1 Genetické algoritmy

Naším cieľom je urobiť triviálnu implementáciu jednoduchého genetického algoritmu v MATLABe. Budeme hľadať maximum funkcie

$$f(x) = (x^2 - 28x + 180)\sin\left(\frac{x}{3 + \cos(2x)}\right)$$

na intervale $< A, B >$, kde $A = 10$ a $B = 22$.

1.1 Pseudokód jednoduchého genetického algoritmu

Vygeneruj počiatočnú populáciu
    ohodnoť všetkých jedincov
    vyber najlepších jedincov k reprodukcii (selekcia)
    kríženie
    mutácia
dokiaľ nie je splnená podmienka zastavenia

1.2 Realizácia v MATLABe

Jedinec bude kódovaný ako vektor bitov dĺžky `codelen=10`. Populácia pre nás bude matica, ktorej riadky budú kódy jedincov. Počet jedincov v populácii označme `popsize=20`. Náš kód dovoľuje zakódovať $n = 2^{codelen} = 2^{10}$ rôznych hodnôt. Tie rovnomerne rozložíme na interval $< A, B >$. Vektor $v = [v_1, \ldots, v_n]$ bude kódovať číslo

$$x = A + (B - A)\frac{\text{číslo}(v)}{n - 1},$$

kde číslo(v) celé číslo s bitovým zápisom $v_1 v_2 \ldots v_n$.

1.2.1 Generovanie počiatočnej generácie

```
>> codelen=10; popsize=12;
>> A=10; B=22;
>> pop=round(rand(popsize,codelen))

pop =

     1     1     0     1     0     0     1     0     0     1
     0     1     0     1     0     1     1     1     0     1
     1     0     0     0     0     1     0     0     0     0
     0     0     1     1     1     1     1     0     1     0
     1     1     0     1     0     1     1     1     1     1
     1     1     0     0     1     1     0     1     0     1
     0     0     0     1     0     0     1     0     1     0
     0     1     1     0     1     0     1     0     0     1
     1     0     0     1     0     0     1     1     0     1
     0     0     1     1     1     1     1     0     0     0
     1     1     0     1     1     1     1     0     1     1
     1     0     0     0     1     0     1     1     1     0
```

Nastavíme parametre algoritmu – pravdepodobnosť kríženia a pravdepodobnosť mutácie.

```
>> crossprob=0.8; mutprob=0.08;
```

1.2.2 Ohodnotenie jedincov

Bitový kód sa dá previesť na celé číslo nasledovne:

```
>> bin2dec(int2str(pop))

ans =

   841
   349
   528
   250
   863
   821
    74
   425
   589
   248
   891
   558
```

Teda vektory jedincov kódujú čísla:

```
>> x=A + (B-A)*bin2dec(int2str(pop))./(2^codelen-1)
```

Funkcia $f(x)$ nadobúda v týchto bodoch hodnoty:

```
>> y=(x.^2-28*x+180).*sin(x./(3+cos(2*x)))
```

ALE POZOR! V niektorých bodoch je hodnota funkcie záporná. Preto funkciu upravíme pričítaním vhodnej konštanty (napr. 35) tak, aby dávala iba kladné hodnoty:

```
>> y=(x.^2-28*x+180).*sin(x./(3+cos(2*x)))+35
```

Aby sme mohli sledovať priebeh algoritmu, spočítame maximálnu a priemernú hodnotu upravenej funkcie v bodoch vektora `x`:

```
>> MAX=max(y)
>> MEAN=sum(y)/size(y,1)
```

1.2.3 Selekcia jedincov na základe ich ohodnotenia – ruleta

Každému jedincovi pridelíme časť ruletového kola umernú jeho ohodnoteniu:

```
>> prob=cumsum(y)/sum(y)
```

Generujeme náhodné čísla z intervalu $< 0, 1 >$. Podľa úseku kola rulety kam toto číslo padne dáme príslušný reťazec do novej generácie:

```
>> for i= 1 : popsize
>>      newpop(i,:)=pop(find((prob>=rand),1,'first'),:);
>> end
```

1.2.4 Kríženie

Dva po sebe idúce jedince tvoria pár. Preto sme volili veľkosť populácie párne číslo. S pravdepodobnosťou `crossprob` budú tieto dva jedince skrížené v náhodne zvolenom bode `c`:

```
>> for i=1 : popsize/2-1
>>      if rand<=crossprob
>>          c=floor((codelen-1)*rand)+1;
>>          pop(2*i-1,:)=[newpop(2*i-1,1:c) newpop(2*i,c+1:codelen)];
>>          pop(2*i,:)=[newpop(2*i,1:c) newpop(2*i-1,c+1:codelen)];
>>      else
>>          pop(2*i-1,:)=newpop(2*i-1,:);
>>          pop(2*i,:)=newpop(2*i,:);
>>      end
```

1.2.5 Mutácia

Prejdeme postupne všetky bity všetkých reťazcov a každý bit zmeníme s pravdepodobnosťou `mutprob` na opačný:

```
>> for i=1:popsize
>>     for j= 1:codelen
>>         if rand<=mutprob
>>             pop(i,j)=1-pop(i,j);
>>         end
>>     end
>> end
```

Ďalej sa postup opakuje od ohodnocovania jedincov.

Algoritmus končí, keď sme spokojní s ohodnotením najlepšieho jedinca, alebo po zadanom počte opakovaní.

Vyššie uvedený postup je uložený vo forme skriptu (nie funkcie) v súbore ga.m. V tomto skripte je naviac implementovaný tzv. elitizmus – `elit` najlepších reťazcov z danej populácie sa kopíruje bez zmeny do nasledujúcej generácie. Skript je taktiež doplnený o kreslenie obrázku funkcie s vyznačenými hodnotami ohodnocovacej funkcie. Pri prezentovaní výsledku nesmieme zabudnúť, že ohodnocovacia funkcia nie je priamo optimalizovaná funkcia.