ANALISIS, DESAIN, DAN IMPLEMENTASI *GENETIC ALGORITHM* (GA)

Oleh kelompok 15:
Alif Adwitiya Pratama
Sabrina Adinda Sari
Wana Ardilah Iwan

Bandung, Telkom University
alifadwitiyapratama@student.telkomuniversity.ac.id
sabrinaadindasari@student.telkomuniversity.ac.id
wanaardilah@student.telkomuniversity.ac.id

ABSTRAK

Dalam melakukan optimasi suatu fungsi dapat diselesaikan menggunakan algoritma genetika, dimana permasalahan yang terkait mencari nilai optimum maupun minimum dari suatu fungsi akan menjadi lebih mudah yaitu dengan cara mengkombinasikan solusi-solusi (kromosom) untuk menghasilkan solusi baru dengan menggunakan operator genetika(seleksi, crossover dan mutasi). Pada laporan ini, akan dicari nilai optimum dari fungsi $h(x,y) = (cos\ x^2 \ * \ sin\ y^2) + (x+y)$ dengan algoritma genetika menggunakan populasi awal sebanyak 2000 dan akan menghasilkan anak sebanyak 1000 yang nantinya akan diregenerasi sampai generasi ke 300. Untuk probabilitas operasi genetik yang digunakan adalah 0.800 sebagai *cross rate* (p_c)dan 0.3 sebagai *mutasi rate* (p_m). Solusi optimum terbaik yang didapat dari fungsi fitness tersebut adalah x = 0.87636342000000000407 dan y = 0.99934294000000001290 dengan nilai fitness 2.48045912824672.

***Kata Kunci** : Algoritma Genetika, fungsi, Solusi*

PENDAHULUAN

Genetic Algorithm (GA) pertama kali dikembangkan oleh John Holland pada tahun 1970-an di New York, Amerika Serikat. Dia beserta murid-murid dan teman kerjanya menghasilkan buku berjudul "Adaption in Natural and Artificial Systems" pada tahun 1975. (Wikipedia) *Genetic Algorithm* (GA) adalah bagian dari *Evolutionary Algorithm* yaitu suatu algoritma yang mencontoh proses evolusi alami dimana konsep utamanya adalah individu-individu yang paling unggul akan bertahan hidup, sedangkan individu-individu yang lemah akan punah. Keunggulan individu-individu ini diuji melalui suatu fungsi yang dikenal sebagai fitness function. Fitness dalam GA didefinisikan sebagai gambaran kelayakan suatu solusi terhadap suatu permasalahan. Fitness Function akan menghasilkan suatu nilai fitness value yang akan menjadi referensi untuk proses GA selanjutnya. Individu adalah, menyatakan satu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat sedangkan nilai fitness adalah nilai yang menyatakan baik tidaknya suatu solusi (individu), yang sebagai acuan dalam mencapai nilai optimal dalam *genetic algorithm*.

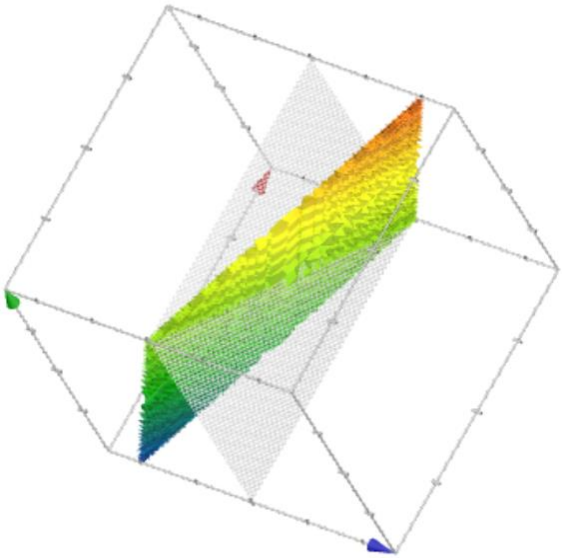
Genetic Algorithm (GA) diterapkan sebagai simulasi pada komputer yang terdapat sebuah populasi atau biasa disebut dengan kromosom dari solusi-solusi calon induk pada sebuah masalah optimasi yang kemudian akan dikembangkan menjadi solusi-solusi yang dilambangkan dalam biner sebagai string ‘1’ dan ‘0’, walaupun dimungkinkan juga penggunaan penyandian (*encoding*) yang berbeda. Evolusi dimulai dari populasi individual acak lengkap dan akan terjadi dalam sebuah gen-gen. Dalam gen, keseluruhan populasi akan dievaluasi kemudian dipilih dari populasi sekarang (*current*) tersebut berdasarkan kemampuan mereka

kemudian dimodifikasi melalui mutasi ataupun rekombinasi untuk menjadi bentuk populasi baru yang menjadi populasi sekarang pada iterasi algoritma berikutnya.

Trigonometri merupakan suatu cabang ilmu matematika yang mempelajari mengenai sudut sisi, dan perbandingan antara sudut terhadap sisi. Salah satu materi yang diajarkan di SMA kelas X adalah materi Trigonometri. Trigonometri berasal dari bahasa Yunani yang berarti pengukuran segitiga. Trigonometri merupakan bagian dari matematika yang mempelajari hubungan antara sisi-sisi dan sudut-sudut pada suatu segitiga (Marwanta, 2009:144). Menurut Rusgianto M.S (2012: 8-9) trigonometri merupakan relasi atau hubungan dari sinus, cosinus, tangen, cotangen, secan, cosecan yang telah memenuhi prasyarat tertentu. Terkadang terdapat suatu fungsi dalam mencari nilai optimasi range dari suatu fungsi sulit untuk dihitung secara manual karena memiliki fungsi dengan syarat yang terbilang rumit sehingga untuk memudahkan penyelesaian dalam kasus ini kita dapat menyelesaikan persoalan mencari solusi optimum dari suatu fungsi dengan *Genetic Algorithm* (GA)

STRATEGI KASUS

Menemukan nilai maksimum dari fungsi : Dengan batasan $-1 \leq x \leq 2, dan\ -1 \leq y \leq 1$



Gambar 1.1 Plot fungsi dari $h(x,y) = (cos\ x^2\ \ * sin\ y^2\ ) + (x+y)$

Berikut strategi yang digunakan untuk menyelesaikan fungsi tersebut dengan *Genetic Algorithm* :

1. Representasi Chromosome

   Dalam hal ini, tiap individu terdiri dari kromosom x dan y,dimana untuk membuat tiap kromosom pada individu akan dibangkitkan angka float secara random yang memenuhi syarat melalui fungsi generate individu, dimana untuk untuk individu yang valid nilai x dan y adalah -1$\leq x \leq$ 2 dan -1$\leq y \leq$ 1 kemudian nilai float tersebut akan di encoding dengan fungsi encode yang akan menghasilkan bentuk representasi ieee 754 float point 64 bit sehingga dapat diterapkan fungsi mutasi dan crossover yang telah dibuat. Sehingga kromosom pada satu individu terdiri dari 64 gen kromosom x dan 64 gen kromosom y.

```
[ ]
    #fungsi fitness utama
    def fitness(x,y):
      if -1 <= x <= 2 and -1 <= y <= 1:
        return (math.cos(x**2)*math.sin(y**2))+(x+y)
      else:
        return np.nan


  #dibuat untuk melakukan iterasi dan memasukannya ke fungsi fitness utama
  def getFitness(value):
    hasil = np.empty((0, 1), float)
    for ind in value:
        hasil=np.insert(hasil,0,fitness(ind[0],ind[1]),axis=0)
    hasil=np.flip(hasil)

    return hasil
```

Gambar 1.2 Fungsi Fitness dari Program GA fungsi dari $h(x,y) = (cos\, x^2 \quad * sin\, y^2\,) + (x+y)$

Encode dan Decode

```
#fungsi encode utama
def encode(value):    # float -> bin
    getBin = lambda x: x > 0 and str(bin(x))[2:] or "-" + str(bin(x))[3:]
    val = struct.unpack('Q', struct.pack('d', value))[0]
    ans=str(getBin(val))
    if len(ans)==62:
      return"00"+ans
    if len(ans)==63:
      return"0"+ans
    return ans

#fungsi decode utama
def decode(value):     #bin -> float
    if value[0]=='1':
      hx = hex(int(value[1:], 2))
      return -1*(struct.unpack("d", struct.pack("q", int(hx, 16)))[0])
    else:
      hx = hex(int(value, 2))
      return (struct.unpack("d", struct.pack("q", int(hx, 16)))[0])
```

```
#melakukan iterasi dan memasukannya kedalam fungsi encode utama
def encodeList(value):
  value=value.tolist()

  for i in range(0,len(value)):
    for j in range(0,len(value[i])):
      value[i][j]=encode(value[i][j])
  return value

#melakukan iterasi dan memasukannya kedalam fungsi decode utama
def decodeList(value):
  result = np.empty((0, 2), float)
  for i in range(0,len(value)):
    for j in range(0,len(value[i])):
      result=np.insert(result,0,decode(value[i][j]),axis=0)
  return result
```

Gambar 1.3 Encode dan decode Program GA fungsi dari $h(x, y) = (cos\ x^2 \ * \ sin\ y^2\ ) + (x + y)$

2. Populasi

Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus evolusi, dalam kasus ini kami menggunakan jumlah populasi sebanyak 2000 sebagai populasi awal yang dihasilkan melalui fungsi generate individu, dengan maksud bahwa semakin banyak individu random yang diciptakan maka anak yang dihasilkan akan beragam pula sehingga mungkin saja solusi terbaik dapat dihasilkan pada generasi pertama. Jika nilai parameter ukuran populasi semakin besar maka akan meningkatkan kemampuan eksplorasi algoritma genetika untuk mencari solusi yang terbaik.

▾ Generate Individu

```
[ ] def generateIndividu(banyak):                                    #input = banyak individu
      population = np.empty((0, 2), float)                           #output = array populasi yang berisi list 2 float

      for i in range(0,banyak):
        population=np.insert(population,0,[round(random.uniform(-1.0, 2.0), 8),round(random.uniform(-1.0, 1.0), 8)], axis=0)
      return population
```

Gambar 1.4 Generate individu fungsi dari $h(x, y) = (cos\ x^2 \ * \ sin\ y^2\ ) + (x + y)$

3. Pemilihan Orang Tua

Terdapat populasi sebanyak 2000 dan anak yang diinginkan sebanyak 1000. Untuk pemilihan orang tua menggunakan roulette wheel dimana setiap individu akan dicari nilai dari fungsi fitnessnya dari individu ke 1 sampai dengan individu ke n (banyaknya populasi). Kemudian akan dicari nilai probabilitas untuk masing-masing individu, dimana rumus untuk mencari probabilitas yaitu nilai fitness individu tersebut dibagi dengan nilai total fitness seluruh populasi. Setelah itu akan disesuaikan proporsi dari tiap individu di *roulette wheel*, dan akan bangkitkan angka random, nanti

angka random yang sesuai dengan proporsi tiap individu di *roulette wheel* tersebut yang terpilih menjadi orang tua.

```
[ ]  def makeProb(fit):
         prob=np.empty((0, 1), float)
         for v in fit:
           temp=v/np.sum(fit)
           prob=np.insert(prob,0,temp,axis=0)
         prob=np.flip(prob)
         return prob

     def seleksiOrangtua(populasi,jumAnak):
         fit=getFitness(populasi)
         prob=makeProb(fit)
         diambil = np.empty((0, 2), float)
         for i in range(0,jumAnak):
           diambil=np.insert(diambil,0,choices(populasi, prob)[0],axis=0)

         return diambil
```

Gambar 1.5 Seleksi Program GA fungsi dari $h(x,y) = (cos\ x^2 \ * sin\ y^2) + (x+y)$

4. *Crossover* dan Mutasi

Crossover yang digunakan adalah *single-point crossover* atau *1-point crossover* yaitu teknik dimana 2 individu yang terpilih melakukan rekombinasi kromosom, akan dipilih lokasi secara random dari gen ke 1 sampai gen ke n (panjangnya kromosom pada satu individu) sebagai titik potong persilangan kromosom dari kedua individu tersebut. Titik potongnya ini dinamakan titik pivot atau titik persilangan. Pada potongan ini, informasi genetik ke kiri (atau kanan) dari titik tersebut bertukar antara dua kromosom induk untuk menghasilkan dua kromosom keturunan (anak-anak). Dan begitu seterusnya untuk setiap pasang individu yang hendak melakukan *crossover*. Namun belum tentu semua pasang individu ini dapat melakukan *crossover*, sebelum terjadi masuk ke fungsi *crossover* terlebih dahulu bangkitkan angka random, jika angka yang dibangkitkan kurang dari sama dengan *cross rate*(p_c) maka kedua individu tersebut dapat melakukan *crossover*, sedangkan jika angka random yang dibangkitkan lebih dari *cross rate*(p_c) maka tidak dapat terjadi *crossover*.

```
[ ]  def mutation(value,rate):
       for i in range(3,len(value)):
         if random.random()<=rate:
           if value[i]==0:
             value=value[:i]+"1"+value[i+1:]
           else:
             value=value[:i]+"0"+value[i+1:]
       return value


     def buatAnak(a,b,crossRate,mutationRate):
         crosspoint=randint(1,len(a))
         a=mutation(a,mutationRate)
         b=mutation(b,mutationRate)

         #cross over
         if random.random() <= crossRate:
           temp=a
           a = a[:crosspoint]+b[crosspoint:]
           b = b[:crosspoint]+temp[crosspoint:]
         return a,b
```

Gambar 1.6 Crossover & MutationProgram GA fungsi dari $h(x,y) = (cos\ x^2 \ * sin\ y^2) + (x+y)$

Untuk mutasi, *mutation rate*(p_m) yang digunakan adalah 0.3. Untuk cara melakukan mutasi adalah dalam sebuah kromosom terdapat 128 gen (pada setiap individu) kemudian dilakukan iterasi pada setiap gen, dari gen pertama bangkitkan angka random dari 0 sampai 1, jika angka yang dibangkitkan itu lebih kecil dari *mutation rate*(p_m), maka gen tersebut akan bermutasi dari 0 menjadi 1 atau dari 1 jadi 0, sedangkan jika angka yang dibangitkan lebih dari *mutation rate*(p_m)maka tidak terjadi mutasi pada gen tersebut dan begitu seterusnya sampai gen ke 128. Setelah itu lakukan hal yang sama pada kromosom individu selanjutnya.

5. Probilitas Operasi Genetik

Dalam kasus ini kita menginginkan untuk mencari solusi yang terbaik dalam mencari nilai optimum dari fungsi $h(x,y) = (cos\ x^2 \ * sin\ y^2) + (x+y)$ dengan probabilitas operasi genetik dalam kasus ini menggunakan *crossover rate*(p_c) = 0.800 dan *mutation rate*(p_m) = 0.3, sehingga dengan menggunakan rate tersebut diharapkan mendapatkan solusi yang terbaik dalam kasus fungsi ini. Kami membangkitkan suatu nilai random dari 0 sampai 1, dan apabila nilai random lebih kecil atau sama dengan probabilitas dari *crossover rate* maka terjadi *crossover* namun apabila lebih dari *crossover* ratenya maka tidak terjadi *crossover* begitu pula halnya dengan *mutasi rate*(p_m) pada kasus ini.

6. Metode Pergantian Generasi

Pada inisialisasi awal terdapat 2000 populasi yang kemudian akan menghasilkan anak sebanyak 1000. Kemudian akan digabungkan populasi orang tua dengan populasi anak yang selanjutnya akan dilakukan penghapusan pada nilai fitness yang kosong (akan kosong apabila tidak memenuhi syarat pada fungsi fitness), kemudian populasi akan di sorting secara descending serta akan dilakukan drop jika nilai kromosom x dan y nya duplicate (karena nilai nilai fitness akan sama jika terjadi duplicate kromosom). Terakhir akan dikembalikan 500 individu terbaik dengan nilai fitnessnya untuk ditampilkan dan dilanjutkan ke generasi berikutnya yang akan melakukan fungsi yang sama seperti generasi berikutnya. Serta menampilkan 5 individu terbaik pada tiap generasinya hingga generasi ke 300

Penerusan Generasi

```
[ ]
    def getNewPopulation(anak,ortu):                                                    #input : populasi anak dan orangtua
                                                                                        #output : populasi generasi baru
      fitnessAnak=getFitness(anak)
      fitnessOrtu=getFitness(ortu)


      #untuk kemudahan proses perlu untuk menjadikan array kedalam pandas dataframe
      dfPopulasiAnak = pd.DataFrame(anak,
                        columns=['pertama',
                                 'kedua'])

      dfFitnessAnak = pd.DataFrame(fitnessAnak,
                  columns=['fitness'])

      dfPopulasiAnak=dfPopulasiAnak.join(dfFitnessAnak)

      dfPopulasiOrtu=pd.DataFrame(ortu,
                          columns=['pertama',
                                   'kedua'])

      dfFitnessOrtu=pd.DataFrame(fitnessOrtu,
                  columns=['fitness'])


      dfPopulasiOrtu=dfPopulasiOrtu.join(dfFitnessOrtu)

      #menggabungkan dataframe orang tua dan anak
      dfPopulasi=pd.concat([dfPopulasiAnak, dfPopulasiOrtu],ignore_index=True)

      #menghapus nilai NA ( karena di fungsi fitness akan diciptakan nila NA jika nilai input diluar batas yang ditentukan)
      dfPopulasi= dfPopulasi.dropna(subset=['fitness'])

      #melakukan pengurutan secara descending
      dfPopulasi = dfPopulasi.sort_values(by=['fitness'],ascending=False,ignore_index=True)

      #menghapus duplicate
      dfPopulasi =dfPopulasi.drop_duplicates(subset=['pertama','kedua'],keep="last")
```

```
#menghapus duplicate
dfPopulasi =dfPopulasi.drop_duplicates(subset=['pertama','kedua'],keep="last")

#ambil 500 terbaik untuk dijadikan populasi baru
dfPopulasi =dfPopulasi[:500]

#memisahkan populasi dengan fitness
populasi=dfPopulasi.drop(['fitness'],axis=1)

fitness=dfPopulasi['fitness']

populasi=populasi.to_numpy()

fitness=fitness.to_numpy()

return populasi,fitness
```

Gambar 1.7 Penerusan generasi GA fungsi dari $h(x,y) = (cos\ x^2 \ * \ sin\ y^2\ ) + (x+y)$

7. Kriteria Penghentian Evolusi

   Dalam hal ini evolusi akan terhenti apabila sudah mencapai generasi ke 300 atau dapat ditentukan lagi dengan mengubah pada program main.

HASIL DAN KESIMPULAN

Kami telah melakukan pengkodean dengan menggunakan *Genetic Algorithm* (GA) dalam menyelesaikan kasus dari fungsi $h(x,y) = (cos\ x^2 \ * \ sin\ y^2) + (x+y)$ dengan main program sebagai berikut

▾ Program MAIN

```
[ ]  #main


     banyakPopulasi=2000
     banyakAnak=1000
     crossoverRate=0.800
     mutationRate=0.3
     generasi=300

     #masukin populasi inisiasi
     population=generateIndividu(banyakPopulasi)

     # akan terlihat bahwa
     for gen in range (0,generasi):
       anak=seleksiOrangtua(population,banyakAnak)
       anak=encodeList(anak)

       newAnak=[]
       for kr in anak:
         newAnak.append(buatAnak(kr[0],kr[1],crossoverRate,mutationRate))
       newAnak=decodeList(newAnak)

       population,fitn = getNewPopulation(newAnak,population)

       print(f'generasi ke {gen+1} ')
       print('top 5 :')
       for i in range(0,5):
         print(f'{i+1}. {population[i]} : {fitn[i]}')
```

Gambar 1.8 Main program Program GA fungsi dari $h(x,y) = (cos\ x^2 \ * \ sin\ y^2) + (x+y)$

Dalam main program tersebut seluruh fungsi akan dieksekusi sehingga dalam *Genetic Algorithm* yang kami buat untuk menyelesaikan kasus fungsi tersebut dideklarasikan sebanyak 300 generasi dengan menampilkan 5 anak terbaik dari setiap generasinya dari fungsi yang kita eksekusi :

```
generasi ke 1
top 5 :
1. [0.8763634200000000407 0.99934294000000001290] : 2.48045912824672
2. [1.99827021999999998592 0.98127578000000004188] : 2.4387173668249535
3. [0.98457838842671940860 0.98457838842671940860] : 2.4356824133771453
4. [0.98098755002914450607 0.98098755002914450607] : 2.430993252011642
5. [0.87059063000000003196 0.97143813999999995001] : 2.430104882565952
generasi ke 2
top 5 :
1. [0.8763634200000000407 0.99934294000000001290] : 2.48045912824672
2. [1.99827021999999998592 0.98127578000000004188] : 2.4387173668249535
3. [0.98457838842671940860 0.98457838842671940860] : 2.4356824133771453
4. [0.98098755002914450607 0.98098755002914450607] : 2.430993252011642
5. [0.87059063000000003196 0.97143813999999995001] : 2.430104882565952
generasi ke 3
top 5 :
1. [0.8763634200000000407 0.99934294000000001290] : 2.48045912824672
2. [0.99821220897413809325 0.99821220897413809325] : 2.4525481514349
3. [0.99234992688447976050 0.99234992688447976050] : 2.445479118095501
4. [1.99827021999999998592 0.98127578000000004188] : 2.4387173668249535
5. [0.98457838842671940860 0.98457838842671940860] : 2.4356824133771453

generasi ke 298
top 5 :
1. [0.8763634200000000407 0.99934294000000001290] : 2.48045912824672
2. [0.99909687328347196456 0.99909687328347196456] : 2.4535908216263564
3. [0.99836637377942094584 0.99836637377942094584] : 2.4527303066922803
4. [0.99836351449952598358 0.99836351449952598358] : 2.452726930026566
5. [0.99836351443267723482 0.99836351443267723482] : 2.4527269299476204
generasi ke 299
top 5 :
1. [0.8763634200000000407 0.99934294000000001290] : 2.48045912824672
2. [0.99909687328347196456 0.99909687328347196456] : 2.4535908216263564
3. [0.99836637377942094584 0.99836637377942094584] : 2.4527303066922803
4. [0.99836351449952598358 0.99836351449952598358] : 2.452726930026566
5. [0.99836351443267723482 0.99836351443267723482] : 2.4527269299476204
generasi ke 300
top 5 :
1. [0.8763634200000000407 0.99934294000000001290] : 2.48045912824672
2. [0.99909687328347196456 0.99909687328347196456] : 2.4535908216263564
3. [0.99836637377942094584 0.99836637377942094584] : 2.4527303066922803
4. [0.99836351449952598358 0.99836351449952598358] : 2.452726930026566
5. [0.99836351443267723482 0.99836351443267723482] : 2.4527269299476204
```

Gambar 1.9 ke-300 Generasi yang dieksekusi dari fungsi dari $h(x,y) = (cos\ x^2 \ * sin\ y^2) + (x+y)$

Sehingga kesimpulan dari hasil observasi dari fungsi $h(x,y) = (cos\ x^2 \ * sin\ y^2) + (x+y)$ bahwa dengan menggunakan *Genetic Algorithm* akan jauh lebih efektif untuk menyelesaikan permasalahan mengenai nilai optimum dari suatu fungsi tersebut. Dengan mempertimbangkan banyaknya gen, populasi, kromosom, *crossover rate*(p_c), dan mutasi rate(p_m)agar mendapatkan nilai optimum. Untuk algoritma yang kami gunakan didapat solusi optimum dari fungsi tersebut adalah x =0.87636342000000000407 dan y=0.99934294000000001290 dengan nilai fitness 2.48045912824672.

Video Presentasi dari masing - masing anggota :

- Alif Adwitiya Pratama (1301190465)

  Link :
  https://drive.google.com/file/d/1opLlvIPBm7Fo00zc5cSG80S0_bf23AjR/view?usp=sharing

- Sabrina Adinda Sari (1301194183)

  Link :
  https://drive.google.com/file/d/1CHK3o4xjY5r30CUtJ6rJD2Qk5CHhYNrD/view?usp=sharing

- Wana Ardilah Iwan (1301194522)

  Link:
  https://drive.google.com/file/d/1ncBZLlC4U72liyd6F4r9yFmaBrYpDvBp/view?usp=sharing

DAFTAR PUSTAKA

Socs.binus.ac.id. (2018, 08 Desember). *Genetic Algorithm.* Diakses pada 21 Maret 2021, dari https://socs.binus.ac.id/2018/12/08/genetic-algorithm/

Yuliana.lecturer.pens.ac.id. (2015, 31 Agustus) Bab 7 Algoritma Genetika. Diakses pada 21 Maret 2021, dari http://yuliana.lecturer.pens.ac.id/Kecerdasan%20Buatan/Buku/Bab%207%20Algoritma%20Genetika.pdf

Eprints.uny.ac.id (). Pengertian Belajar Matematika. Diakses pada 21 Maret 2021, dari http://eprints.uny.ac.id/40801/2/BAB%20II.pdf

Media.neliti.com (2015, 01 Juli) Penerapan Metode Algoritma Genetika Untuk Permasalahan Penjadwalan Perawat. Diakses pada tanggal 23 Maret 2021, dari https://media.neliti.com/media/publications/66056-ID-penerapan-metode-algoritma-genetika-untu.pdf

Media.neliti.com (2014, 18 Juli) Penerapan Algoritma Genetika *Traveling Salesman Problem with Time Window* : Studi Kasus Rute Antar Jemput *Laundry*, Diakses pada tanggal 23 Maret 2021, dari https://media.neliti.com/media/publications/77747-ID-penerapan-algoritma-genetika-traveling-s.pdf