# BAB IV

# PEMBAHASAN PERANGKAT LUNAK

Bab ini dibagi menjadi tiga sub bab, yaitu penerapan algoritma genetika dan pembuatan perangkat lunak, pengujian perangkat lunak, dan sisi *entrepreneurship* dari perangkat lunak ini.

## 4.1 Penerapan Algoritma Genetika

Pada bagian ini akan dijelaskan mengenai proses pembuatan dari perangkat lunak menggunakan teori-teori dari algoritma genetika.

### 4.1.1 Fungsi Representasi Gen

Representasi gen merupakan merupakan fungsi pembuatan jadwal ujian dalam satu hari yang berisi ruangan, waktu, dan kelas (suatu mata kuliah yang diajar seorang dosen pada kumpulan mahasiswa tertentu). Fungsi ini mengambil data dari parameter vector kelas yang dikirim dari fungsi generateKromosom. Paramater yang diterima dari fungsi ini adalah parameter kumpulan kelas yang diambil dari basis data, jumlah kelas dalam satu hari yang disesuaikan dengan jumlah hari ujian, serta variabel newGen dengan tipe data boolean untuk menentukan apakah gen ini merupakan gen baru yang belum pernah dibuat. Proses ini dimulai dengan mengambil kelas secara acak dari vector kelas yang diterima dari fungsi generateKromosom.

Untuk menentukan jumlah waktu yang tersedia apakah cukup dengan jumlah sks dari kelas tersebut maka perlu dilakukan proses

evaluasi. Apabila dalam index ruangan dan waktu tertentu tidak ada kelas yang ada, maka akan dilakukan proses penambahan slot kosong mulai dari index ruangan dan waktu tersebut sampai jumlah waktu yang ada. Jika jumlah slot lebih besar daripada jumlah sks maka akan dilakukan pemasukkan kelas ke dalam index ruangan dan waktu tersebut.

Hasil dari proses ini mengembalikan vector Kelas yang berisi kumpulan objek Kelas dalam bentuk array dua dimensi dengan ruangan sebagai baris dan waktu sebagai kolom. Fungsi representasiGen ini berada dalam *class* Gen. Segmen Kode 4.1 merupakan penerapan dari pseudocode 3.1.

### 4.1.2 Fungsi Pembuatan Kromosom

Fungsi ini digunakan untuk pembuatan jadwal UAS selama delapan hari sesuai dengan rentang waktu ujian. Pembuatan kromosom dimulai dengan mengambil kumpulan kelas (mata kuliah dan dosen) dari basis data yang dibuat berdasarkan database Program Studi Teknik Informatika Universitas Ciputra. Kemudian dilakukan proses pengulangan dari fungsi representasiGen sebanyak paramater panjang kromosom.

Hasil dari fungsi ini adalah vector Gen yang berisi kumpulan dari gen yang telah dibuat. Fungsi ini dibuat di dalam *class* Kromosom. Segmen Kode 4.2 menunjukkan implementasi dari pseudocode 3.2

```
public Vector RepresentasiGen(Vector kelasV, int
kelasDalamSatuHari, boolean newGen) {
       waktuV = service.readAllWaktu();
       ruanganV = service.readAllRuangan();
       Random randomRuangan = new Random();
       Random randomWaktu = new Random();
       if (newGen) {
       gen = new Kelas[ruanganV.size()][waktuV.size()];
       }
       for (int i = 0; i < kelasDalamSatuHari; i++) {
       int idx = (int) (Math.random() * kelasV.size());
              if (kelasV.get(idx) == null) {
              while (kelasV.get(idx) == null) {
                           idx = (int) (Math.random() *
                           kelasV.size());
                           kelas = (Kelas) kelasV.get(idx);
                     }
              } else {
                     kelas = (Kelas) kelasV.get(idx);
              }
               int sks = kelas.getJumlahSks();
              boolean kelasSudahDiambil = false;
              while (kelasSudahDiambil == false) {
                     int y = randomRuangan.nextInt(ruanganV.size());
                     int x = randomWaktu.nextInt(waktuV.size());
                     ruangan = (Ruangan) ruanganV.get(y);
                     int slot = 0;
                     for (int j = x; j < waktuV.size(); j++) {
                     if (gen[y][j] == null) {
                                  slot++;
                           } else {
                           slot = 0;
                           }
                     }
                     if (slot >= sks) {
                           if(kelas.getTipeRuangan().equals(ruangan.
                           getTipeRuangan()) &&
                           kelas.getTipeKelas().equals(ruangan.getTi
                           peKelas())) {
                                  for (int k = 0; k < sks;) {
                                        if (gen[y][x + k] == null) {
                                               gen[y][x + k] = kelas;
                                               vectorOfKelas.add(gen[
                                               y][x + k]);
                                        }
                                        k++;
                           }
                                  kelasSudahDiambil = true;
                           } else {
                           kelasSudahDiambil = false;
                           }
                     }
              }
              kelasV.remove(idx);
        }
        return kelasV;
    }
```

**Segmen Kode 4.1 Fungsi RepresentasiGen**

```
public Vector generateKromosom(int length) {
        int kelasDalamSatuHari = kelasV.size() / length;
        for (int i = 0; i < length; i++) {
            gen = new Gen();
            kelasV = gen.RepresentasiGen(kelasV,
            kelasDalamSatuHari, true);
                  if (i < length - 1) {
                        vectorOfGen.add(gen);
                  }
        }
        kelasV = gen.RepresentasiGen(kelasV, 1, false);
        cekFitness(gen);
        vectorOfGen.add(gen);
        return vectorOfGen;
}
```

**Segmen Kode 4.2 Fungsi GenerateKromosom**

### 4.1.3 Fungsi Pembuatan Populasi

Fungsi ini berfungsi untuk pembuatan kumpulan kromosom sebanyak jumlah parameter yang diterima. Fungsi ini mengembalikan vector populasi yang berisi kumpulan kromosom. Fungsi ini ada dalam *class* Populasi. Segmen Kode 4.3 merupakan penerapan dari pseudocode 3.3.

```
public Vector inisialisasiPopulasi (int length) {
      kromosom = new Kromosom[length];
      for (int i = 0; i < length; i++) {
            kromosom[i] = new Kromosom();
            int a = i + 1;
            kromosom[i].generateKromosom(8);
            for (int j = 0; j < genV.size(); j++) {
                gen = (Gen) genV.get(j);
            }
            populasi.add(kromosom[i]);
       }
       return populasiV;
 }
```

**Segmen Kode 4.3 Pembuatan Populasi**

### 4.1.4 Fungsi Fitness Dosen

Fungsi ini berfungsi menghitung total kesalahan dari bentrokan dosen yang menjaga ujian dalam waktu bersamaan. Proses fungsi ini dimulai dengan menerima parameter gen untuk dievaluasi. Setiap dosen mata kuliah pada index ruangan dan index waktu dalam gen tersebut akan dievaluasi apakah sama dengan dosen mata kuliah pada index ruangan + 1. Apabila sama maka jumlah kesalahan akan bertambah satu. Hasil yang didapatkan dari fungsi ini adalah total kesalahan dalam gen tersebut yang dilihat dari segi dosen. Fungsi ini ada pada kelas FungsiFitness. Segmen Kode 4.4 menunjukkan penerapan dari pseudocode 3.4.

```
public Gen fungsiFitnessDosen(Gen tmpGen) {
        gen = tmpGen.getKelas();
        int errorDosen = 0;
        for (int i = 0; i < waktuV.size(); i++) {
            for (int j = 0; j < ruanganV.size() - 1; j++) {
                if (gen[j][i] != null) {
                    kelas = gen[j][i];
                    String d = kelas.getIdDosen();
                    for (int k = j + 1; k <
                    ruanganV.size(); k++) {
                        if (gen[k][i] != null) {
                            if
                            (d.equals(gen[k][i].getIdDo
                            sen())) {
                                errorDosen++;
                            }
                        }
                    }
                }
            }
        }
        totalErrorDosen += errorDosen;
        tmpGen.setKelas(gen);
        return tmpGen;
}
```

**Segmen Kode 4.4 Fungsi Fitness Dosen**

### 4.1.5 Fungsi Fitness Mahasiswa

Fungsi tujuan ini berfungsi untuk menghitung total kesalahan dari bentrokan mahasiswa yang mengikuti ujian dalam waktu bersamaan. Proses fungsi ini dimulai dengan menerima paramater gen untuk dievaluasi. Setiap mahasiswa yang mengambil mata kuliah pada index ruangan dan index waktu dalam gen tersebut akan dievaluasi apakah sama dengan mahasiswa yang mengambil mata kuliah pada index ruangan + 1. Apabila sama maka jumlah kesalahan akan bertambah satu. Hasil yang didapatkan adalah total kesalahan dalam gen tersebut dilihat dari segi mahasiswa. Fungsi ini ada pada *class* FungsiFitness. Segmen Kode 4.5 menunjukkan penerapan dari pseudocode 3.5.

### 4.1.6 Fungsi Fitness Sebaran

Fungsi ini berfungsi untuk menghitung total kesalahan dari sebaran merata, yaitu satu mahasiswa harus mengikuti maksimal dua hingga tiga ujian dalam satu hari tergantung bobot dari ujian yang diikuti. Fungsi proses ini dimulai dengan menerima parameter gen untuk dievaluasi. Setiap mahasiswa yang aktif mengambil mata kuliah pada index ruangan dan index waktu dalam gen tersebut akan dievaluasi apakah juga mengambil mata kuliah pada index waktu + 1. Jika sama maka bobot mahasiswa akan bertambah dan apabila jumlah bobot mahasiswa lebih dari enam maka jumlah kesalahan akan bertambah satu. Hasil yang didapatkan adalah total kesalahan dalam

gen tersebut dilihat dari segi sebaran merata. Fungsi ini berada pada *class* FungsiFitness. Segmen Kode 4.6 menunjukkan penerapan pseudocode 3.6.

```
public Gen fungsiFitnessMahasiswa(Gen tmpGen) {
      gen = tmpGen.getKelas();
      int errorMahasiswa = 0;
for (int i = 0; i < waktuV.size(); i++) {
      for (int j = 0; j < ruanganV.size() - 1; j++) {
            if (gen[j][i] != null) {
                  kelas = gen[j][i];
                  Vector nim1 = kelas.getIdMahasiswa();
                  for (int k = 0; k < nim1.size(); k++) {
                        DaftarMahasiswaKelas mahasiswa =
                        (DaftarMahasiswaKelas) nim1.get(k);
                        String cekMahasiswa1 =
                        mahasiswa.getIdMahasiswa();
                        for (int l = j + 1; l <
                              ruanganV.size(); l++){
                              if (gen[l][i] != null) {
                                    kelas2 = gen[l][i];
                                    Vector nim2 =
                                    kelas2.getIdMahasiswa();
                                    for (int m = 0; m <
                                    nim2.size(); m++) {
                                    DaftarMahasiswaKelas
                                    mahasiswa2 =
                                    (DaftarMahasiswaKelas)
                                    nim2.get(m);
                                    String cekMahasiswa2 =
                                    mahasiswa2.getIdMahasiswa()
                                    ;
                                    if
                                    (cekMahasiswa1.equals(cekMa
                                    hasiswa2)) {
                                          errorMahasiswa++;
                                    }
                                    }
                              }
                        }
                  }
            }
      }
 }
      totalErrorMahasiswa += errorMahasiswa;
      tmpGen.setKelas(gen);
      return tmpGen;
 }
```

**Segmen Kode 4.5 Fungsi Fitness Mahasiswa**

```
public Gen fungsiFitnessBobotUjianDalamSatuHari(Gen tmpGen) {
      gen = tmpGen.getKelas();
      int bobotUjian = 0;
      DaftarMahasiswaAktif mahasiswaAktif = new
      DaftarMahasiswaAktif();
      int counter = 0;
      int errorBobotUjian = 0;
      for (int i = 0; i < mahasiswaAktifV.size(); i++) {
            mahasiswaAktif = (DaftarMahasiswaAktif)
            mahasiswaAktifV.get(i);
            String cekMahasiswa =
            mahasiswaAktif.getIdMahasiswaAktif();
            for (int j = 0; j < ruanganV.size(); j++) {
                  for (int k = 0; k < waktuV.size(); k++) {
                        if (gen[j][k] != null && oldKelas !=
                        gen[j][k]) {
                        kelas = gen[j][k];
                        Vector nim = kelas.getIdMahasiswa();
                              for (int l = 0; l < nim.size();
                              l++) {
                                    DaftarMahasiswaKelas
                                    mahasiswaKelas =
                                    (DaftarMahasiswaKelas)
                                    nim.get(l);
                                    String cekMahasiswa2 =
                                    mahasiswaKelas.getIdMahasis
                                    wa();
                              if
                              (cekMahasiswa.equals(cekMahasiswa
                              2))  {
                              mahasiswaAktif.setBobotUjian(kela
                              s.getBobotUjian() +
                              mahasiswaAktif.getBobotUjian());
                              bobotUjian =
                              mahasiswaAktif.getBobotUjian();
                               if
                                (mahasiswaAktif.getBobotUjian()
                                > 6) {
                                      errorBobotUjian++;
                                 }
                              }
                              }
                        oldKelas = gen[j][k];
                              }
                  }
            }
      }
      totalErrorBobotUjian += errorBobotUjian;
      tmpGen.setKelas(gen);
      return tmpGen;
}
```

**Segmen Kode 4.6 Fungsi Fitness Sebaran Merata**

### 4.1.7 Fungsi Fitness Laboratorium

Fungsi ini berfungsi melakukan evaluasi terhadap gen apakah ujian dengan tipe ruangan laboratorium terdapat di satu hari secara berurutan. Proses yang pertama kali dilakukan adalah menerima parameter gen untuk dievaluasi apakah dalam index ruangan dan waktu ada kelas yang mempunyai tipe laboratorium. Jika ada maka fungsi ini akan mengembalikan nilai satu, jika tidak ada maka nilai yang dikembalikan adalah nol. Proses mencari kelas dengan tipe ruangan laboratorium tersebut berada dalam *class* FungsiFitness.

Setelah setiap gen dievaluasi maka jumlah dari nilai yang dikembalikan tersebut akan dijumlah. Jika kurang dari jumlah kelas dengan tipe ruangan laboratorium dibagi dengan jumlah kelas dalam satu hari kemudian hasilnya ditambah satu, maka hasil yang didapatkan adalah kromosom dengan nilai baik. Fungsi untuk mencari apakah sebuah kromosom tergolong baik dari segi penggunaan lanboratorium berada pada *class* Kromosom. Segmen Kode 4.7 merupakan penerapan dari pseudocode 3.7

```
public int fungsiFitnessLab(Gen tmpGen){
      for (int i = 0; i < ruanganV.size(); i++) {
            for (int j = 0; j < waktuV.size(); j++) {
                  if (gen[i][j] != null) {
                        kelas = gen[i][j];
                        if
                        (kelas.getTipeRuangan().equals("Lab")){
                              return 1;
                              break;
                        }
                  }
            }
      return 0;
}


public void cekKualitasSoftContraint(int kelasDalamSatuHari){
      int kualitas = 0;
      int batasLab = 0;
      int kelasLab = kelas.getJumlahKelasLab();
      batasLab = kelasLab / kelasDalamSatuHari + 1;
      for (int i = 0; i < vectorOfGen.size(); i++) {
            gen = (Gen) vectorOfGen.get(i);
            kualitas2 += fungsiFitnessLab(gen);
      }
      if (kualitas2 <= batasLab) {
            kualitas = 1;
      } else {
            kualitas = 0;
      }
}
```

**Segmen Kode 4.7 Fungsi Fitness Lab**

### 4.1.8 Fungsi Seleksi

Fungsi ini dimulai dengan menerima paramater vector populasi yang berisi kumpulan kromosom untuk diambil total kesalahan dari setiap kromosom dalam populasi tersebut. Setelah total kesalahan didapatkan, dilakukan perhitungan total total nilai distribusi (1 dibagi total error) dilanjutkan dengan menghitung nilai probabilitas kumulatif dengan cara membagi nilai distribusi masing-masing kromosom dengan total nilai distribusi lalu dijumlah dengan nilai probabilitas kromosom sebelumnya. Setelah itu dilakukan pengambilan sebuah

angka secara acak antara nol sampai dengan satu. Apabila angka yang diambil lebih kecil daripada nilai probabilitas sebuah kromosom maka kromosom tersebut yang akan dijadikan kromosom induk dalam operasi persilangan. Hasil dari fungsi ini adalah vector Kromosom yang berisi dua kromosomParent yang akan dijadikan induk dalam operasi persilangan. Fungsi ini ada pada *class* Seleksi. Segmen Kode 4.8 merupakan penerapan dari pseudocode 3.8.

### 4.1.9 Fungsi Persilangan

Fungsi ini mulai dijalankan dengan mengambil parameter vector kromosom yang berisi dua kromosomParent hasil proses seleksi. Dari masing-masing kromosomParent akan ditentukan satu titik potong terhadap index gen secara acak sebagai acuan untuk melakukan pertukaran gen dari masing-masing kromosom. Dari titik potong tersebut, gen dari kromosomParent akan diambil sejumlah mulai dari gen pada titik potong sampai jumlah gen pada kromosomParent. Kemudian untuk masing-masing kromosomParent akan ditukar gen satu sama lain sesuai dengan titik potong yang ditentukan. Hasil yang didapatkan adalah dua kromosomAnak yang disimpan dalam bentuk vector. Segmen Kode 4.9 menunjukkan penerapan pseudocode 3.9

```
public Vector seleksi(Vector tmpVector) {
      Kromosom[] kromosom = new Kromosom[tmpVector.size()];
      Kromosom[] kromosomParent = new Kromosom[2];
      Double[] probabilitas;
      totalNilaiDistribusi = 0;
      Vector kromosomParentV = new Vector();
      nilaiDistribusiTiapKromosom = new
      double[tmpVector.size()];
      for (int i = 0; i < tmpVector.size(); i++) {
            kromosom[i] = (Kromosom) tmpVector.get(i);
            double totalError = kromosom[i].getTotalError();
            nilaiDistribusiTiapKromosom[i] = 1 / totalError;
            totalNilaiDistribusi +=
            nilaiDistribusiTiapKromosom[i];
      }
      probabilitas = new double[tmpVector.size()];
      for (int i = 0; i < probabilitas.length; i++) {
            probabilitas[i] = 0;
      }
      for (int i = 0; i < probabilitas.length; i++) {
            if (i == 0) {
                  probabilitas[i] =
                  nilaiDistribusiTiapKromosom[i] /
                  totalNilaiDistribusi;
            } else {
                  probabilitas[i] =
                  nilaiDistribusiTiapKromosom[i] /
                  totalNilaiDistribusi + probabilitas[i - 1];
            }
      }
      double x1 = (double) (Math.random() * 1);
      for (int j = 0; j < probabilitas.length; j++) {
            if (x1 < probabilitas[j]) {
                  kromosomParent[0] = (Kromosom)
                  tmpVector.get(j);
                  kromosomParentV.add(kromosomParent[0]);
                  break;
            }
      }
      double x2 = (double) (Math.random() * 1);
      for (int j = 0; j < probabilitas.length; j++) {
            if (x2 < probabilitas[j] && j != taken) {
                  kromosomParent[1] = (Kromosom)
                  tmpVector.get(j);
                  kromosomParentV.add(kromosomParent[1]);
                  break;
            }
      }
      return kromosomParentV;
`
```

**Segmen Kode 4.8 Fungsi Seleksi**

```
public Vector operateCrossover(Vector tmpVector) {
      seleksiV = tmpVector;
      Kromosom[] kromosomParent = new
      Kromosom[tmpVector.getSize()];
      kromosomParent[0] = new Kromosom();
      kromosomParent[1] = new Kromosom();
      kromosomParent[0] = (Kromosom) seleksiV.get(0);
      kromosomParent[1] = (Kromosom) seleksiV.get(1);
      Vector kromosomAnakV = new Vector();
      kromosomAnakV.removeAllElements();
      Kromosom[] kromosomAnak = new Kromosom[2];
      kromosomAnak[0] = new Kromosom();
      kromosomAnak[0].setGen(kromosomParent[0].getGen());
      kromosomAnak[1] = new Kromosom();
      kromosomAnak[1].setGen(kromosomParent[1].getGen());
      int idx = (int) (Math.random() * 8);
      for (int i = idx; i < 8; i++) {
            Gen genTmp1 = kromosomAnak[0].getGenAt(i);
            kromosomAnak[0].setGenAt(i,
            kromosomAnak[1].getGenAt(i));
            kromosomAnak[1].setGenAt(i, genTmp1);
      }
      kromosomAnakV.add(kromosomAnak[0]);
      kromosomAnakV.add(kromosomAnak[1]);
      return kromosomAnakV;
}
```

**Segmen Kode 4.9 Fungsi Persilangan**

## 4.2 Pengujian Perangkat Lunak

Bagian ini menjelaskan tentang hasil dari kegiatan pengujian pada perangkat lunak pembuatan jadwal UAS ini. Proses pengujian dilakukan untuk mendapatkan hasil paling optimal dari pembuatan jadwal UAS. Hasil yang paling optimal adalah ketika jumlah bentrokan yang didapatkan paling minimal. Untuk mendapatkan jumlah bentrokan tersebut maka dilakukan proses pengujian dengan variabel jumlah populasi dan generasi.

### 4.2.1 Tujuan Pengujian

Tujuan dari pengujian ini adalah mendapatkan kromosom terbaik, baik dari segi batasan keras maupun batasan lunak. Untuk mencapai tujuan tersebut maka dilakukan proses pengujian dengan

variabel jumlah populasi dan generasi yang akan diubah sebanyak batas maksimal jumlah pengujian dalam perangkat lunak ini.

### 4.2.2 Lingkungan Pengujian

Dalam melakukan pengujian, perangkat keras yang digunakan adalah notebook Acer dengan tipe Aspire 5920G dengan spesifikasi sebagai berikut:

- Prosessor : Intel Core 2 Duo T7500 2,2 GHz
- VGA : 512 MB
- RAM : 2 GB
- Harddisk : 250 GB
- Sistem Operasi : Windows Vista
- Bahasa Pemrograman : Java

Data yang digunakan untuk melakukan pengujian ini diambil dari Biro Administrasi dan Akademik yaitu kelas dan mahasiswa aktif pada periode akademik semester genap tahun ajaran 2010/2011.

### 4.2.3 Skenario Pengujian

Skenario ini digunakan untuk menguji berapa jumlah kromosom yang memenuhi batasan keras dan batasan lunak dengan variabel jumlah populasi dan generasi yang diubah sampai batas maksimal yang ditentukan dalam pengujian perangkat lunak ini. Setiap perubahan akan dilakukan sebanyak 20 kali eksekusi.

### 4.2.4 Hasil Pengujian

Hasil pengujian perangkat lunak dapat dilihat pada tabel 4.1, 4.2, 4.3, dan 4.4 dengan index baris merupakan jumlah generasi dan index kolom merupakan jumlah populasi. Tabel 4.1 merupakan hasil pengujian batasan keras dengan jumlah populasi mulai dari 10 sampai dengan 100 dan jumlah generasi mulai dari 10 sampai dengan 200. Tabel 4.2 merupakan hasil pengujian batasan keras dengan jumlah populasi mulai dari 110 sampai dengan 200 dan jumlah generasi mulai dari 10 sampai dengan 200. Tabel 4.3 merupakan hasil pengujian batasan lunak dengan jumlah populasi mulai dari 10 sampai dengan 100 dan jumlah generasi mulai dari 10 sampai dengan 200. Tabel 4.4 merupakan hasil pengujian batasan lunak dengan jumlah populasi mulai dari 110 sampai dengan 200 dan jumlah generasi mulai dari 10 sampai dengan 200.

**Tabel 4.1 Hasil Pengujian Batasan Keras Populasi 10 Sampai Dengan 100**

| | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 0 | 0 | 1 | 1 | 2 | 2 | 4 | 3 | 4 | 5 |
| **20** | 1 | 1 | 0 | 0 | 1 | 2 | 4 | 4 | 3 | 4 |
| **30** | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 4 | 4 | 6 |
| **40** | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 4 |
| **50** | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 4 | 6 | 5 |
| **60** | 1 | 2 | 3 | 3 | 4 | 5 | 4 | 5 | 6 | 6 |
| **70** | 2 | 3 | 4 | 5 | 4 | 5 | 6 | 6 | 7 | 9 |
| **80** | 3 | 3 | 4 | 5 | 5 | 4 | 6 | 7 | 7 | 6 |
| **90** | 3 | 3 | 4 | 6 | 5 | 4 | 6 | 8 | 7 | 8 |
| **100** | 4 | 3 | 4 | 5 | 6 | 5 | 7 | 6 | 7 | 7 |
| **110** | 4 | 5 | 4 | 6 | 5 | 7 | 6 | 8 | 8 | 7 |
| **120** | 5 | 5 | 6 | 7 | 6 | 8 | 7 | 7 | 9 | 8 |
| **130** | 4 | 4 | 5 | 6 | 6 | 7 | 7 | 6 | 8 | 9 |
| **140** | 5 | 4 | 4 | 5 | 6 | 7 | 7 | 8 | 8 | 6 |
| **150** | 4 | 5 | 5 | 6 | 5 | 7 | 6 | 6 | 8 | 8 |
| **160** | 4 | 4 | 6 | 5 | 4 | 6 | 8 | 7 | 7 | 9 |
| **170** | 6 | 4 | 5 | 6 | 7 | 8 | 7 | 9 | 10 | 10 |
| **180** | 5 | 6 | 7 | 8 | 9 | 8 | 9 | 11 | 10 | 12 |
| **190** | 7 | 5 | 6 | 5 | 7 | 8 | 10 | 11 | 12 | 11 |
| **200** | 7 | 6 | 7 | 9 | 8 | 8 | 9 | 10 | 11 | 10 |

**Tabel 4.2 Hasil Pengujian Batasan Keras Populasi 110 Sampai Dengan 200**

| | 110 | 120 | 130 | 140 | 150 | 160 | 170 | 180 | 190 | 200 |
|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 5 | 6 | 5 | 6 | 7 | 8 | 7 | 9 | 9 | 10 |
| **20** | 5 | 6 | 5 | 6 | 7 | 7 | 8 | 8 | 7 | 9 |
| **30** | 5 | 6 | 8 | 9 | 7 | 8 | 9 | 10 | 10 | 11 |
| **40** | 6 | 7 | 8 | 8 | 10 | 8 | 19 | 9 | 11 | 11 |
| **50** | 4 | 6 | 7 | 8 | 10 | 9 | 11 | 10 | 12 | 12 |
| **60** | 7 | 6 | 8 | 9 | 10 | 11 | 10 | 13 | 12 | 11 |
| **70** | 8 | 9 | 9 | 11 | 12 | 10 | 11 | 12 | 13 | 13 |
| **80** | 7 | 8 | 8 | 10 | 11 | 9 | 12 | 14 | 15 | 14 |
| **90** | 9 | 9 | 11 | 10 | 11 | 12 | 14 | 15 | 13 | 15 |
| **100** | 9 | 8 | 11 | 12 | 14 | 13 | 13 | 14 | 15 | 14 |
| **110** | 9 | 10 | 12 | 11 | 11 | 15 | 14 | 13 | 15 | 16 |
| **120** | 8 | 9 | 11 | 11 | 12 | 12 | 14 | 15 | 13 | 14 |
| **130** | 9 | 10 | 11 | 10 | 12 | 12 | 14 | 14 | 13 | 15 |
| **140** | 9 | 11 | 10 | 10 | 13 | 13 | 14 | 12 | 14 | 15 |
| **150** | 9 | 9 | 11 | 11 | 10 | 12 | 13 | 14 | 15 | 16 |
| **160** | 11 | 10 | 10 | 11 | 14 | 13 | 15 | 15 | 14 | 14 |
| **170** | 12 | 11 | 10 | 12 | 13 | 14 | 14 | 15 | 12 | 15 |
| **180** | 12 | 15 | 12 | 13 | 12 | 14 | 15 | 14 | 15 | 16 |
| **190** | 10 | 12 | 13 | 13 | 14 | 12 | 14 | 15 | 16 | 16 |
| **200** | 10 | 11 | 12 | 13 | 14 | 14 | 16 | 15 | 16 | 17 |

**Tabel 4.3 Hasil Pengujian Batasan Lunak Populasi 10 Sampai Dengan 100**

| | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **20** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **30** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **40** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **50** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **60** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **70** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **80** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **90** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **100** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **110** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **120** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **130** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **140** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **150** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **160** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **170** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **180** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **190** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **200** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Tabel 4.4 Hasil Pengujian Batasan Lunak Populasi 110 Sampai Dengan 200**

| | 110 | 120 | 130 | 140 | 150 | 160 | 170 | 180 | 190 | 200 |
|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **20** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **30** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **40** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **50** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **60** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **70** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **80** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **90** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **100** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **110** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **120** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **130** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **140** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **150** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **160** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| **170** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 |
| **180** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 |
| **190** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 |
| **200** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 4 |

### 4.2.5 Analisa Pengujian

Dari hasil pengujian perangkat lunak dengan 20 kali eksekusi, maka persentase untuk mendapatkan kromosom yang memenuhi batasan keras paling besar adalah 85 % dengan jumlah populasi 200 dan jumlah generasi 200. Dari 20 kali eksekusi maka persentase untuk mendapatkan kromosom yang memenuhi batasan lunak perlu melakukan 20 % dengan jumlah populasi 200 dan jumlah generasi 200.

### 4.2.6 Skenario Pengujian Tambahan

Skenario ini digunakan untuk mendapatkan berapa jumlah kromosom yang memenuhi batasan keras dan batasan lunak dengan variabel jumlah mahasiswa sebanyak 320 orang dengan jumlah populasi mulai dari 100 sampai 500 dan jumlah generasi mulai dari 100 dan 200 dan berapa lama untuk mendapatkan hasil tersebut.

### 4.2.7 Hasil Pengujian Skenario Tambahan

Hasil pengujian skenario tambahan dapat dilihat pada tabel 4.5. Tabel 4.5 merupakan hasil pengujian berapa lama untuk mendapatkan kromosom yang memenuhi  yang memenuhi batasan keras dan batasan lunak dengan variabel jumlah mahasiswa sebanyak 320 orang dengan jumlah mulai dari 100 sampai 500 dan jumlah generasi mulai dari 100 dan 200. Tabel 4.6 menunjukkan berapa lama untuk mendapatkan hasil tersebut.

**Tabel 4.5 Hasil Pengujian Batasan Keras dan Batasan Lunak Dengan Populasi Dan Generasi 200 dan 300**

| | 100 | 200 | 300 | 400 | 500 |
|---|---|---|---|---|---|
| **100** | 19 | 19 | 20 | 20 | 20 |
| **200** | 18 | 20 | 20 | 20 | 20 |

**Tabel 4.6 Hasil Pengujian Waktu Dengan Populasi Dan Generasi 200 dan 300**

| | 100 | 200 | 300 | 400 | 500 |
|---|---|---|---|---|---|
| **100** | 150 detik | 145 detik | 141 detik | 142 detik | 135 detik |
| **200** | 146 detik | 139 detik | 140 detik | 137 detik | 134 detik |

### 4.2.8 Analisa Pengujian Skenario Tambahan

Dari hasil pengujian perangkat lunak dengan 20 kali eksekusi, maka persentase untuk mendapatkan kromosom yang memenuhi batasan keras dan batasan lunak adalah sebesar 100 % dengan jumlah populasi minimal 200 dan jumlah generasi minimal 200 dan waktu yang digunakan untuk mendapatkan hasil tersebut adalah 139 detik. Variabel yang mengalami perubahan adalah mahasiswa dengan jumlah 320 orang.

### 4.2.9 Kesalahan Perangkat Lunak

Dari hasil pengujian perangkat lunak didapatkan beberapa kesalahan, yaitu sebagai berikut:

1. Terdapat kemungkinan muncul kesalahan berupa *java.lang.ArrayIndexOutOfBondsException: Array index out of range: 0* ketika perangkat lunak ini dijalankan. Kesalahan ini muncul karena ada sebuah matrix yang ukurannya melebihi ukuran

yang seharusnya ditetapkan, dan seharusnya hal tersebut tidak dapat terjadi karena ukuran matrix sudah ditentukan.

2. Ukuran skalabilitas memiliki batas tertentu, yaitu jumlah populasi maksimal 800 dengan jumlah mahasiswa maksimal 80. Jika melebihi batas tersebut akan muncul kesalahan berupa *java.lang.OutOfMemoryError: Java heap space*.

## 4.3. Aspek Entreprenurship

Sub bab ini menjelaskan aspek enrepreneurship apa saja yang ada dalam perangkat lunak ini. Dari teori entrepreneurship yang ada, di dalam pembuatan perangkat lunak ini ada dua aspek penting yang menjadi keunggulan utama dari perangkat lunak ini, yaitu antara lain:

### 4.3.1 *Indentifying And Recognizing Oppurtinities*

Ide dari pembuatan jadwal UAS ini dibuat dengan tujuan menyelesaikan sebuah permasalahan yang ada. Melalui indentifikasi permasalahan yang ada dalam pembuatan jadwal UAS di Program Studi Teknik Informatika Universitas Ciputra maka hal tersebut dapat dijadikan sebuah peluang yang baik. Dari hasil pembuatan perangkat lunak ini maka tujuan yang diharapkan adalah mampu menyelesaikan permasalahan. Untuk dapat menyelesaikan permasalahan tersebut adalah dengan cara membuat sebuah perangkat lunak yang dibuat dengan kreatif dan inovasi agar dapat digunakan untuk mendapatkan solusi yang paling optimal.

Selain dalam penjadwalan pada institut pendidikan, pengembangan perangkat lunak ini dapat dilakukan sampai pada pembuatan jadwal yang ada pada industri-industri. Algoritma genetika yang digunakan dalam perangkat lunak ini dapat menyelesaikan permasalahan yang cukup kompleks dimana banyak variabel yang diperhitungkan. Dalam mesin-mesin yang digunakan di pabrik-pabrik, ada banyak variabel yang harus diperhitungkan, contohnya adalah waktu maksimal operasional mesin dalam satu hari, biaya operasional setiap mesin dalam satu hari, dan sebagainya agar dapat dioperasikan dengan efisien dan optimal. Dengan bantuan pengembangan perangkat lunak ini, maka semua variabel tersebut akan diperhitungkan oleh algoritma genetika sehingga jadwal operasional mesin yang dihasilkan adalah yang paling optimal.

### 4.3.2 *Market Sensitivity*

Pembuatan perangkat lunak ini telah melalui proses analisa terhadap permasalahan dalam penyusunan jadwal UAS di Program Studi Teknik Infotmatika Universitas Ciputra dan sudah diterima oleh Program Studi Teknik Informatika Universitas Ciputra. Pengembangan dari perangkat lunak ini tidak hanya sebatas dalam ruang lingkup di Universitas Ciputra saja, atau pada institut pendidikan dengan masalah penjadwalannya tetapi dapat dikembangkan dalam pembuatan jadwal dalam bidang yang lain. Pengembangan perangkat lunak ini dapat digunakan pada masalah penjadwalan yang ada pada mesin-mesin

industri. Setiap mesin akan dijalankan sesuai dengan jadwal yang dibuat dari pengembangan perangkat lunak ini.

Perangkat lunak ini pada dasarnya dapat dijalankan pada sebuah komputer dengan spesifikasi yang tidak terlalu tinggi. Biaya yang digunakan dari penggunaan perangkat lunak ini relatif kecil karena yang dibutuhkan tidak banyak. Dalam pengembangannya, perangkat lunak ini dapat dijual kepada perusahaan-perusahaan yang membutuhkan solusi penyelesaian dalam penjadwalan, tidak hanya penjadwalan kerja manusia, namun juga penjadwalan mesin. Dalam penjadwalan mesin, variabel yang diperhitungkan cukup banyak sehingga membutuhkan waktu kalau diselesaikan dengan manual. Pengembangan perangkat lunak ini mampu melakukan penyelesaian tersebut sehingga nilai jual yang dimiliki semakin tinggi.

### 4.3.3 Analisis *Feasibility*

Berikut ini adalah analisis kelayakan perangkat lunak ini, yaitu antara lain:

**a. Rencana Keuangan**

Berikut ini adalah perkiraan dana yang dibutuhkan untuk membuat perangkat lunak ini, yaitu:

- Rp 6.000.000,00, untuk pembelian komputer dalam pembuatan perangkat lunak.
- Rp 2.000.000,00, untuk biaya lain-lain, seperti listrik dan kebutuhan sehari-hari

**b. Struktur Organisasi**

Untuk pembuatan perangkat lunak ini, hanya dibutuhkan satu orang saja, yaitu bagian *developer* yang berperan sebagai pembuat perangkat lunak.

**c. Strategi Pemasaran**

Strategi pemasaran yang dilakukan untuk menjual perangkat lunak ini dan pengembangannya, yaitu antara lain:

1. Segmentasi

   Segmentasi dari perangkat lunak ini adalah institut pendidikan di Surabaya yang membutuhkan solusi dalam permasalahan penjadwalan. Segmentasi untuk pengembangan perangkat lunak ini adalah pabrik-pabrik di Surabaya yang memiliki penjadwalan mesin.

2. *Targetting*

   Target pembeli per bulan dari perangkat lunak ini adalah dua institut pendidikan dan lima pabrik di Surabaya.

3. *Positioning*

   Perangkat lunak ini dibuat sebagai solusi penyelesaian yang paling efisien dari permasalahan penjadwalan pada institut pendidikan serta penjadwalan mesin pabrik di Surabaya.

**d. Eksekusi Bisnis**

Berikut ini adalah proses yang dilakukan untuk menjual perangkat lunak ini dan pengembangannya ke pasar, yaitu antara lain:

1. *Price*

Untuk strategi harga dari perangkat lunak dan pengembangannya, maka akan dilakukan penjualan dalam tiga paket. Untuk paket yang pertama diberikan dengan gratis sebagai contoh dengan fitur-fitur penjadwalan standar yang ada pada setiap institut pendidikan dan pabrik dan memiliki masa aktif 30 hari. Paket yang kedua dijual dengan harga Rp 4.000.000,00 untuk institut pendidikan dan Rp 12.000.000,00 untuk pabrik dengan masa aktif satu tahun dan garansi tiga bulan. Kelebihan paket yang kedua ini adalah penambahan lima fitur penjadwalan sesuai dengan kesepakatan. Paket ketiga dijual dengan harga Rp 6.000.000,00 untuk institut pendidikan dan Rp 22.000.000,00 untuk pabrik dengan masa akfif dua tahun dan garansi sepuluh bulan. Kelebihan paket yang ketiga ini adalah penambahan sepuluh fitur penjadwalan sesuai dengan kesepakatan. Garansi dari paket kedua dan ketiga memiliki kelebihan yaitu biaya gratis selama masa garansi untuk pelatihan sumber daya manusia dalam menjalankan perangkat lunak dan biaya perbaikan perangkat lunak.

2. *Product*

Dari segi produk, kelebihan dari perangkat lunak ni adalah dapat menyelesaikan permasalahan dalam masalah penjadwalan baik di institut pendidikan maupun pada pabrik. Kedua permasalahan tersebut mengandung banyak variabel yang harus diperhitungkan sehingga cukup sulit apabila harus diselesaikan dengan metode komvensional. Selain itu perangkat lunak ini dibungkus dalam paket-paket yang sesuai dengan kebutuhan dan garansi perangkat lunak.

3. *Place*

Untuk melakukan proses distribusi, dilakukan dengan mengirimkan perangkat lunak langsung kepada pembeli.

4. *Promotion*

Untuk promosi menggunakan kemampuan *word power*, yaitu penyebaran informasi dari mulut ke mulut. Selain itu dapat digunakan media promosi, seperti blog atau dimasukkan ke forum-forum yang berhubungan dengan pendidikan dan perindustrian.