**Instituto Politécnico de Tomar**



Optimização com Números Reais

Índice

[Função de avaliação 3](#_Toc326416054)

[Valor do indivíduo 4](#_Toc326416055)

[Seleção 4](#_Toc326416056)

[Recombinação 4](#_Toc326416057)

[Mutação 5](#_Toc326416058)

[Intermediate recombination 5](#_Toc326416059)

[Fluxogramas 7](#_Toc326416060)

[Crossover 8](#_Toc326416061)

[Mutação Binaria 9](#_Toc326416062)

[P.S.O 9](#_Toc326416063)

[Recombinação Binaria 10](#_Toc326416064)

[Pseudo-codigo 11](#_Toc326416065)

[Crossover 12](#_Toc326416066)

[Mutação binaria 12](#_Toc326416067)

[PSO 12](#_Toc326416068)

[Recombinacao\_binaria 13](#_Toc326416069)

# Função de avaliação

f(x1,x2)=21.5 + x1 m(4π\*x1)+x2 m(20 π\*x2)

com, -3,0 ≤ x1 ≤12,1 e 4,1 ≤ x2 ≤5,8

Representação binaria para 4 casas decimais:

Dx1=15,1=12,1 –(-3,0)

15,1 \*10 000 = 15100 217 ≤ 151000 ≤ 218

Dx2=1,7=5,8 – 4,1

1,7 \*10 000 =17000 214 ≤ 17000 ≤ 215

Indivíduo i é composto de 2 genes (x1 e x2)

X1 =18 alelos

X2 =15 alelos

Exemplo:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **0** |

X1= =250552d

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **1** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |  |  |

X2= =4800d

X´1=

X´2=

## Valor do indivíduo

Avaliação = f(11,4323 ; 4,3490) = 13,1733

## Seleção

Ordena-se a população (por exemplo: através da roleta) colocando neste caso (maximização) no topo dar lista os indivíduos que obtiverem maior valor calculado através da função de avaliação.

## Recombinação

1. Escolhem-se 2 indivíduos do topo da lista obtida (que ainda não tivessem sido escolhidos)
2. Calcula-se um numero aleatório entre [0;1[

3.1 Se o número aleatório for maior do que 0.65, os dois indivíduos escolhidos passam diretamente para a descendência.

3.2 Se o número aleatório for menor ou igual a 0.65, calcula-se aleatoriamente um ponto de corte para os primeiros genes dos dois indivíduos escolhidos recombinando-os e procedendo de igual forma para os segundos genes de ambos. Passando para a discordância os dois indivíduos cujos novos genes são constituídos até aos cortes, pelos seus próprios e são contas em diante pelos do indivíduos com o qual recombinaram.

4 Volta-se ao ponto 1 até terminassem os indivíduos da lista ordenada.

Exemplo gene x1=a+b

Pontos de corte aleatória

gene x2=c+d

Indivíduos:i1 e i2

i1= x1 & x2 ⬄ i1= a+b & c+d

i2= x1 & x2 ⬄ i1= A+B & C+D

Recombinando fica:

i´1 =a +B & c +D

i´2 =A +b & C +d

## Mutação

1. É escolhida uma percentagem de mutação (por exemplo 1%)
2. É escolhido um individuo
3. É escolhido o primeiro gene: e para cada um dos alelos é calculada aleatoriamente uma percentagem, se essa percentagem for menos do que 1% o bit em causa passa a “1”.

3.1 É efetuada a mesma operação no segundo gene.

1. Volta-se ao ponto 2 até terminar a população.

Exemplo:

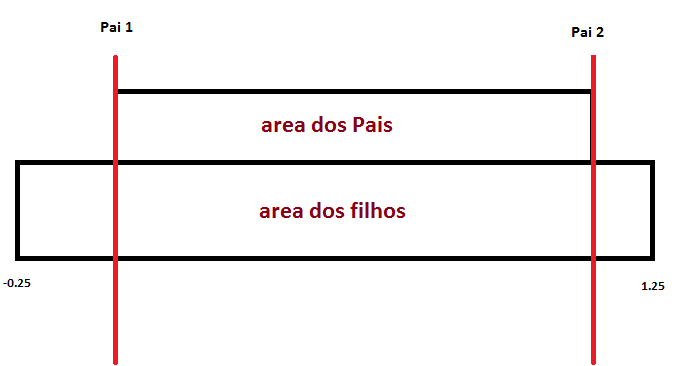
i´3= 101101101000111010 & 111001000101110

# Intermediate recombination

1. Selecionam-se 2 indivíduos, P1 e P2, assumindo o papel de “Pais” (P1 “Pai1” )
2. Selecionam-se os genes Genes iP1 e Genes iP2 , dos indivíduos P1 e P2 respetivamente , em que i (1,2,3…. numero genes)
3. Gera-se um número aleatório a; pertencente ao intervalo [ -0.25 ; 1.25 ]
4. Calcula-se o gene GenesiF do novo indivíduo F “filho ”, através da seguinte formula.

GenesiF = GenesiP1 \* ai+ GenesiP2 \* ai

1. Volta-se ao ponto 2 até que todos os genes dos indivíduos ” Pais ” P1 e P2 sejam percorridos (i=numero genes).



Exemplo

Considerar os seguintes indivíduos, com 3 genes cada

|  |  |  |
| --- | --- | --- |
| **12** | **25** | **5** |

Pai 1

|  |  |  |
| --- | --- | --- |
| **123** | **4** | **34** |

Pai 2

Genes1P2 Genes2P2  Genes3P2

Considerar os seguintes valores de a para esta exemplo

a1  a2 a3

0.5 1.1 -0.1

O novo filho calculado

Genes1F Genes2F  Genes3F

|  |  |  |
| --- | --- | --- |
| **67.5** | **1.9** | **2.1** |

Pai 2

GenesiF = GenesiP1 \* ai+ GenesiP2 \* (1- ai)

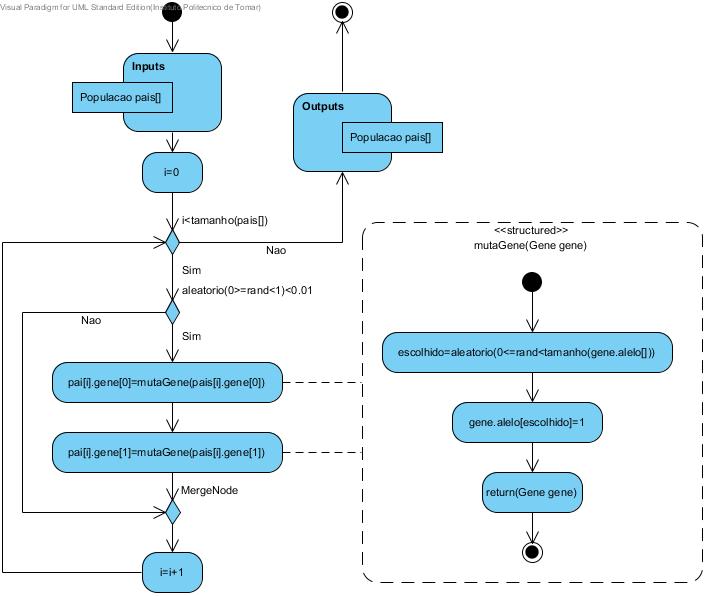
=12 \* 0.5 + 123 \* (1-0.5)

= 6 + 61.5

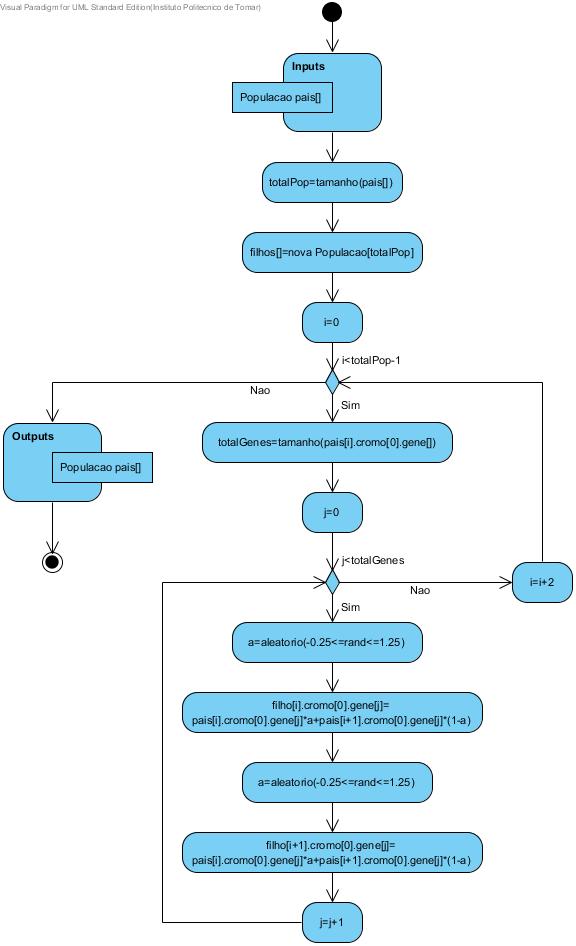
=67.5

# Fluxogramas

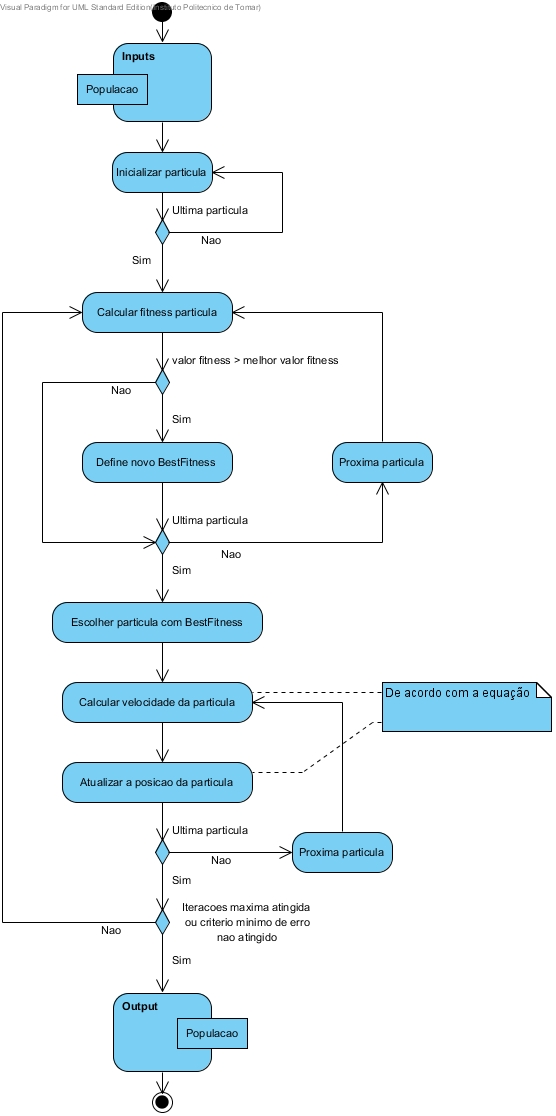
## Crossover



## Mutação Binária



## P.S.O.



## Recombinação Binaria



# Pseudo-codigo

## Crossover

entra(Populacao pais[])

totalPop := tamanho(pais[])

filhos[] := nova Populacao[totalPop]

para i := 0 ate i < totalPop-1 fazer

totalGenes := tamanho(pais[i].cromo[0].gene[])

para j := 0 ate j < totalGenes fazer

a = aleatorio(-0.25<=rand<=1.25)

filho[i].cromo[0].gene[j] := pais[i].cromo[0].gene[j] \* a + pais[i+1].cromo[0].gene[j] \* (1-a)

a = aleatorio(-0.25<=rand<=1.25)

filho[i+1].cromo[0].gene[j] := pais[i].cromo[0].gene[j] \* a + pais[i+1].cromo[0].gene[j] \* (1-a)

j := j+1

i := i+2

sai(Populacao pais[])

## Mutação binaria

entra(Populacao pais[])

para i := 0 até i < tamanho(pais[]) //percorre a populaçao progenitora

se (aleatorio(0<=rand<1) < 0.01) entao //se factor de ponderaçao for menor,

pais[i].gene[0] := mutaGene(pais[i].gene[0]) //que 1% entao

pais[i].gene[1] := mutaGene(pais[i].gene[1]) //muta o alelo em causa

i := i+1 //nova iteraçao

sai(Populacao pais[])

mutaGene(Gene gene)

escolhido := aleatorio(0<=rand<tamanho(gene.alelo[])) //alelo escolhido aleatoriamente

gene.alelo[escolhido] := 1 //toma o valor de verdadeiro

retorna(Gene gene)

## PSO

entra(Populacao pop[])

Para cada pop[].partícula

Inicializar pop[i].partícula

Fazer

Para cada pop[].partícula

calcularValor(pop[i].partícula.fitness)

Se (valor de fitness > melhor valor de fitness (pBest))

definaValor(pBest)

gBest := partícula com o melhor valor de fitness

Para cada pop[].partícula

v[] := v[] + c1 \* rand() \* (pbest[] - present[]) + c2 \* rand() \* (gbest[] - present[])

present[] := persent[] + v[]

Enquanto (iterações máximas não forem atingidas ou critérios mínimos erro não forem atingidos)

sai(Populacao pop[])

## Recombinacao\_binaria

entra(Populacao pais[])

dim := tamanho(pais[]) //numero de indivíduos da população progenitora

dimGene0 := tamanho(pais[].gene[0].alelo[]) //dimensão do primeiro gene

dimGene1 := tamanho(pais[].gene[1].alelo[]) //dimensão do segundo gene

Populacao filhos[] := nova Populacao[dim] //criação de uma geração vazia do tamanho da população progenitora

para i := 0 até i < dim //percorrer a população progenitora

pais[i].fitness := fitness(pais[i]) //calculo do fitness dos indivíduos da população progenitora

i := i+1 //nova iteração

pais[] := ordenaDecrescente(pais[].fitness) //ordena a população progenitora pelo valor do fitness decrescente

para i := 0 até i < dim-1 //ciclo para recombinação dos indivíduos

se (aleatorio(0<=rand<=1) > 0.65) entao //se factor de ponderação for maior,

filhos[i] := pais[i] // então os indivíduos transitam como estão

filhos[i+1] := pais[i+1] //

senao //se factor de ponderação não for maior,

corte := aleatorio(1<=rand<=dimGene0-1) //calculo do ponto de corte do primeiro gene

para j := 0 até j < corte //até ao corte do primeiro gene

filhos[i].gene[0].alelo[j] := pais[i].gene[0].alelo[j] //copia dos pais para os filhos

filhos[i+1].gene[0].alelo[j] := pais[i+1].gene[0].alelo[j] //

j := j+1 //nova iteração

para k := corte até k < dim //depois do corte do primeiro gene

filhos[i].gene[0].alelo[k] := pais[i+1].gene[0].alelo[k] //copia dos pais para os filhos,

filhos[i+1].gene[0].alelo[k] := pais[i].gene[0].alelo[k] //com os progenitores trocados

k := k+1 //nova iteração

corte := aleatorio(1<=rand<=dimGene1-1) //calculo do ponto de corte do segundo gene

para m := 0 até m < corte //até ao corte do segundo gene

filhos[i].gene[1].alelo[m] := pais[i].gene[1].alelo[m] //copia dos pais para os filhos

filhos[i+1].gene[1].alelo[m] := pais[i+1].gene[1].alelo[m] //

m := m+1 //nova iteração

para n := corte até n < dim //depois do corte do segundo gene

filhos[i].gene[1].alelo[n] := pais[i+1].gene[1].alelo[n] //copia dos pais para os filhos

filhos[i+1].gene[1].alelo[n] := pais[i].gene[1].alelo[n] //com os progenitores trocados

n := n+1 //nova iteração

i := i+2 //nova iteração com um incremento de 2

sai(Populacao filhos[])