

---

# Algoritmos Genéticos Capítulo 4

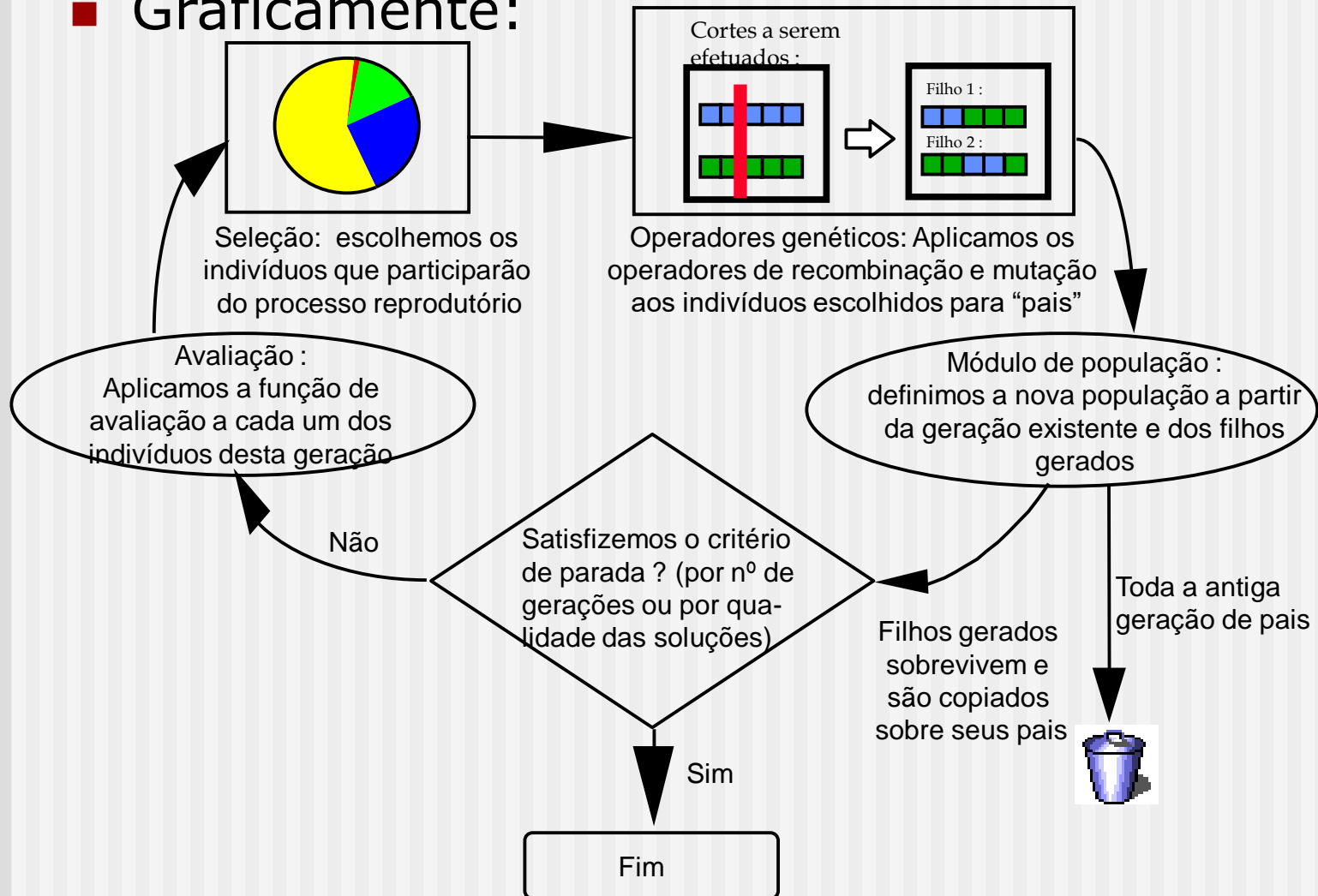
Ricardo Linden

# Esquema de um GA

- Algoritmos Genéticos são um ramo da computação evolucionária
- Seu funcionamento pode ser resumido algoritmicamente através dos seguintes passos:
- Inicialize a população de cromossomos
  - a) Avalie cada cromossomo na população
  - b) Selecione os pais para gerar novos cromossomos.
  - c) Aplique os operadores de recombinação e mutação a estes pais de forma a gerar os indivíduos da nova geração
  - d) Apague os velhos membros da população
  - e) Avalie todos os novos cromossomos e insira-os na população
  - f) Se o tempo acabou, ou o melhor cromossomo satisfaz os requerimentos e desempenho, retorne-o, caso contrário volte para o passo c).

# Esquema de um GA

## ■ Graficamente:



# Esquema de um GA

---

- Esta é somente uma visão de alto nível de nosso algoritmo.
- O que ela esconde é a complexidade do processo de obtenção dos seguintes elementos
  - uma representação cromossomial que seja adequada ao problema
  - uma função de avaliação que:
    - penalize soluções implausíveis para nosso problema
    - avalie satisfatoriamente o grau de adequação de cada indivíduo como solução do problema em questão.

# Esquema de um GA

---

- Um GA é altamente genérico.
- Vários de seus componentes são invariáveis de um problema para outro.
- Isto favorece sua implementação em uma linguagem orientada a objeto, permitindo o reaproveitamento do código para solução de vários problemas diferentes.

# Representação Cromossomial

---

- A representação cromossomial é fundamental para o nosso algoritmo genético.
- Ela consiste em uma maneira de traduzir a informação do nosso problema em uma maneira viável de ser tratada pelo computador.
- Quanto mais ela for adequada ao problema, maior a qualidade dos resultados obtidos.
- Resista à tentação de adequar o problema à sua representação!

# Representação Cromossomial

---

- Cada pedaço indivisível desta representação é chamado de um gene.
- É importante notar que a representação cromossomial é completamente arbitrária.
- É interessante apenas que algumas regras gerais sejam seguidas :
  - a) A representação deve ser a mais simples possível
  - b) Se houver soluções proibidas ao problema, então elas não devem ter uma representação
  - c) Se o problema impuser condições de algum tipo, estas devem estar implícitas dentro da nossa representação.

# Representação Cromossomial

---

- No capítulo 10 veremos outros exemplos de como estas regras podem ser seguidas.
- Neste momento, vamos adotar a representação binária.
  - Mais simples e mais usada pelos praticantes da área dos algoritmos genéticos.
  - Um cromossomo nada mais é do que uma sequência de bits  $C$
  - Cada gene é somente um bit.
  - O conceito representado por cada bit e/ou conjunto de bits é inerente ao problema.



# Representação Cromossomial

---

- Essa representação foi a adotada inicialmente por Holland, em seu livro seminal .
- Hoje em dia, por estes motivos históricos e pelo fato de ser muito simples, ela é amplamente adotada por pesquisadores da área de GA.
- Os operadores genéticos, como discutiremos a seguir, são compreensíveis e implementáveis.

# Representação Cromossomial

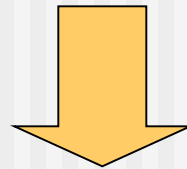
- Representamos números (inteiros ou reais), strings, etc. usando a representação binária.
- Para representar números reais como números binários, temos primeiro que saber duas coisas:
  - A faixa de operação de cada uma das variáveis;
  - A precisão desejada.
- Sabendo isto, convertemos bits para números usando a seguinte fórmula:

$$real = \inf_i + \frac{\sup_i - \inf_i}{2^k - 1} * r_i$$

# Representação Cromossomial

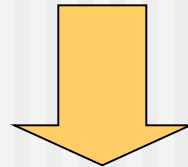
- Exemplo de interpretação de números reais:

$$\underbrace{000011}_{x_1} \mid \underbrace{110011}_{x_2}$$



$$r_1 = 000011 = 3$$

$$r_2 = 110011 = 51$$



$$x_1 = -2 + 3 * (2 - (-2)) / (2^6 - 1) = -1,809$$

$$x_2 = 0 + 51 * (1 - 0) / (2^6 - 1) = 0,809$$

# Inicializando um elemento

---

```
private void inicializaElemento(int tamanho) {  
    int i;  
    this.valor="";  
    for(i=0;i<tamanho;++i) {  
        if (java.lang.Math.random()<0.5) {  
            this.valor=this.valor+"0";  
        } else {  
            this.valor=this.valor+"1";  
        }  
    }  
}
```

# Inicializando toda a população

---

```
public void inicializaPopulacao(int tamanho) {  
    int i;  
    this.populacao=new Vector();  
    for(i=0;i<tamanho;++i) {  
        this.populacao.add(new ElementoGA1());  
    }  
}
```

# Função de Avaliação

---

- A função de avaliação é a maneira utilizada pelos GAs para determinar a qualidade de um indivíduo como solução do problema em questão.
- É uma nota dada ao indivíduo na resolução do problema.
- Será usada para a escolha dos indivíduos pelo módulo de seleção de pais, sendo a forma de diferenciar entre as boas e as más soluções para um problema.

# Função de Avaliação

---

- Dada a generalidade dos GAs, a função de avaliação, em muitos casos, é a única ligação verdadeira do programa com o problema real.
- Mesmo GA pode ser usado para descobrir o máximo de toda e qualquer função de  $n$  variáveis sem nenhuma alteração das estruturas de dados e procedimentos adotados, alterando-se, apenas, a função de avaliação.
- Por isto, classes básicas definidas para este livro são abstratas, sendo a função de avaliação um método abstrato.

# Função de Avaliação

---

- Também chamada de função de custo
- Calcula então um valor numérico que reflete quão bons os parâmetros representados no cromossomo resolvem o problema.
- Usa todos os valores armazenados no cromossomo (os parâmetros) e retorna um valor numérico, cujo significado é uma métrica da qualidade da solução obtida usando-se aqueles parâmetros.
- A função de avaliação deve ser tal que se o cromossomo **c1** representa uma solução melhor do que o cromossomo **c2**, então a avaliação de **c1** deve ser maior do que a avaliação de **c2**.



# Função de Avaliação

---

- A função de avaliação deve portanto ser escolhida com grande cuidado.
- Deve embutir todo o conhecimento que se possui sobre o problema a ser resolvido, tanto suas restrições quanto seus objetivos de qualidade.
- Quanto mais conhecimento embutirmos em um GA, menos serão válidas as críticas sobre eles serem algoritmos genéricos
- Deve diferenciar entre duas soluções sub-ótimas, deixando claro qual delas está mais próxima da solução procurada.

# Seleção de Pais

---

- O método de seleção de pais deve simular o mecanismo de seleção natural:
  - Pais mais capazes geram mais filhos;
  - Pais menos aptos também podem gerar descendentes.
- Temos que privilegiar os indivíduos com função de avaliação alta, sem desprezar completamente aqueles indivíduos com função de avaliação extremamente baixa;
- Até indivíduos com péssima avaliação podem ter características genéticas que sejam favoráveis à criação de um indivíduo ótimo;
- Estas características podem não estar presentes em nenhum outro cromossomo.

# Seleção de Pais

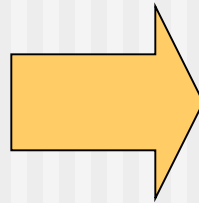
---

- Método simples e muito adotado: método da roleta viciada.
  - Criamos uma roleta (virtual) na qual cada cromossomo recebe um pedaço proporcional à sua avaliação (a soma dos pedaços não pode superar 100%).
  - Rodamos a roleta
  - Selecionado será o indivíduo sobre o qual ela parar.

# Seleção de Pais

## ■ Exemplo:

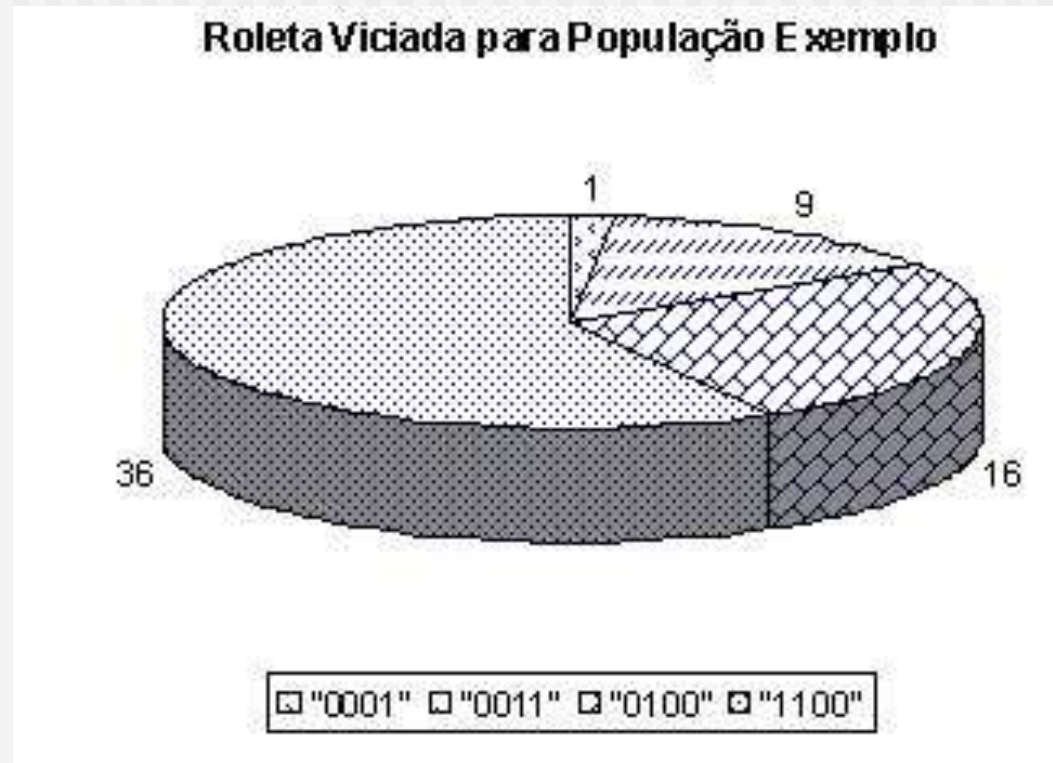
<i>Indivíduo</i>	<i>Avaliação</i>
0001	1
0011	9
0100	16
0110	36
<i>Total</i>	<i>62</i>



<i>Pedaço da roleta (%)</i>	<i>Pedaço da roleta (°)</i>
1.61	5.8
14.51	52.2
25.81	92.9
58.07	209.1
<i>100.00</i>	<i>360.0</i>

# Seleção de Pais

- Exemplo (cont.) – Graficamente, temos:



# Seleção de Pais

- Não podemos girar uma roleta dentro do computador
- Trabalhamos com conceitos abstratos, e não roletas físicas.
- Algoritmo:
  - a) Some todas as avaliações para uma variável soma*
  - b) Selecione um número  $s$  entre 0 e soma (Não incluídos)*
  - c)  $i=1$*
  - d)  $aux=$ avaliação do indivíduo 1*
  - e) enquanto  $aux < s$* 
    - f)  $i = i + 1$*
    - g)  $aux=aux+$ avaliação do indivíduo  $i$*
  - h) fim enquanto*

# Seleção de Pais

## ■ Implementação em Java:

```
1. public int roleta() {
2.     int i;
3.     double aux=0;
4.     calculaSomaAvaliacoes();
5.     double limite=Math.random()*this.somaAvaliacoes;
6.     for(i=0;
           ((i<this.populacao.size()) && (aux<limite))
           ;++i) {
7.         aux+=( (ElementoGA)populacao.get(i)).getAvaliacao();
8.     }
9.     i--;
10.    return(i);
11. }
```

# Observação

---

- Todas as avaliações devem ser estritamente positivas;
- Se tivéssemos um ou mais indivíduos com avaliação negativa, a soma total ainda seria  $360^\circ$ ;
- Entretanto, a soma dos espaços alocados apenas para os de avaliação positiva excederia  $360^\circ$ ;
- Teríamos que lidar com o problema de alocar um espaço negativo para o indivíduo com avaliação negativa;
- Indivíduos com avaliação igual a zero nunca seriam selecionados.



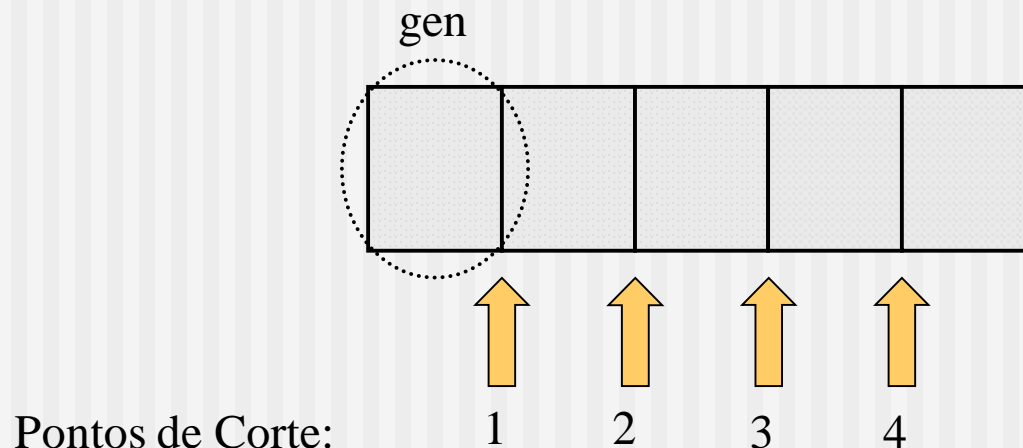
# Operadores de Crossover e Mutação

---

- Iremos trabalhar agora com a versão mais simples dos operadores genéticos
- Nesta versão, eles atuam em conjunto, como se fossem um só.
- Depois veremos versões mais avançadas.

# Operador de Crossover

- Vamos começar com o operador de crossover mais simples, chamado de operador de crossover de um ponto.
- Depois de selecionados dois pais pelo módulo de seleção de pais, um ponto de corte é selecionado.
- Um ponto de corte constitui uma posição entre dois genes de um cromossomo.
- Cada indivíduo de  $n$  genes contem  $n-1$  pontos de corte.



# Operador de Crossover

---

- Depois de sorteado o ponto de corte, nós separamos os pais em duas partes: uma à esquerda do ponto de corte e outra à direita.
- É importante notar que não necessariamente estas duas partes têm o mesmo tamanho.
- O primeiro filho é composto através da concatenação da parte esquerda do primeiro pai com a parte direita do segundo pai.
- O segundo filho é composto através da concatenação das partes que sobraram (a metade esquerda do segundo pai com a metade à direita do primeiro pai).

# Operador de Mutação

---

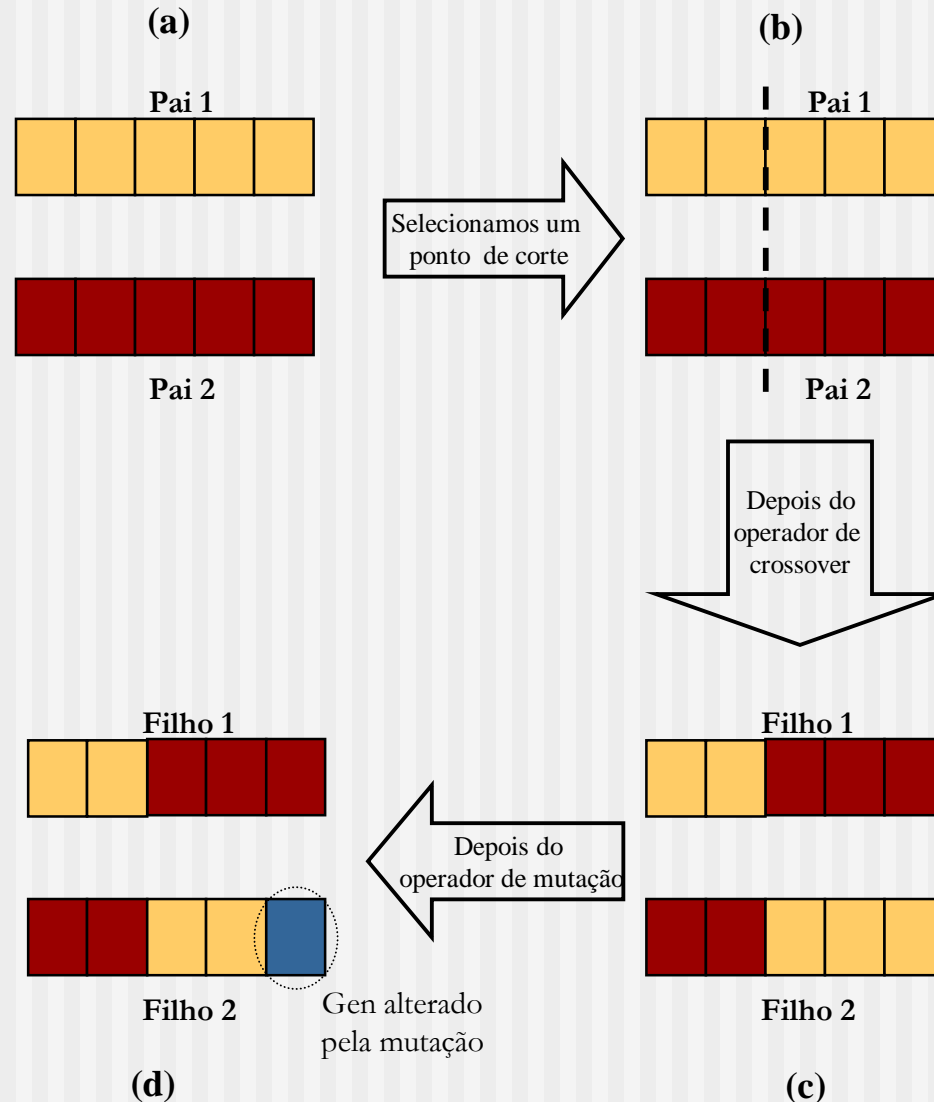
- Depois de compostos os filhos, entra em ação o operador de mutação.
- Este opera da seguinte forma:
  - Ele tem associada uma probabilidade extremamente baixa (da ordem de 0,5%);
  - Nós sorteamos um número entre 0 e 1.
  - Se ele for menor que a probabilidade pré-determinada então o operador atua sobre o gene em questão, alterando-lhe o valor aleatoriamente.
  - Repete-se então o processo para todos os gens componentes dos dois filhos.

# Comentários

---

- Valor da probabilidade deve ser baixo.
  - Se ele for muito alto, o algoritmo genético se parecerá muito com uma técnica chamada "random walk"
- Alguns textos preferem que o operador de mutação não aja de forma aleatória, mas sim, alterando o valor do gene para outro valor válido do nosso alfabeto genético.
  - Corresponde em multiplicar a probabilidade do operador de mutação por  $n/(n-1)$ , onde  $n$  é a cardinalidade do alfabeto genético.

# Juntando os operadores



# Módulo de População

---

- O módulo de população é responsável pelo controle da nossa população.
- Por simplicidade, população não pode crescer
  - permite que armazenemos a população em um vetor de tamanho constante.
- Pais têm que ser substituídos conforme os filhos vão nascendo
  - Pode parecer estranho, visto que estamos acostumados a ver a população humana sempre crescendo.
  - Quando nasce um bebê, não é obrigatório que alguém de alguma geração anterior caia fulminado!
  - Entretanto, simula bem ambientes de recursos limitados

# Módulo de População

---

- O módulo de população que utilizaremos por enquanto é extremamente simples.
- Sabemos que a cada atuação do nosso operador genético estamos criando dois filhos.
- Estes vão sendo armazenados em um espaço auxiliar até que o número de filhos criado seja igual ao tamanho da nossa população.
- Neste ponto o módulo de população entra em ação.
- Todos os pais são então descartados e os filhos copiados para cima de suas posições de memória, indo tornar-se os pais da nova geração.



# Execução Manual (1)

- Vamos tentar resolver, usando um GA, o problema de maximizar a função do exemplo 4.1, dada por , com  $x$  e  $y$  pertencentes ao intervalo  $[0,15]$ .

$$f(x, y) = \left| x * y * \text{sen}\left(\frac{y\pi}{4}\right) \right|$$

- Como é possível que esta função retorne um valor igual a zero, usaremos uma função de avaliação

$$g(x, y) = 1 + f(x, y)$$

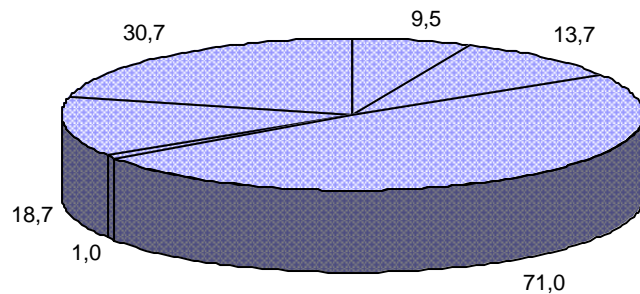
# Execução Manual (2)

- População inicial, sorteada aleatoriamente:

<b>Cromossomo</b>	<b>x</b>	<b>y</b>	<b>g(x,y)</b>
01000011	4	3	9,5
00101001	2	9	13,7
10011011	9	11	71,0
00001111	0	15	1,0
10011001	5	5	18,7
11100011	14	3	30,7
<i>Somatório das avaliações:</i>			144,6

# Execução Manual (3)

## ■ Roleta completa



### Intervalos para função de seleção

Cromossomo	g(x,y)	Intervalo
01000011	9,5	[0; 9,5[
00101001	13,7	[9,5; 23,2[
10011011	71,0	[23,2; 94,2[
00001111	1,0	[94,2; 95,2[
10011001	18,7	[95,2; 113,9[
11100011	30,7	[113,9; 144,6[

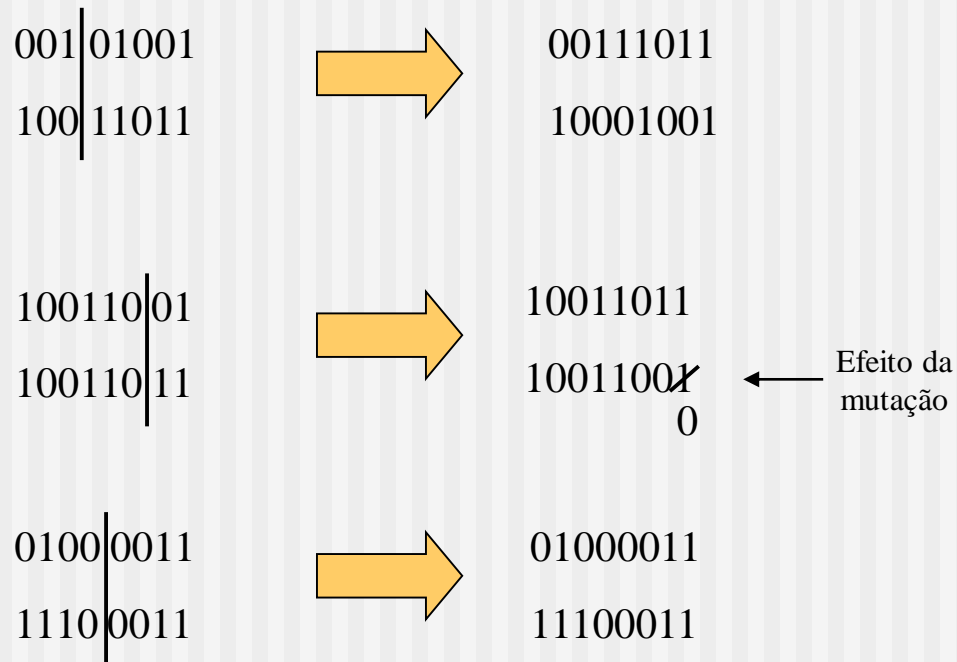
# Execução Manual (4)

## ■ Sorteio de Pais

Número Sorteado	Cromossomo Escolhido
12,8	00101001
65,3	<i>10011011</i>
108,3	10011001
85,3	<i>10011011</i>
1,8	01000011
119,5	11100011

# Execução Manual (5)

## ■ Operadores:



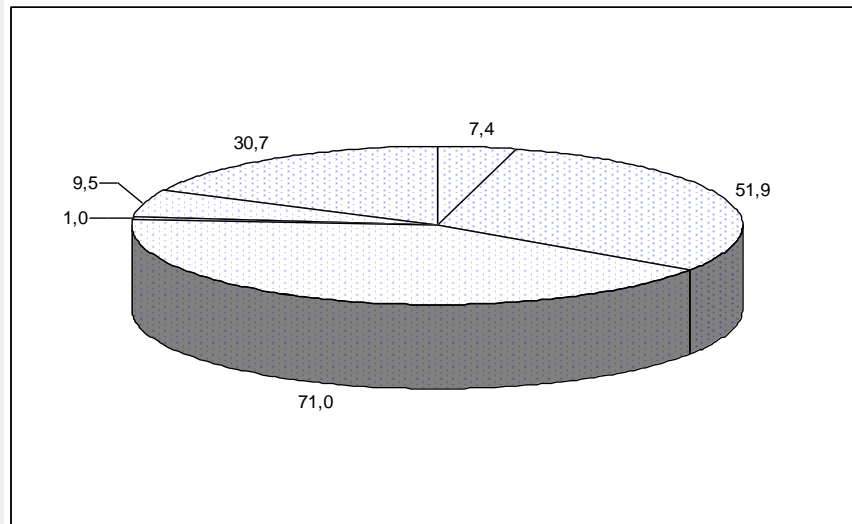
# Execução Manual (6)

## ■ Nova geração:

<b>Cromossomo</b>	<b>x</b>	<b>y</b>	<b>g(x,y)</b>
00111011	3	3	7,4
10001001	8	9	51,9
10011011	9	11	71,0
10011000	9	8	1,0
01000011	4	3	9,5
11100011	14	3	30,7
<i>Somatório das avaliações:</i>			171,5

# Execução Manual (7)

## ■ Nova roleta:



### Intervalos para função de seleção

Cromossomo	g(x,y)	Intervalo
00111011	7,4	[0; 7,4[
10001001	51,9	[7,4; 59,3[
10011011	71,0	[59,3; 130,3[
10011000	1,0	[130,3; 131,3[
01000011	9,5	[131,3; 140,8[
11100011	30,7	[140,8; 171,5[

# Execução Manual (8)

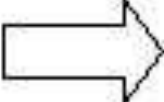
## ■ Sorteio de Pais

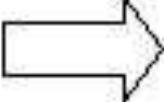
Número Sorteado	Cromossomo Escolhido
10,4	10001001
132,5	01000011
61,2	10011011
148,6	11100011
129,7	10011011
75,2	10011011

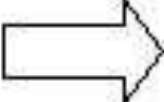


# Execução Manual (9)

## ■ Operadores:

1 0001001		00001001
0 1000011		11000011

1001 1011		10010011
1110 0011		11101011

1001101 1		10011011
1001101 1		10011011

# Execução Manual (10)

---

- Comentários:
  - Pais da terceira reprodução são iguais, um ao outro.
  - Logo, qualquer ponto de corte que seja selecionado para ambos gerará filhos iguais.
  - População perde diversidade, pois agora temos apenas 5 indivíduos diferentes, contra seis da geração anterior.
  - Este efeito, de convergência genética, é muito comum em populações que realizam cruzamentos endógenos.

# Execução Manual (11)

## ■ Nova geração:

<b>Cromossomo</b>	<b>x</b>	<b>y</b>	<b>g(x,y)</b>
00001001	0	9	1,0
11000011	3	3	7,4
10010011	9	3	20,1
11101011	14	11	109,9
10011011	9	11	71,0
10011011	9	11	71,0
<i>Somatório das avaliações:</i>			280,4

# Execução Manual (12)

---

- Neste momento você poderia achar que o algoritmo só funcionou porque o sorteio foi direcionado
- Esta é uma dúvida extremamente razoável neste ponto
- Só será apagadase executar os códigos deste capítulo e ver que tudo que fizemos aqui realmente acontece.

# Observação

---

- Existem várias melhorias possíveis no nosso GA;
- Vamos vê-las aos poucos, nos próximos capítulos.