

1. Computação Evolutiva

Prof. Renato Tinós

Programa de Pós-Graduação Em
Computação Aplicada

Depto. de Computação e Matemática
(FFCLRP/USP)



1.2. Algoritmos Genéticos

1.2.1. Introdução

1.2.2. Elementos de Algoritmos Genéticos (AGs)

1.2.3. Projeto de AGs

1.2.4. Exemplos

1.2.5. Implementação Computacional

1.2.5. Implementação Computacional

```
// Baseada em [Goldberg, 1989]
```

```
...
```

```
/* Definição de constantes */
```

```
defina    maxpop = 100
```

```
// tamanho maximo da populacao
```

```
defina    tamcrom = 30
```

```
// tamanho do cromossomo
```

```
/* Definição dos tipos de dados */
```

```
tipo_de_dado alelo : booleana
```

```
// tipo de dado que os alelos
```

```
// podem assumir (poderia ser
```

```
// outros tipos como real e
```

```
// inteiro)
```

```
tipo_de_dado cromossomo : alelo[tamcrom]
```

```
tipo_de_dado individuo :
```

```
{
```

```
    crom :    cromossomo
```

```
// cromossomo
```

```
    fitness : real
```

```
// funcao de avaliacao
```

```
    pai1 , pai2 , jcross : inteiro
```

```
// indice dos pais e ponto de
```

```
// crossover
```

```
}
```

```
tipo de dado populacao :    individuo[maxpop]
```

```
...
```

1.2.5. Implementação Computacional

...

/* Definição da variáveis globais */

declare popvelha , popnova : população

declare poptam, lcrom, gen , maxgen : inteiro

declare pcross , pmutacao , somafitness : real

declare nmutacao , ncross : inteiro

declare media , max , min : real

// tipos usados nas estatísticas

// tipos usados nas estatísticas

...

1.2.5. Implementação Computacional

/* Programa principal */

procedimento ag()

gen ← 0

inicialização ()

// procedimento para inicialização
// das variáveis e da população

faça

gen ← gen + 1

// número de gerações

geração ()

estatística(poptam , max , media , min , somafitness , popnova)

impressão ()

// proc. para impressão na tela

popvelha ← popnova

enquanto (gen ≤ maxgen)

1.2.5. Implementação Computacional

procedimento geração()

declare j , pai1, pai2 , jcross : inteiro

j ← 1

faça

// Escolha dos Pais

pai1 ← seleção(poptam , somafitness , popvelha)

pai2 ← seleção(poptam , somafitness , popvelha)

// Reproducao

crossover(popvelha[pai1].crom , popvelha[pai2].crom ,
popnova[j].crom , popnova[j+1].crom,
lcrom , ncross , nmutacao , &jcross , pcross , pmutacao)

// Filho j

popnova[j].fitness ← calcfitness(popnova[j].crom , lcrom)

popnova[j].pai1 ← pai1 // pai 1 do filho j

popnova[j].pai2 ← pai2 // pai 2 do filho j

popnova[j].jcross ← jcross // ponto de crossover do filho j

// Filho j + 1

popnova[j+1].fitness ← calcfitness(popnova[j+1].crom , lcrom)

popnova[j+1].pai1 ← pai1 // pai 1 do filho j

popnova[j+1].pai2 ← pai2 // pai 2 do filho j

popnova[j+1].jcross ← jcross // ponto de crossover do filho j

j ← j + 2

// incremento do índice da populacao

enquanto (j ≤ poptam)

1.2.5. Implementação Computacional

```
função seleção( poptam : inteiro ; somafitness : real ; pop : população ) : inteiro

    declare randomico , somaparcial : real
    declare j : inteiro
    somaparcial  $\leftarrow$  0
    j  $\leftarrow$  0
    randomico  $\leftarrow$  rand(0,1) * somafitness    // gera o ponto de parada da roleta
                                              // rand(l,r) gera um variável aleatória real
                                              //   entre [ l , r ] com distribuição uniforme

    faça
        j  $\leftarrow$  j + 1
        somaparcial  $\leftarrow$  somaparcial + pop[ j ].fitness
    enquanto (somaparcial < randomico e j < poptam)
    retorne ( j )
```

1.2.5. Implementação Computacional

```
procedimento crossover( pai1 , pai2 , filho1 , filho2 : cromossomo ;  
                        lcrom , ncross , nmutacao , jcross : inteiro;  
                        pcross , pmutacao : real)  
  
  declare j : inteiro  
  se ( rand(0,1) ≤ pcross )  
    jcross ← randint( 1 , lcrom-1 )    // gera o ponto de crossover  
                                         // randint(l,r) gera um variável aleatória  
                                         // inteira entre [ l , r ] com distrib. uniforme  
    ncross ← ncross + 1                // contador de crossovers  
  
  senão  
    jcross ← lcrom  
  
  fim se  
  para j ← 1 até j ← jcross  
    filho1[ j ] ← mutacao( pai1[ j ] , pmutacao , nmutacao )  
    filho2[ j ] ← mutacao( pai2[ j ] , pmutacao , nmutacao )  
  
  fim para  
  se ( jcross < lcrom )  
    para j ← jcross+1 até j ← lcrom  
      filho1[ j ] ← mutacao( pai2[ j ] , pmutacao , nmutacao )  
      filho2[ j ] ← mutacao( pai1[ j ] , pmutacao , nmutacao )  
    fim para  
  
  fim se
```

1.2.5. Implementação Computacional

```
função mutacao( alelovalor : alelo ; pmutacao : real ; nmutacao : inteiro ) : inteiro

    se ( rand(0,1) ≤ pmutacao )
        nmutacao ← nmutacao + 1    // contador de mutações
        retorne ( !alelovalor )    // not alelovalor
                                   // se os alelos assumissem valores
                                   // inteiros, deveria
                                   // retornar um numero inteiro aleatorio
    senão
        retorne ( alelovalor )
    fim se
```

1.2.5. Implementação Computacional

/* Calcula estatísticas da população */

procedimento estatística(poptam : inteiro ; max , media , min , somafitness : real ;
pop : populacao)

declare j : inteiro

somafitness ← pop[1].fitness // soma do fitness da população
min ← pop[1].fitness // mínimo valor de fitness da população
max ← pop[1].fitness // máximo valor de fitness da população

para j ← 2 **até** j ← poptam

 somafitness ← somafitness + pop[j].fitness

se (pop[j].fitness > max)
 max ← pop[j].fitness

fim se

se (pop[j].fitness < min)
 min ← pop[j].fitness

fim se

fim para

media ← somafitness / poptam // média de fitness da população

1.2.5. Implementação Computacional

Exemplo 2.4. Achar o máximo da função

$$f(z) = (z)^{10}$$

Sendo z um número inteiro positivo entre 0 e $(2^{30}-1)$
[Goldberg, 1989]

– Observações

- Se um esquema enumerativo é utilizado, deve-se analisar 2^{30} possíveis soluções
 - se o tempo de análise de cada solução for de 1 ms, então serão necessários cerca de 2 anos para a análise de todas as soluções
- Se a função for normalizada entre 0 e 1, então apenas 1,05 % das possíveis soluções resultam em valores da função maiores que 0,9

1.2.5. Implementação Computacional

Exemplo 2.4.

- Algoritmo Genético

- Função de avaliação

- $fitness(z_i) = (z_i / z_{max})^{10}$

- Varia entre 0 e 1

- Decodificação

- Vetor de bits $\mathbf{x}_i$ com dimensão 30

- Exemplo: $\mathbf{x}_i = [1 \ 0 \ 1 \ \dots \ 0]^T$

- O vetor de bits é decodificado para números inteiros

- $z_i = x_i(1) 2^0 + x_i(2) 2^1 + x_i(3) 2^2 + \dots + x_i(30) 2^{29}$

1.2.5. Implementação Computacional

Exemplo 2.4.

- AG: função de avaliação

/* Calculo da função de avaliação $f(z) = (z/z_{max})^{10}$ */

função calcfitness(crom : cromossomo ; lcrom : inteiro) : real

declare zmax , z : real

zmax ← 1073741823.0

z = decodificação(crom , lcrom)

retorne (potência(z / zmax , 10))

// fator de normalização

// o máximo valor de z é $2^{30} - 1$ (ver próx. slide)

// decodifica o vetor de bits em um valor int

// fenótipo (pode assumir outros tipos)

1.2.5. Implementação Computacional

Exemplo 2.4.

- AG: decodificação

```
/* Decodifica um vetor de bits em um número inteiro */  
função decodificação( vetor : cromossomo ; lbits : inteiro ) : inteiro  
  
    declare j , valor , potdedois : inteiro  
    potdedois  $\leftarrow$  1  
    valor  $\leftarrow$  0  
    para j  $\leftarrow$  1 até j  $\leftarrow$  lbits  
        se ( vetor[ j ] )  
            valor  $\leftarrow$  valor + potdedois  
        fim se  
        potdedois  $\leftarrow$  2*potdedois  
    fim para  
    retorne ( valor )
```

1.2.5. Implementação Computacional

Exemplo 2.4.

- Ver [Goldberg, 1989]
- AG: parâmetros
 - $p_{optam} = 30$
 - $p_{cross} = 0,6$
 - $p_{mutação} = 0,0333$

How Well Does It Work?

73

Population Report									
Generation 0					Generation 1				
#	string	x	fitness		# parents xsite	string	x	fitness	
1)	11100001100110000010111110100	9.4621E+08	0.2824		1) (1,19)	30	11100001100110000010110110100	9.4621E+08	0.2824
2)	110011001011010111000000100001	8.5862E+08	0.1069		2) (1,19)	30	110101011110001110010011000101	8.9712E+08	0.1658
3)	010101111001011110000010001101	3.6739E+08	0.0000		3) (23,19)	19	110101011100000011010110000001	8.9655E+08	0.1647
4)	011001111000011101101111011010	4.3423E+08	0.0001		4) (23,19)	19	111111001000000010010011000101	1.0591E+09	0.8715
5)	011111111010010101011010110110	5.3539E+08	0.0009		5) (19, 1)	11	111000011001100000110011000101	9.4621E+08	0.2824
6)	101101111001000011000101101101	7.6993E+08	0.0359		6) (19, 1)	11	110101011110000010001111110100	8.9707E+08	0.1657
7)	000110101111100001001011111000	1.1312E+08	0.0000		7) (16, 1)	6	111000011001100000101111100100	9.4621E+08	0.2824
8)	010100111010111010001111100010	3.5099E+08	0.0000		8) (16, 1)	6	110011010101101100011001110100	8.6132E+08	0.1103
9)	011011001110001010011110001011	4.5670E+08	0.0002		9) (23,17)	30	101111001000010011110110010001	7.9071E+08	0.0469
10)	010011010110101000001101011011	3.2470E+08	0.0000		10) (23,17)	30	110011101001000100011101100011	8.6640E+08	0.1170
11)	010111011010101011001101000010	3.9287E+08	0.0000		11) (6,26)	30	101101110001000011000101101101	7.6783E+08	0.0350
12)	010011010011001010110010001110	3.2379E+08	0.0000		12) (6,26)	30	110010000101010100110110011110	8.4026E+08	0.0861
13)	110000100101101110110000100111	8.1520E+08	0.0636		13) (2,19)	10	110101011110100010010000100001	8.9720E+08	0.1659
14)	010110001001111101000100110001	3.7171E+08	0.0000		14) (2,19)	10	110011011011010111000011000101	8.6281E+08	0.1122
15)	010110111110000101101110011010	3.8538E+08	0.0000		15) (17,17)	26	110011111001000100011100100011	8.7060E+08	0.1228
16)	110011010101100100011001100000	8.6129E+08	0.1103		16) (17,17)	26	110010111101000100011100100011	8.5487E+08	0.1023
17)	110011111101000100011100100011	8.7165E+08	0.1243		17) (19,17)	30	110101011110000010010011000111	8.9707E+08	0.1657
18)	100000000100010111100101011100	5.3802E+08	0.0010		18) (19,17)	30	110011111101000000011101000011	8.7163E+08	0.1243
19)	110101011110000010010011000101	8.9707E+08	0.1657		19) (19,19)	24	110101011110000010010011000101	8.9707E+08	0.1657
20)	010011111000010001000011011101	3.3352E+08	0.0000		20) (19,19)	24	110101011110000010010011000101	8.9707E+08	0.1657
21)	001101011100110111010010000011	2.2567E+08	0.0000		21) (26,17)	30	110010000101010100110111011110	8.4026E+08	0.0861
22)	000110011111000001100100110110	1.0880E+08	0.0000		22) (26,17)	30	110000111101000100011100100011	8.2132E+08	0.0686
23)	10111100100000001110110010001	7.9064E+08	0.0469		23) (23, 1)	3	111000011001100000001111110001	9.4621E+08	0.2824
24)	011101100001100010100101100111	4.9533E+08	0.0004		24) (23, 1)	3	101111001000000011110100010100	7.9064E+08	0.0469
25)	010110111010001101010001010010	3.8436E+08	0.0000		25) (27, 1)	10	110000011001100000101001110011	8.1199E+08	0.0612
26)	110010000101010100110110011110	8.4026E+08	0.0861		26) (27, 1)	10	10100110000101010100110110100	6.9660E+08	0.0132
27)	101001100101110101001001100011	6.9778E+08	0.0134		27) (1,17)	25	11001000100110000010111110100	8.4135E+08	0.0873
28)	100011110111010000100000110010	6.0169E+08	0.0031		28) (1,17)	25	111001111101000100011100100011	9.7231E+08	0.3707
29)	010010100010000100001101100011	3.1092E+08	0.0000		29) (1,19)	23	11010101000110000010111110100	8.9378E+08	0.1597
30)	001011001111001110010101100011	1.8854E+08	0.0000		30) (1,19)	23	111000011110000000010011000101	9.4739E+08	0.2859

Note: Generation 1 & Accumulated Statistics: max=0.8715, min=0.0132, avg=0.1732, sum=5.1967, nmutation=35, ncross= 10

FIGURE 3.16 SGA run, generation report $t = 0-1$.

Mapping Objective Functions to Fitness Form

75

Population Report					
Generation 6			Generation 7		
#	string	x fitness	#	parents xsite	string x fitness
1)	11110001101001001010111110001	1.0135E+09 0.5615	1) (8, 6)	7	111111001011000000101111000100 1.0599E+09 0.8779
2)	11111100100010001001111110011	1.0592E+09 0.8726	2) (8, 6)	7	111111001000000001001101101100 1.0591E+09 0.8715
3)	11111100100000001001111110100	1.0591E+09 0.8715	3) (9, 24)	9	111111001000000010011010100111 1.0591E+09 0.8715
4)	111110011110000000101111100100	1.0481E+09 0.7849	4) (9, 24)	9	111111010000000010000111110111 1.0675E+09 0.9430
5)	111111001101100001101111010110	1.0605E+09 0.8834	5) (6, 18)	3	11111101100000111010111100100 1.0633E+09 0.9070
6)	11111100101100000010111101100	1.0599E+09 0.8779	6) (6, 18)	3	11111000101100000011111101100 1.0431E+09 0.7484
7)	11111100100110000010111101100	1.0595E+09 0.8747	7) (10, 22)	22	10111001110000001001111110111 7.7910E+08 0.0405
8)	111111001000000010011011000100	1.0591E+09 0.8715	8) (10, 22)	22	11111100111101000010111100100 1.0610E+09 0.8872
9)	111111101000000010000011100111	1.0675E+09 0.9430	9) (15, 15)	30	11111001101000001001111110001 1.0470E+09 0.7772
10)	11111100110000001001111110111	1.0601E+09 0.8801	10) (15, 15)	30	11111001101000001001111111001 1.0470E+09 0.7772
11)	11111100100000001001111110111	1.0591E+09 0.8715	11) (12, 12)	5	1111111100000000001111010001 1.0716E+09 0.9807
12)	11111110100000001001111110001	1.0675E+09 0.9430	12) (12, 12)	5	11111110100000001001111110001 1.0675E+09 0.9430
13)	11101101100000001000001000101	9.9616E+08 0.4724	13) (26, 16)	15	11111100101000000010101110111 1.0596E+09 0.8757
14)	11111100100111000010111110000	1.0595E+09 0.8752	14) (26, 16)	15	11111001011000001010111100100 1.0460E+09 0.7694
15)	11111001101000001001111110001	1.0470E+09 0.7772	15) (20, 1)	30	11111100100000001001001000101 1.0591E+09 0.8715
16)	11111100101000001010111100100	1.0596E+09 0.8758	16) (20, 1)	30	11110001101001001010111110001 1.0135E+09 0.5615
17)	11111001111000001010011101111	1.0481E+09 0.7850	17) (9, 10)	30	111111010000000010011011100111 1.0675E+09 0.9430
18)	11111101100000111010111100100	1.0633E+09 0.9070	18) (9, 10)	30	11111100110000001001111110111 1.0601E+09 0.8801
19)	11111100100000001001111110000	1.0591E+09 0.8715	19) (3, 8)	14	11111101100000001001111110100 1.0633E+09 0.9066
20)	111111001000000010010011000101	1.0591E+09 0.8715	20) (3, 8)	14	111111001000000010011011000100 1.0591E+09 0.8715
21)	11111001111000001010111110000	1.0481E+09 0.7850	21) (1, 26)	8	111110010110000000001110110001 1.0460E+09 0.7694
22)	10111001111001001010111100100	7.7969E+08 0.0408	22) (1, 26)	8	11110001101001001010111111111 1.0135E+09 0.5615
23)	11111110100000001001111110001	1.0675E+09 0.9430	23) (18, 20)	30	11101101100000111010111110100 9.9621E+08 0.4726
24)	11111100100000001001111110111	1.0591E+09 0.8715	24) (18, 20)	30	111111001000000010010010000101 1.0591E+09 0.8715
25)	111111111000000010011111100100	1.0717E+09 0.9807	25) (14, 30)	3	11111100100110000010111100000 1.0595E+09 0.8747
26)	11111001011000000010111110111	1.0460E+09 0.7694	26) (14, 30)	3	11111100100111000010111010100 1.0595E+09 0.8752
27)	11111100100111000110111110000	1.0595E+09 0.8752	27) (23, 3)	18	11111100100100001001111110001 1.0593E+09 0.8736
28)	011110011000010010000011000101	5.0968E+08 0.0006	28) (23, 3)	18	11111110110000001001111110100 1.0685E+09 0.9523
29)	11111100110000001001101110001	1.0601E+09 0.8801	29) (24, 2)	30	11111100100001001000111110111 1.0591E+09 0.8720
30)	111111001001100000101111100100	1.0595E+09 0.8747	30) (24, 2)	30	11111100100010001001111110011 1.0592E+09 0.8726

Note: Generation 7 & Accumulated Statistics: max=0.9807, min=0.0405, avg=0.8100, sum=24.2997, nmutation=201, ncross= 71

FIGURE 3.18 SGA run, generation report $t = 6-7$.

Comentários

- Referências
 - Goldberg, D. E. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Pub. Co., 1989
 - Capítulo 3