Computação Evolucionária (ou Evolutiva)

- Métodos de Resolução de Problemas
  - **Métodos Fortes**: são concebidos para resolverem problemas genéricos, mas foram desenvolvidos para operarem em um mundo específico, onde impera linearidade, continuidade, diferenciabilidade e/ou estacionariedade.
  - **Métodos Específicos**: são concebidos para resolverem problemas específicos em mundos específicos.

Computação Evolucionária (ou Evolutiva)

- Métodos de Resolução de Problemas
  - **Métodos Fracos**: são concebidos para resolverem problemas genéricos em mundos genéricos. Operam em mundos não-lineares e não-estacionários, embora não garantam eficiência total na obtenção da solução. No entanto, geralmente garantem a obtenção de uma "boa aproximação" para a solução, em um tempo que aumenta a uma taxa menor que exponencial com o aumento do "tamanho" do problema.

Computação Evolucionária (ou Evolutiva)

- Independente da aplicação, métodos fracos (soluções baseadas em computação evolutiva) devem ser considerados se e somente se métodos fortes (soluções clássicas) e métodos específicos (soluções dedicadas) não existem, não se aplicam, ou falham quando aplicados.
- **conclusão:** soluções baseadas em computação evolutiva devem ser consideradas como o último recurso (reforçado ainda mais pela atual imaturidade desta área de pesquisa).
- **alento:** subtraindo-se os problemas tratáveis pelos métodos fortes e métodos específicos, o campo de aplicação para técnicas de computação evolutiva continua sendo extremamente vasto (nem tão vasto assim se considerarmos as restrições impostas pelo seu nível de maturidade).

Computação Evolucionária (ou Evolutiva)

"...Se **variações úteis** para qualquer organismo devam ocorrer para que ele venha a existir, certamente indivíduos assim caracterizados terão a **melhor chance** de serem preservados na luta por sobrevivência; e do forte princípio de **hereditariedade**, eles tenderão a produzir gerações com características similares. Este princípio de **preservação**, eu batizei, para ser sucinto, de **Seleção Natural**."

(Darwin, 1859)

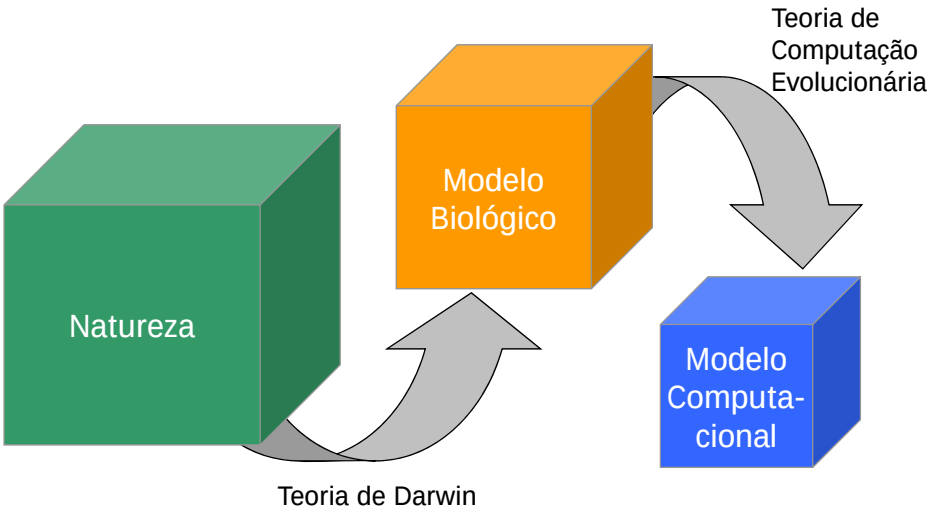
Computação Evolucionária (ou Evolutiva)

- Inspirada nos processos subjacentes à evolução:
  - Moldar uma população de indivíduos através da sobrevivência de seus membros mais ajustados.
  - Pressões seletivas não surgem apenas do ambiente externo, mas também de interações entre membros de uma população.

Computação Evolucionária (ou Evolutiva)

- Algoritmos genéticos, sistemas classificadores, estratégias evolutivas, programação genética e programação evolutiva
  - A computação evolutiva pode desempenhar os seguintes papéis básicos:
    - 1. ferramenta adaptativa para a solução de problemas;
    - 2. modelo computacional de processos evolutivos naturais.
- **Sistemas Naturais:** metáfora diretora, fonte de inspiração.

Computação Evolucionária (ou Evolutiva)



Fundamentos dos AGs

- Os algoritmos genéticos e outras analogias evolucionárias formais produzem soluções para problemas com capacidade crescente, operando sobre populações de soluções candidatas para o problema.
- **Uma Definição**
  - Algoritmos Genéticos (Ags) são métodos computacionais de busca baseados nos mecanismos de evolução natural e na genética. Em Ags, uma população de possíveis soluções para o problema em questão evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua.

Fundamentos dos AGs

- Foram desenvolvidos por John Holland, University of Michigan (1970's) com o objetivo de:
  - Entender os processos adaptativos dos sistemas naturais
  - Projetar sistemas artificiais (software) que possuíssem a robustez dos sistemas naturais
- Provêem uma técnica eficiente e efetiva para otimização e aplicações de aprendizado de máquina
- Largamente utilizados atualmente em negócios e problemas científicos e de engenharia

Componentes de um AG

Um problema a resolver, e ...

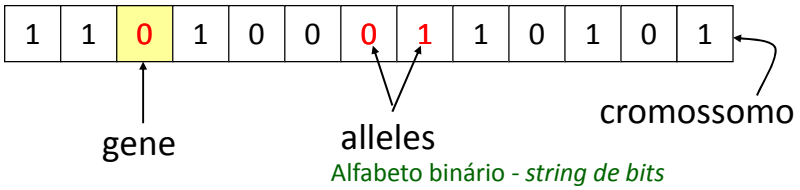
- Técnica de codificação (*gene, cromossomo*)
- Procedimento de Inicialização *(genesis)*
- Função de Avaliação *(meio ambiente)*
- Seleção de pais *(reprodução)*
- Operadores Genéticos *(mutação, recombinação)*
- Ajuste de Parâmetros *(prática e arte)*

Características Primárias

- AGs operam numa população (conjunto) de pontos, e não a partir de um ponto isolado.
- AGs operam num espaço de soluções codificadas, e não no espaço de busca diretamente.
- AGs necessitam somente de informação sobre o valor de uma função objetivo para cada membro da população, e não requerem derivadas ou qualquer outro tipo de conhecimento.
- AGs usam transições probabilísticas, e não regras determinísticas.

Representação Cromossômica

- Para poder aplicar AGs deve-se primeiramente representar cada possível solução ***x*** no espaço de busca como uma seqüência de símbolos ***s*** gerados a partir de um dado alfabeto finito A.
- Geralmente usa-se o alfabeto binário A={0,1}, mas no caso geral tanto o método de representação quanto o alfabeto dependem de cada problema.
- Cada seqüência ***s*** corresponde a um cromossomo.
- Cada elemento de ***s*** é equivalente a um gene.



Alfabeto binário - *string de bits*

Representação Cromossômica

Outras representações:

- com números reais: (43.2 -33.1 ... 0.0 89.2)
- *Strings* simbólicas: ◎⌘❁❖✡☯ ...
- Permutações de elementos (E11 E3 E7 ... E1 E15)
- Listas de regras (R1 R2 R3 ... R22 R23)
- Programas (para programação genética)
- Qualquer estrutura de dados

Representação Cromossômica

- Na maior parte dos AGs assume-se que cada indivíduo seja constituído de um único cromossomo.
- A grande maioria dos AGs propostos na literatura usam uma população de número fixo de indivíduos, com cromossomos também de tamanho constante.

Representação Cromossômica

- Para cada problema, encontrar uma representação cromossômica conveniente é sempre o primeiro passo.
- Usa-se um vetor binário para representar cada ponto no espaço de busca.
- Como há um número infinito de pontos no intervalo de interesse, a dimensão deste vetor ou seqüência binária depende da precisão requerida para o problema.

Representação Cromossômica

- Assumamos, como exemplo, uma precisão de 4 casas com um espaço de busca entre os números -2 até +2.
- Tal precisão significa que o processo de busca deve distinguir pelo menos 4 x 10.000 = 40.000 pontos.
- Como resultado, seqüências binárias (cromossomos) de 16 bits são necessários, uma vez que $40.000 < 2_{16} = 65.536$.
- Divide-se assim, o intervalo de busca em 65.536 partes iguais.

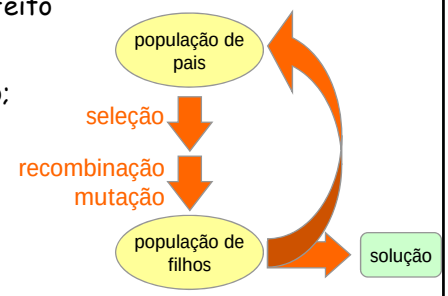
Representação Cromossômica

- Assim, cada possível solução x será representada por uma seqüência $s=[b_{15}b_{14}...b_2b_1b_0]$ onde cada $b \in \{0,1\}$.
- Primeiro converte-se o valor do cromossomo para um valor em base 10.
- Em seguida, o número é mapeado de volta ao espaço de busca de acordo com:

$$x = -x_{\min} + \frac{x_{\max} - x_{\min}}{2^{16} - 1} x$$

Fluxo Básico

```
{
    inicializar população;
    avaliar população;
    while
       Criterio_de_Termino_Nao_Satisfeito
    {
        selecionar pais para reprodução;
        realizar recombinação e mutação;
        avaliar população;
    }
}
```



- Implementações específicas do algoritmo particularizam esta estrutura de diferentes maneiras.

Fluxo Básico

- Inicialização
  - Geralmente a população inicial de N indivíduos é gerada aleatoriamente ou através de algum processo heurístico.
  - É importante que a população inicial cubra a maior parte possível do espaço de busca.

Fluxo Básico

- Avaliação e Adequabilidade
  - Ags necessitam da informação do valor de uma função objetivo para cada membro da população, que deve ser um valor não negativo.
  - Nos casos mais simples, usa-se o próprio valor da função que se quer maximizar.
  - A função objetivo dá, para cada indivíduo, uma medida de quão bem adaptado ao ambiente ele está.
  - A avaliação de cada indivíduo resulta num valor denominado de fitness.
  - Quanto maior o fitness, maiores são as chances do indivíduo sobreviver e se reproduzir.

Fluxo Básico

- Seleção
  - Emula os processos de reprodução assexuada e seleção natural.
  - Em geral, gera-se uma população temporária de N indivíduos extraídos com probabilidade proporcional ao fitness relativo de cada indivíduo no população.
  - A probabilidade de seleção de um cromossomo S é dada por:

$$P_{sel}(s) = \frac{f(s)}{\sum_{j=1}^{|\mathbf{P}|} f(s_j)}$$

Fluxo Básico

- Exemplo: Seleção dos pais

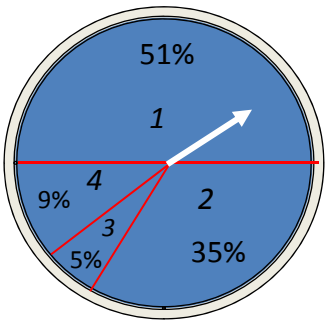
Chromosome	*Chromosome Value*	*Evaluation (fitness)*	*% of Total*
1	10111	201	51
2	11001	136	35
3	11110	20	5
4	00010	34	9
Total:		**391**	**100**

Fluxo Básico

- Seleção
  - Neste processo, indivíduos com baixo fitness (adequabilidade) terão alta probabilidade de desaparecerem da população (serem extintos).
  - Indivíduos adequados terão grandes chances de sobreviverem.
  - Como a seleção é probabilística, membros fracos recebem menores probabilidades, mas não são diretamente eliminados.
  - É importante que alguns candidato menos aptos sobrevivam, pois eles podem ainda conter algum componente essencial de uma solução.

Fluxo Básico

- Exemplo: Seleção dos pais
  - A seleção dos pais é realizada usando-se uma roleta com tamanhos de setores de acordo com os valores obtidos na avaliação dos cromossomos.

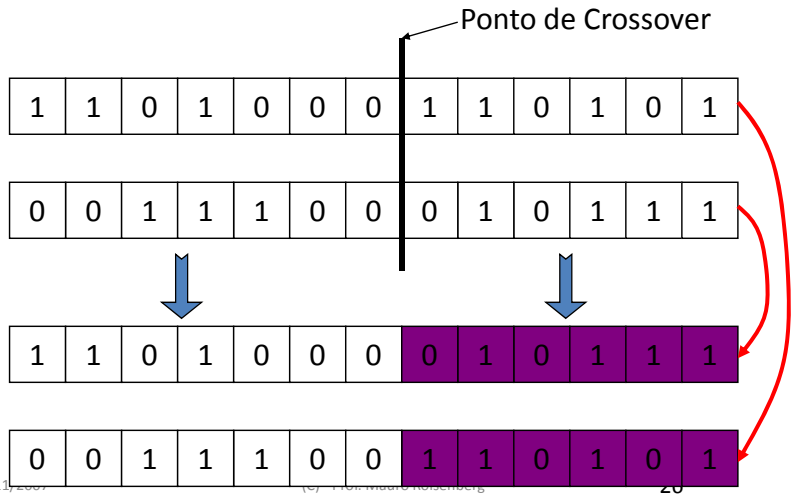


Fluxo Básico

- Reçombinação (crossover)
  - É um processo sexuado - ou seja, envolve mais de um indivíduo.
  - Emula o fenômeno de "crossover", a troca de fragmentos entre pares de cromossomos.
  - Na forma mais simples, é um processo aleatório que ocorre com probabilidade fixa p_{rec} que deve ser especificada pelo usuário.
  - O ponto de divisão pode ser ajustado aleatoriamente ou mudado durante o processo de solução.

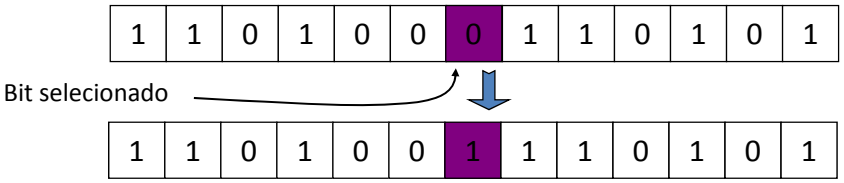
Fluxo Básico

- Recombinação (crossover)



Fluxo Básico

- Mutação
  - O processo de mutação é equivalente à busca aleatória.
  - Seleciona-se uma posição num cromossomo e muda-se o valor do gene correspondente aleatoriamente para um outro alelo possível.
  - O processo é geralmente controlado por um parâmetro fixo p_{mut} que indica a probabilidade de um gene sofrer mutação.



Fluxo Básico

- Mutação
  - É um processo importante, pois a população inicial pode excluir um componente essencial da solução.
  - O processo de reprodução, após uma série de iterações, tende a um equilíbrio estático. A mutação tende quebrar a estacionariedade, incorporando aspectos de criatividade no AG.
  - Se a mutação não gerar um bom resultado, a probabilidade de reprodução será bastante pequena, porém, sendo bom poderá trazer mudanças radicais no processo de busca da solução.
  - O mecanismo de mutação pode levar a resultados, que possivelmente não eram esperados.

Fluxo Básico

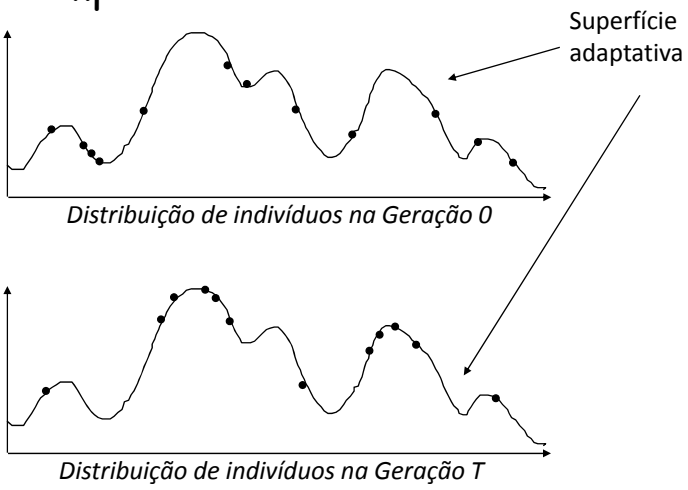
- Condições de Término
  - Como normalmente estamos tratando de problemas de otimização, o ideal seria que o algoritmo terminasse assim que o ponto ótimo fosse descoberto.
  - Pode haver situações onde todos ou o maior número possível de pontos ótimos sejam desejados.
  - Na prática raramente se pode afirmar se um dado ponto ótimo corresponde a um ótimo global.

Fluxo Básico

- Condições de Término
  - Normalmente usa-se o critério de número máximo de gerações ou um tempo limite de processamento para parar um AG.
  - Outro critério usa a idéia de estagnação, ou, seja, para-se o algoritmo quando não se observa melhoria da população depois de várias gerações.

Fluxo Básico

- Um exemplo abstrato



Exemplo

- Imagine uma população inicial de 4 indivíduos, cada um representado por uma cadeia de 10 bits.
- A função objetivo a maximizar é o número de bits em 1 em uma cadeia.
- Para normalizar a função objetivo a valores entre 0 e 1 divide-se por 10 o número de bits em 1 de cada indivíduo.

População P1:	
String (indivíduos)	Valor de Aptidão
0000011100	0,3
1000011111	0,6
0110101011	0,6
1111111011	0,9

Exemplo

- Na população P1 os quatro indivíduos possuem valor de aptidão 0.3, 0.6, 0.6 e 0.9.
- Teoricamente o mecanismo de seleção proporcional deveria alocar 12%, 25%, 25% e 38% de descendentes para cada indivíduo.
- Neste caso a alocação final de decendentes é 0, 1, 1 e 2 respectivamente.

População P2: Após a Seleção

String (indivíduos)	Valor de Aptidão
1000011111	0.6
0110101011	0.6
1111111011	0.9
1111111011	0.9

Exemplo

- A seguir as 4 cadeias são pareadas randomicamente para cruzamento.
- As cadeias 1 e 4 formam um par e as cadeias 2 e 3 outro par.
- Com uma taxa de cruzamento de 0.5, apenas as cadeias 1 e 4 são selecionadas para cruzamento, enquanto as cadeias 2 e 3 são deixadas intactas.
- O ponto de cruzamento cai entre o quinto e o sexto bit das cadeias.
- Os bits 1 a 5 das cadeias são trocados.

População P2: Após a Seleção

String (indivíduos)	Valor de Aptidão
1000011111	0.6
0110101011	0.6
1111111011	0.9
1111111011	0.9

População P3: Após o Cruzamento

String (indivíduos)	Valor de Aptidão
10000|11011	0.5
0110101011	0.6
1111111011	0.9
11111|11111	1.0

Exemplo

- A operação de mutação na população P3 pode ser visto na população P4, no bit 10 do indivíduo 4.
- Apenas um bit de um total de 40 foi trocado, representando uma taxa de mutação de 0.025.
- A população P4 representa a próxima geração.

População P4: Após a Mutação

String (indivíduos)	Valor de Aptidão
1000011011	0.5
0110101011	0.6
1111111011	0.9
0111111111	0.9

Exemplo

- Este exemplo tem caráter apenas ilustrativo.
- Um AG típico usa uma população de 30 a 200 indivíduos, taxas de cruzamento de 0.5 a 0.9 e taxas de mutação de 0.001 a 0.05.

Algoritmo Genético

- Implementam mecanismos de evolução natural incluindo *cruzamento, mutação e aptidão para sobrevivência.*
- Trabalha com populações de indivíduos.
- Os indivíduos são submetidos a um processo de evolução.

Programação Evolutiva

- Concebida como uma estratégia de otimização similar aos AG's.
- Proposta original trata de predição de comportamento de máquinas de estado finitos, porém se adapta a qualquer estrutura de problema.
- Enfatiza o relacionamento *comportamental* entre progenitores e sua descendência, ao invés de tentar emular operadores genéticos específicos observados na natureza.

Programação Evolutiva

- Não implementam cruzamentos. Ao invés disso, confiam na aptidão para sobrevivência e mutação.
- Cada indivíduo gera descendentes através de mutação, e a seguir a (melhor) metade da população ascendente e a (melhor) metade da população descendente são reunidas para formar a nova geração.
- Não se exige que a população permaneça constante, ou que cada ascendente gere apenas um descendente.

Algoritmos

ALGORITMO GENÉTICO

```
{
  inicializar população (P);
  avaliar população;
  while  Criterio_Nao_Satisfeito
  {
     selecionar pais para reprodução;
     realizar recombinação;
     realizar mutação;
     avaliar população;
     P = sobreviventes;
   }
}
```

PROGRAMAÇÃO EVOLUTIVA

```
{
  inicializar população (P);
  avaliar população;
  while  Criterio_Nao_Satisfeito
  {
     realizar mutação;
     avaliar população;
     P = sobreviventes;
   }
}
```

Estratégias Evolutivas

- Concebidos para solucionar problemas técnicos de otimização.
- Por muito tempo, foi empregado quase que exclusivamente em engenharia civil, como uma alternativa às soluções convencionais. Normalmente não há uma função objetiva fechada para os PTO's e portanto nenhum método aplicável de otimização além da intuição do engenheiro.
- Em uma Estratégia de Evolução dupla (1+1), cada progenitor produz um decendente por geração através da aplicação de mutações normalmente distribuídas, onde pequenos passos ocorrem mais provavelmente que grandes saltos, até que o descendente exibe um desempenho superior ao progenitor e assume o seu lugar.
- Ênfase na auto-adaptação. O papel da recombinação é aceito, mas como operador secundário.

Estratégias Evolutivas

- (μ + 1) - uma população de indivíduos (μ) se recombina de maneira randômica para formar um descendente, o qual, após sofrer mutação, substitui (se for o caso) o pior elemento da população.
- Estratégia Soma: (μ + λ) - os ancestrais e os descendentes convivem.
- Estratégia Vírgula: (μ, λ) - os ancestrais "morrem", deixando apenas os descendentes "vivos".

Estratégias Evolutivas

- A abordagem é adequada a uma vasta gama de problemas de otimização, pois não necessita de muitas informações sobre o problema.
- É capaz de resolver problemas multidimensionais, multimodais e não lineares sujeitos a restrições lineares ou não lineares.

Estratégias Evolutivas

Algoritmo Básico:

- 1. Uma dada população consiste de μ indivíduos. Cada um é caracterizado pelo seu genótipo, consistindo de n genes, que determina de modo não ambíguo a aptidão para a sobrevivência.
- 2. Cada indivíduo da população produz (λ / μ) descendentes, na média, de modo que um total de indivíduos novos são gerados. O genótipo dos descendentes difere ligeiramente dos genótipos de seus ancestrais.
- 3. Apenas os μ melhores indivíduos dos λ gerados permanecem vivos, tornando-se os ancestrais na próxima geração.

Programação Genética

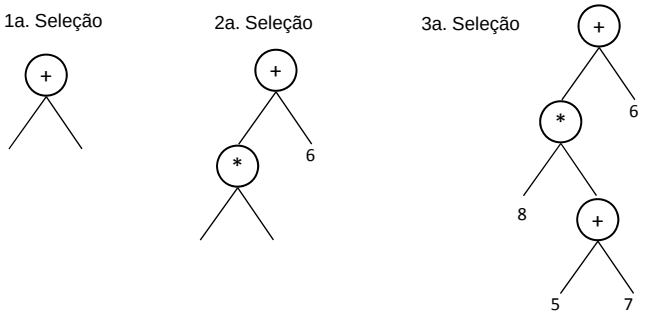
- Koza (1992) sugeriu que um programa de computador bem-sucedido poderia evoluir através de aplicações sucessivas de operadores genéticos.
- Na programação genética, as estruturas que são adaptadas são segmentos de programas organizados hierarquicamente.
- O algoritmo de aprendizagem mantém uma população de programas.
- A aptidão é medida pela capacidade de resolver um conjunto de tarefas e os programas são modificados aplicando recombinação e mutação de subárvores.

Programação Genética

- População inicial: pedaços de programas.
  - Adequados para um domínio de problema podem ser:
    - Operações aritméticas padrão, outras operações de programação e funções matemáticas, bem como funções lógicas e específicas do domínio.
- A produção de novos programas advém da aplicação de operadores genéticos.
- Qualquer programa que se sair bem, sobreviverá para ajudar a produzir os filhos da próxima geração.

Programação Genética

- Necessita-se:
  - **F:** conjunto de Funções → +, *, -, /, sen(x), cos(x) ou operações de matrizes.
  - **T:** conjunto de Valores terminais.
- População de "programas" iniciais:



Programação Genética

Recombinação: