YILDIZ TEKNIK UNIVERSITESI BILGISAYAR MUHENDISLIGI

Ogrenci No:21011011

Ad Soyad:Turabi Yıldırım

Öğrenci E Postası: turabi.yildirim@std.yildiz.edu.tr

BLM-1012-YAPISAL PROGRAMLAMAYA GİRİŞ FİNAL PROJESİ

GENETIK ALGORITMA

Ders Yürütücüsü

Öğretim Görevlisi Dr. Ahmet ELBİR

Haziran 2021

İÇERİK

- **Genetik Algoritma nedir? Ne işe yarar? Nasıl çalışır?**
- **Kullanım Yerleri**
- **Avantajları-Dezavantajları**
- **Karmaşıklığı**
- **Sınırları ve rakipleri**
- **Analizde kullanılan ekran çıktıları**
- **C kodu**
- **Kaynaklar**

VİDEO ADRESİ

https://youtu.be/59ZkcyYph0Q

Genetik Algoritma

- İlk başta rastgele bir algoritma üzerinden dener.
- Sonra hedefe yakın olan algoritmalara daha çok birleşme şansı vererek diğer algoritmalarla birleştirir.
- Bu sayede hedefe uzak algoritmalar yaşarken hedefe yakın algoritmalar ölür.

Nasıl Çalışır

- Genetik algoritma çalışma sistemi diğer yapay zeka sistemlerine göre ve çoğu algoritmaya göre basittir.

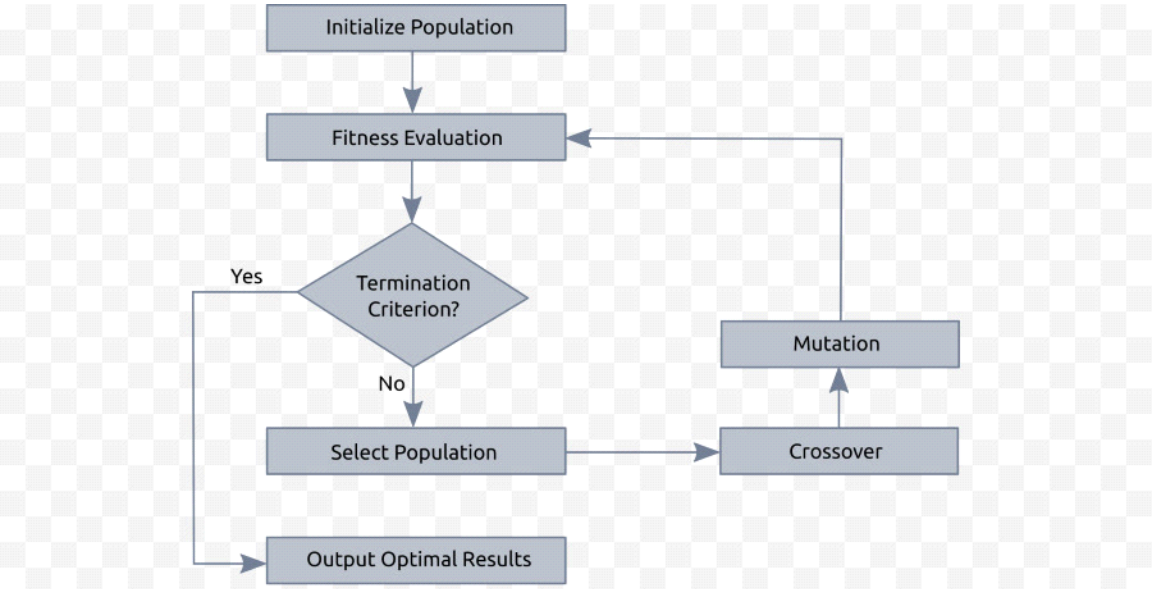

Figure 2: Basic structure of Genetic Algorithm

- Crossover:Yüksek olasılıkla diğer algoritmalara göre başarılı 2 algoritmayı birleştirir.
- Fitness:Genetik algoritmanın en zor ve en önemli kısmı. Oluşan algoritmaları hedefe uygunluğuna ya da başka parametrelere göre(diğer algoritmalara göre farklılığı da etkileyebilir) verilen sonraki nesillere kalma şansı.
- Mutation:Rastgele oluşan algoritmanın kendiliğinden değişme şansı

Kullanım Yerleri

- Sadece genetik algoritmanın kullanıldığı başarılı pek örnek yok.
- Evrimsel algoritmaların (arı kovanı algoritması vb.) kullanıldığı örnekler:
- Borsa tahmini
- Oyun Yapay Zekaları
- Teaching computer walk examples https://www.youtube.com/watch?v=GOFws_hhZs8

Avantajları

- Uygunluğu kolay test edilebilen alanlarda kendini çok rahatlıkla geliştirebiliyor.
- Performans açısından yeterince iyi bir fitness fonksiyonuyla belki bir çok algoritmadan hızlı çalışabilir

Dezavantajları

- Fitness fonksiyonunu belirlemek çok zor çünkü dünyamızdaki ya da optimum bir dünyadaki doğal seçilim sistemini bilmiyoruz hesaplayacak bir matematik formülü de yok.
- Bu yüzden fitness fonksiyonları genelde deneme yanılma yöntemleriyle belirleniyor.
- Uygunluğu test edilmesi zorsa hedefe ulaşması imkansızlaşabiliyor.

Rakipleri

- Artifical bee colony algorithm(D.Karaboğa)
- Bees algorithm
- The runner Root Algorithm
- Rakipleri arasında avantajları ve dezavantajları istenen göreve göre değişse de genelde bu algoritmalar beraber kullanılmaktadır

Yer Karmaşıklığı

- Fitness fonksiyonundan fonksiyonuna değişir ama benim fonksiyonumda matrisin boyutundan(matris uzunluk*matris genişlik) ya da struct boyutundan dolayı(struct sayısı*order sayısı) O(n^2)

Zaman Karmaşıklığı

- İlk cocuklara random order verme=ORDERNUM*CS
- Cocukları basa koyma=CS
- Cocukları Yurutme=CS*ORDERNUM
- Uzaklık Hesaplama=CS
- Uzaklık Sıralama=Selection Sort/O(n^2)
- SagSolAyırma=CS
- Havuz ousturma=HAVUZ

O(ORDERNUM*CS+CS+CS*ORDERNUM+CS+n^2+CS+HAVUZ)

En büyük katsayı yaklasık n^2 kare oldugundan O(n^2)

Algoritma Analizi

Zaman karmaşıklığı işleme göre değişmediğinden mutasyonla optimizasyonu arasında bir analiz yaptım.

Mutasyon olasılığına göre hedefe ulaşana kadar kac jenerasyon üretildiğini gösteriyor.

1/100 Mutasyon olasılığı

```
Generation Sayisi=41
```

```
Generation Sayisi=38
```

```
Generation Sayisi=1178
```

```
Generation Sayisi=40187
```

```
Generation Sayisi=47
```

1/500

```
Maksimum jenerasyona ulasildi,hedefe ulasilamadi.
Generation Sayisi=100002
```

```
Generation Sayisi=67
```

```
Generation Sayisi=10791
```

```
Generation Sayisi=13169
```

```
Generation Sayisi=252
```

1/1000

```
Generation Sayisi=103
```

```
Maksimum jenerasyona ulasildi,hedefe ulasilamadi.
Generation Sayisi=100002
```

```
Maksimum jenerasyona ulasildi,hedefe ulasilamadi.
Generation Sayisi=100002
```

```
Generation Sayisi=384
```

```
Maksimum jenerasyona ulasildi,hedefe ulasilamadi.
Generation Sayisi=100002
```

Zaman Karmaşıklığı analizi

- Jenerasyon sayısına göre ne işlemin ne kadar sürdüğüyle ilgili bir analiz yaptım ve jenerasyon sayısının artışıyla işlem sayısının artışının oranının lineer olduğunu buldum.

```
10 jenerasyon 29418 milisaniye surdu
```

```
100 jenerasyon 29919 milisaniye surdu
```

```
1000 jenerasyon 177171 milisaniye surdu
```

```
10000 jenerasyon 1265491 milisaniye surdu
```

```
100000 jenerasyon 10932399 milisaniye surdu
```

```
1000000 jenerasyon 88154082 milisaniye surdu
```

C dilinde kod

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <Windows.h>

#define MB 50

#define CS 20

#define HAVUZ 2048

#define MUTASYONSANSI 100//1 / mutasyon sansini belirler.

#define ORDERNUM 98

#define LIMIT 100000

//Uzakliktaki sirasina gore fitness olacak(fitnessi sirasina gore verilecek).

//Fitness/Total Fitness = Ciftlesme sansi.

//Cocuk sayisi burada ebeveyn sayisindan az ya da fazla olabilir.

//Ata birey sayisi 50 olacak.

//Crossover birinin yarisini oburunun yarisini alacak.

//Yukari orderi 1 sag orderi 2 sol orderi 3.

//Ilk basta mutasyonlar cok olmasi iyi sonra isi yola soktuktan sonra mutasyon azalmasi iyi.

//En cok fitnessa sahip olanlarin daha cok oldugu bir kume olustur ve o kumeden rastgele kisiler secerek ciftlestir.

//Sola gidenle solu saga gidenle sagi ciftlestir.

//Heralde 1 valueya 1 farkli random sayiyi atiyor ayni value olunca ayni random sayiyi atiyor.

//Duvara carpinca duvar uzakligini da ekle duvara ekstra penaltyli de dene.

//Duvar penaltysi uzerinden algoritma duvar penaltysi uzerinden sayisal analiz yapilarak da gelistirilebilir de.

//Bu yapay zeka algoritmasi farkli algoritmalar kullanarak kendi kendini de gelistirebilir de.

typedef struct cocuk{

   int order[ORDERNUM];

   int uzaklik;

   int konum[2];//0 x 1 y

   int sagSol;//0 sol 1 sag

} cocuk;

//Yazdirma fonksiyonlari

void haritaYazdir(int matris[MB][MB]);

void cocukAllAttributesYazdir(cocuk c[CS]);
```

```
void havuzYazdir(int havuz[HAVUZ]);
//Olusturma fonksiyonlari
int haritaOlustur(int matris[MB][MB]);
void cocuklarRandomOrder(cocuk c[CS]);
void cocuklarBasaKoy(cocuk c[CS]);
void cocuklarYurut(cocuk c[CS],int matris[MB][MB]);
void uzakliklariHesapla(cocuk c[CS], int xCoord);
void uzaklikSirala(cocuk c[CS]);
void swapChildren(cocuk* c1,cocuk* c2);
void sagSolAyir(cocuk c[CS],int xCoord);
void havuzOlustur(int havuz[HAVUZ]);
void cocuklariYap(cocuk c[CS],int havuz[HAVUZ]);//GenerationNum sonradan ekle
void ciftlestir(cocuk* c1,cocuk* c2,cocuk* c3);
void yeniNesilEslestir(cocuk ata[CS],cocuk yeniNesil[CS]);
void sonrakiNesil(cocuk cocuklar[CS],int matris[MB][MB],int xCoord,int havuz[HAVUZ]);
void ilkNesil(cocuk cocuklar[CS],int matris[MB][MB],int xCoord,int havuz[HAVUZ]);
void copyChildrens(cocuk c1[CS],cocuk c2[CS]);
//Menu fonksiyonlari
void generationTarget();
void generationNumCreate();
void menu();

int main(){
menu();
}
void menu(){
int order=0;
do{
printf("1-Jenerasyon sayisi girerek olusturma\n2-Hedefe ulasana kadar yeni jenerasyon olusturma\n(Cikmak icin -1 yaziniz)\n");
scanf("%d",&order);
switch(order){
case 1:
system("cls");
generationNumCreate();
break;
```

```
case 2:
system("cls");
generationTarget();
break;
}
}
while(order!=-1);
}
void generationTarget(){
int matris[MB][MB];
   cocuk cocuklar[CS];
   cocuk eskiCocuklar[CS];
int xCoord = haritaOlustur(matris);
   int havuz[HAVUZ];
   int generationNum=0;
   int i;
   srand(time(NULL));
   int innerOrder=0;
   int devamke=1;
   printf("Harita olusturuldu...\n");
   ilkNesil(cocuklar,matris,xCoord,havuz);
   generationNum++;
   printf("Ilk generation olusturuldu...\n");
   copyChildrens(cocuklar,eskiCocuklar);
   while(devamke) {
                printf("Nesil olusturuldu.");
      sonrakiNesil(cocuklar,matris,xCoord,havuz);
      if(cocuklar[0].uzaklik==0){
devamke=0;
}
if(generationNum>LIMIT){
devamke=0;
}
generationNum++;
   }
```

```
   system("cls");
   do{
   if(generationNum==100002){
            printf("Maksimum jenerasyona ulasildi,hedefe ulasilamadi.\n");
}
   printf("Generation Sayisi=%d\n1-Haritayi yazdir\n2-Ilk generationu yazdir\n3-Son generationu yazdir\n(Deneyi bitirmek icin -1 yaziniz)\n",generationNum);

   scanf("%d",&innerOrder);
   system("cls");
   switch(innerOrder){
            case 1:
                     haritaYazdir(matris);
break;
case 2:
cocukAllAttributesYazdir(eskiCocuklar);
break;
case 3:
cocukAllAttributesYazdir(cocuklar);
break;
}
}
while(innerOrder!=-1);
}
void generationNumCreate(){
int matris[MB][MB];
   cocuk cocuklar[CS];
   cocuk eskiCocuklar[CS];
int xCoord = haritaOlustur(matris);
   int havuz[HAVUZ];
   int generationNum;
   int i;
   srand(time(NULL));
   int innerOrder=0;
   printf("Harita olusturuldu...\n");
```

```
   ilkNesil(cocuklar,matris,xCoord,havuz);
   printf("Ilk generation olusturuldu...\n");
   copyChildrens(cocuklar,eskiCocuklar);
   printf("Generation sayisini giriniz:");
   scanf("%d",&generationNum);
   for(i=0;i<generationNum;i++) {
            printf("Nesil olusturuldu.");
      sonrakiNesil(cocuklar,matris,xCoord,havuz);

   }
   system("cls");
   do{
   printf("1-Haritayi yazdir\n2-Ilk generationu yazdir\n3-Son generationu yazdir\n(Deneyi bitirmek icin -1 yaziniz)\n");
   scanf("%d",&innerOrder);
   system("cls");
   switch(innerOrder){
            case 1:
                     haritaYazdir(matris);
break;
case 2:
cocukAllAttributesYazdir(eskiCocuklar);
break;
case 3:
cocukAllAttributesYazdir(cocuklar);
break;
}
}
while(innerOrder!=-1);
}
void sonrakiNesil(cocuk cocuklar[CS],int matris[MB][MB],int xCoord,int havuz[HAVUZ]){
cocuklariYap(cocuklar, havuz);
   cocuklarBasaKoy(cocuklar);
   cocuklarYurut(cocuklar, matris);
   uzakliklariHesapla(cocuklar, xCoord);
   uzaklikSirala(cocuklar);
```

```
   sagSolAyir(cocuklar, xCoord);
   havuzOlustur(havuz);
}
void ilkNesil(cocuk cocuklar[CS],int matris[MB][MB],int xCoord,int havuz[HAVUZ]){
cocuklarRandomOrder(cocuklar);
   cocuklarBasaKoy(cocuklar);
   cocuklarYurut(cocuklar,matris);
   uzakliklariHesapla(cocuklar,xCoord);
   uzaklikSirala(cocuklar);
   sagSolAyir(cocuklar,xCoord);
   havuzOlustur(havuz);
}
void yeniNesilEslestir(cocuk ata[CS],cocuk yeniNesil[CS]){
   int i;
   for(i=0;i<CS;i++){
      swapChildren(&ata[i],&yeniNesil[i]);
   }
}
void ciftlestir(cocuk* c1,cocuk* c2,cocuk* c3){
   int i;
   for(i=0;i<ORDERNUM;i++){
      if(i<ORDERNUM/2){
         c3->order[i]=c1->order[i];
      }
      else{
         c3->order[i]=c2->order[i];
      }
      if(rand()%MUTASYONSANSI==0){//
         c3->order[i]=(c3->order[i]+1)%3;
      }
   }
}
void cocuklariYap(cocuk atalar[CS],int havuz[HAVUZ]){
   int i,j=0;
```

```
    cocuk yeninesil[CS];
    int randomlar[CS*2];
    for(i=0;i<CS*4;i++){
        randomlar[i]=rand()%2048;
    }
    for(i=0;i<CS;i++){
        if(atalar[havuz[randomlar[(i+j)*2]]].sagSol==atalar[havuz[randomlar[(i+j)*2+1]]].sagSol) {//Sag sola gore ayni taraftaysa ciftlestirir.
            ciftlestir(&atalar[havuz[randomlar[(i + j) * 2]]], &atalar[havuz[randomlar[(i + j) * 2 + 1]]],
                    &yeninesil[i]);
        }
        else{
            i--;
            j++;
        }
    }
    yeniNesilEslestir(atalar,yeninesil);
}
void havuzYazdir(int havuz[HAVUZ]){
    int i;
    for(i=0;i<HAVUZ;i++){
        printf("%d ",havuz[i]);
    }
    printf("\n");
}
void havuzOlustur(int havuz[HAVUZ]){
    int havuzBuyukluk = HAVUZ/4,i;
    int j=0;
    int baslangic=0;
    while(havuzBuyukluk>1){
        for(i=baslangic;i<baslangic+havuzBuyukluk;i++){
            havuz[i] = j;
        }
        baslangic=baslangic+havuzBuyukluk;
        j++;
        if(j%2==0){
```

```
            havuzBuyukluk= havuzBuyukluk/2;
        }


    }
}
void cocukAllAttributesYazdir(cocuk c[CS]){
    int i,j;
    for(i=0;i<CS;i++){
        printf("\n");
        printf("Order:");
        for (j=0; j<ORDERNUM; j++) {
            printf("%d,", c[i].order[j]);
        }
        printf("\n");
        printf("[x][y]=[%d][%d]", c[i].konum[0], c[i].konum[1]);
        printf("\n");
        printf("%d.Uzaklik=%d\n",i+1,c[i].uzaklik);
        printf("%d.cocuk sag sol=%d\n",i+1,c[i].sagSol);
        printf("\n");
    }
}
void sagSolAyir(cocuk c[CS],int xCoord){
    int i;
    for(i=0;i<CS;i++){
        if(c[i].konum[0]<xCoord){
            c[i].sagSol=0;
        }
        else{
            c[i].sagSol=1;
        }
    }


}
void swapChildren(cocuk* c1,cocuk* c2){
    int temp,i;
```

```
   temp=c1->uzaklik;
   c1->uzaklik=c2->uzaklik;
   c2->uzaklik=temp;
   temp=c1->sagSol;
   c1->sagSol=c2->sagSol;
   c2->sagSol=temp;
   for(i=0;i<ORDERNUM;i++){
      temp=c1->order[i];
      c1->order[i]=c2->order[i];
      c2->order[i]=temp;
   }
   for(i=0;i<2;i++){
      temp=c1->konum[i];
      c1->konum[i]=c2->konum[i];
      c2->konum[i]=temp;
   }
}
void copyChildrens(cocuk c1[CS],cocuk c2[CS]){
int i,j;
for(i=0;i<CS;i++){
c2[i].uzaklik=c1[i].uzaklik;
c2[i].sagSol=c1[i].sagSol;
for(j=0;j<ORDERNUM;j++){
                c2[i].order[j]=c1[i].order[j];
                }
                for(j=0;j<2;j++){
                        c2[i].konum[j]=c1[i].konum[j];
}
}
}
void uzaklikSirala(cocuk c[CS]){
   int i,j,enKucuk,enKucukPos,temp;
   //Uzakliklar sorting
   for(i=0;i<CS;i++){
      enKucuk=9999999;
```

```
        for(j=i;j<CS;j++){
            if(c[j].uzaklik<enKucuk){
                enKucuk=c[j].uzaklik;
                enKucukPos=j;
            }
        }
        swapChildren(&c[i],&c[enKucukPos]);
        enKucuk=9999999;
    }
}
void uzakliklariHesapla(cocuk c[CS], int xCoord){
    int x,y,i;
    for(i=0;i<CS;i++){
        x=c[i].konum[0];
        y=c[i].konum[1];
        c[i].uzaklik=abs(xCoord-x)*abs(xCoord-x)+abs(MB-y)*abs(MB-y);
        if(y==MB/2){
            c[i].uzaklik=c[i].uzaklik*2;
        }
    }
}
void cocuklarYurut(cocuk c[CS],int matris[MB][MB]){
    int i,j;
    for(j=0;j<CS;j++){
        for(i=0;i<ORDERNUM;i++){
        if(c[j].order[i]==0){
            if(matris[c[j].konum[1]][c[j].konum[0]+1]==-1){
            }
            else if(c[j].konum[1]+1==MB+1){
            }
            else{
                c[j].konum[1]++;
            }
        }
```

```
        if(c[j].order[i]==1){
            if(c[j].konum[0]+1==MB+1){
            }
            else{
                c[j].konum[0]++;
            }
        }
        if(c[j].order[i]==2){
            if(c[j].konum[0]==0){
            }
            else{
                c[j].konum[0]--;
            }
        }
    }
    }
}
void cocuklarBasaKoy(cocuk c[CS]){
    int i;
    for(i=0;i<CS;i++) {
        c[i].konum[1] = 0;
        c[i].konum[0] = MB / 2;
    }
}
void cocuklarRandomOrder(cocuk c[CS]){
    int i,j;
    for(j=0;j<CS;j++) {
        for (i = 0; i < ORDERNUM; i++){
            c[j].order[i] = rand() % 3;
        }
    }
};
int haritaOlustur(int matris[MB][MB]){//matris[y][x]
int i,j,xCoord;
    printf("Haritada gidilecek yerin x koordinati: ");
```

```
   scanf("%d",&xCoord);
for(i=0;i<MB;i++){
for(j=0;j<MB;j++){
matris[i][j]=0;
}
}
for(i=MB/4;i<=3*MB/4;i++){
matris[MB/2][i]=-1;
}
matris[MB-1][xCoord]=2;
matris[0][MB/2]=1;
   return xCoord;
}

void haritaYazdir(int matris[MB][MB]){
int i,j;
for(i=0;i<MB;i++){
for(j=0;j<MB;j++){
if(matris[i][j]==0){
printf("* ");
}
if(matris[i][j]==1){
printf("# ");
}
if(matris[i][j]==-1){
printf("X ");
}
if(matris[i][j]==2){
printf("$ ");
}
}
printf("\n");
}
}
```

Kaynaklar:

- **https://abc.erciyes.edu.tr/**
- Network Cartoon png download - 600*513 - Free Transparent Genetic Algorithm png Download. - CleanPNG / KissPNG