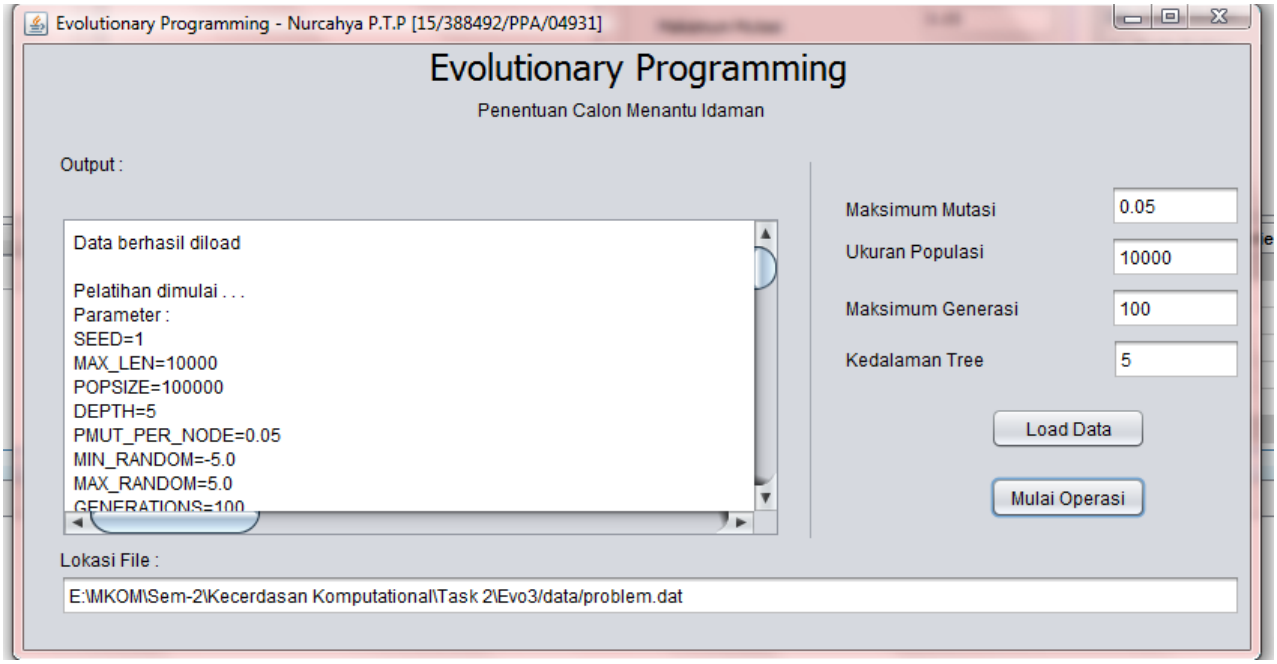
6. Fungsi Evaluasi

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. Screenshot Program

1. Perintah load data dan inisialisasi



2. Perintah mulai operasi dan hasilnya.