# Algoritmos Genéticos (AG's)

## Sumário







## 1 Introdução

- Os algoritmos genéticos (AG's) foram desenvolvidos por HOLLAND (1975; 1992), da Universidade de Michigan.

- Metas:
  - abstrair e rigorosamente explicar os processos adaptativos em sistemas naturais
  - desenvolver simulações em computador que retenham os mecanismos originais encontrados em sistemas naturais

- Características principais:
  - existência de 2 espaços de trabalho: espaço **genotípico** e espaço **fenotípico**
  - AG's fazem busca sobre uma **população** de pontos e não sobre um único ponto
  - AG's fazem uso de descrições genéricas do que se quer ver presente na solução, através de **funções de *fitness*** (funções-objetivo)
  - AG's utilizam regras de transição **probabilísticas**, e não regras determinísticas

# 2 Algoritmo Genético Clássico

- população de tamanho fixo e estrutura de dados do tipo cadeias binárias
- seleção natural proporcional ao fitness via algoritmo Roulette Wheel
- crossover simples (crossover de um ponto)
- mutação pontual

## 2.1 Representação (Estrutura de Dados)

- Cadeias binárias de comprimento fixo.

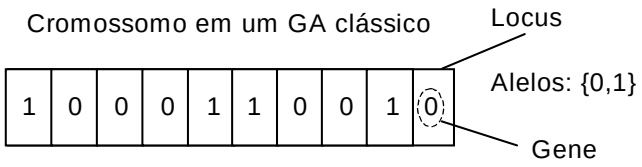



**Figura 1:** Cadeia binária de comprimento $l = 10$ correspondente à estrutura de dados de um GA clássico.



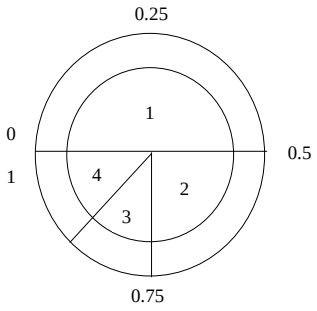


- O problema a ser resolvido é capturado em uma função objetivo que indica o fitness (adaptabilidade) de cada candidato a solução.

## 2.2 Mecanismo de Seleção

- A seleção no GA clássico é proporcional ao fitness e é geralmente implementada utilizando um algoritmo denominado de *Roulette Wheel*.
- **Exemplo**:

| N | Cromossomo | Fitness | Graus |
|---|---|---|---|
| 1 | 0001100101010 | 6.0 | 180 |
| 2 | 0101001010101 | 3.0 | 90 |
| 3 | 1011110100101 | 1.5 | 45 |
| 4 | 1010010101001 | 1.5 | 45 |




- Implementação: gerador de números pseudo-aleatórios com distribuição uniforme.
- Note que este procedimento permite a perda ("morte") do melhor indivíduo e também permite que um indivíduo seja selecionado mais do que uma vez.

- As probabilidades de reprodução de cada indivíduo irão resultar na geração de uma nova população composta por indivíduos probabilisticamente selecionados a partir da população atual.
- Os indivíduos selecionados irão gerar probabilisticamente filhos (descendentes) através de operadores genéticos específicos, particularmente, *crossover* e *mutação*.

## 2.3 Operador de Recombinação (Crossover Simples)

- Nos sistemas biológicos o crossover pode ocorrer durante a reprodução sexuada permitindo a troca de material genético entre dois indivíduos.
- Este processo pode ser abstraído como um operador geral para as estruturas de dados do tipo cadeia binária utilizada no GA clássico (HOLLAND, 1975):
  - Duas cadeias $\boldsymbol{x} = x_1x_2\ldots x_l$ e $\boldsymbol{y} = y_1y_2\ldots y_l$ de comprimento $l$ são selecionadas com probabilidade de crossover $pc$.
  - Um número $r \in \{1,2,\ldots,l-1\}$ indicando o ponto de cruzamento (crossover) é selecionado.





  - Duas novas cadeias são formadas a partir de $\boldsymbol{x}$ e $\boldsymbol{y}$ através da troca de um conjunto de atributos à direita da posição $r$, resultando em $\boldsymbol{x'} = x_1\ldots x_iy_{i+1}\ldots y_l$ e $\boldsymbol{y'} = y_1\ldots y_ix_{i+1}\ldots x_l$.
- Os dois novos cromosomos gerados $\boldsymbol{x'}$ e $\boldsymbol{y'}$ são os filhos (*offspring*) de $\boldsymbol{x}$ e $\boldsymbol{y}$.

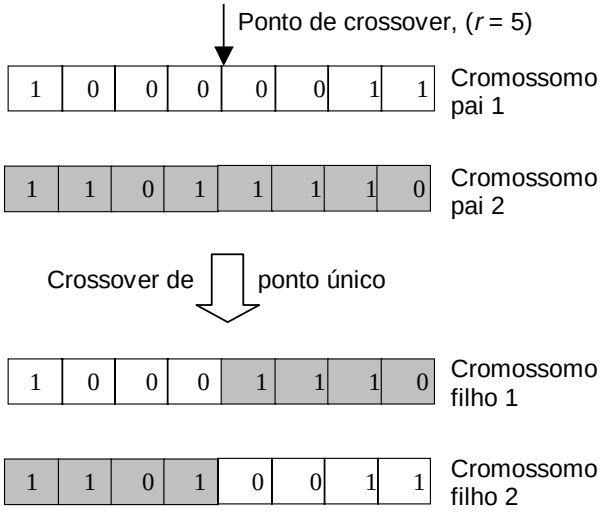


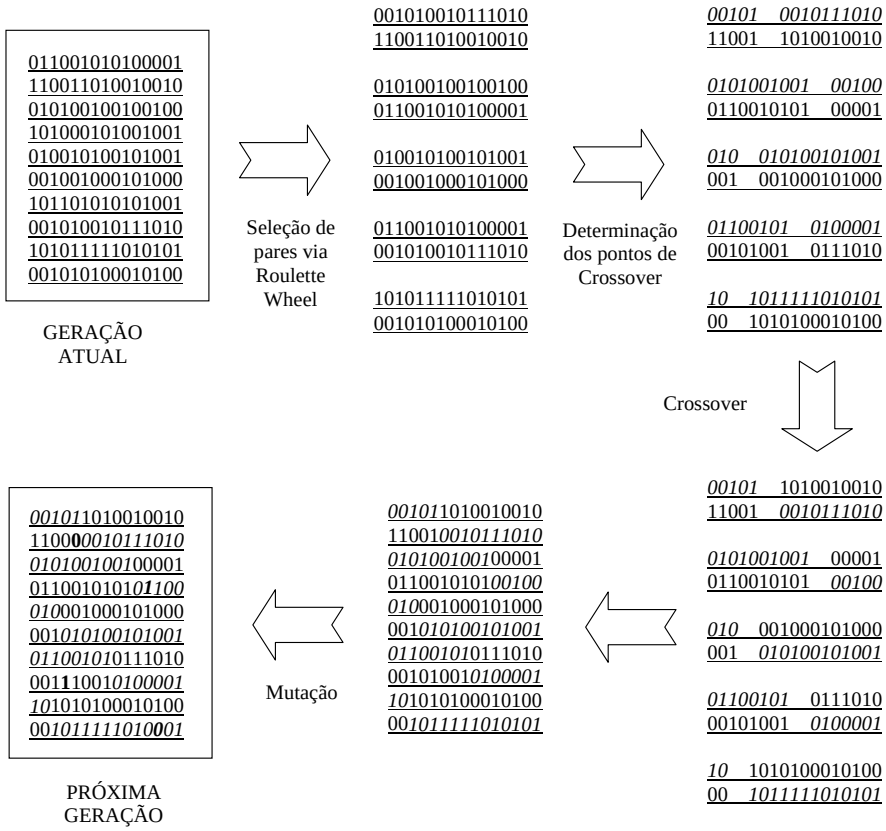

**Figura 2:** Crossover de um único ponto para cadeias de comprimento $l = 8$.

## 2.4 Operador de Mutação

- Em genética a mutação pontual é um processo no qual um alelo de um gene é aleatoriamente substituído (ou modificado) por outro, resultando em um novo cromossomo.
- Geralmente existe uma baixa probabilidade de mutar cada gene de um cromossomo.
- Isso significa que cada bit na população *P* é operado da seguinte forma:
  - Os números *r,…,u* indicando as posições que irão sofrer mutação são determinadas aleatoriamente de forma que cada posição possui uma pequena probabilidade *pm* de sofrer mutação independente das outras posições.
  - Uma nova cadeia $\boldsymbol{x'} = x_1 \ldots x_r \ldots x_u \ldots x_l$ é gerada onde $x_r \ldots x_u$ são determinadas aleatoriamente partindo do conjunto de alelos para cada gene. No caso de cadeias binárias, se uma posição possui alelo '0', então ela se torna '1' e vice-versa.







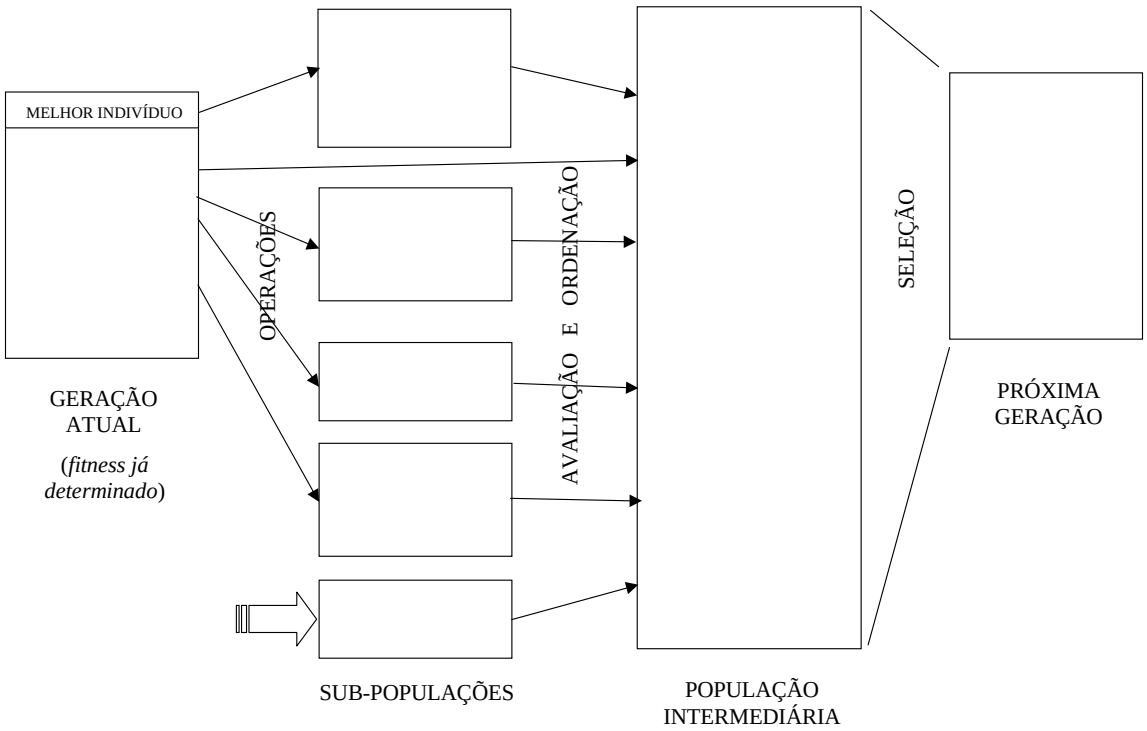
- **Problemas com o algoritmo clássico:**
  - política de reprodução/seleção permite a perda do melhor indivíduo
  - posição dos genes no cromossomo influi na probabilidade de permanecerem no mesmo cromossomo após crossover
  - dificuldades na codificação binária de números reais
- **Algumas estratégias de solução:**
  - mecanismos alternativos de seleção
  - crossover uniforme
  - codificação em arranjos de números reais

## 3 Algoritmo Genético Modificado

- geração de sub-populações por meio da aplicação de operadores genéticos e outros operadores sobre membros da geração atual
- avaliação (cálculo do *fitness*) e ordenação da população intermediária
- seleção para nova geração







# 4 Exercícios computacionais

**EXERCÍCIO COMPUTACIONAL 3 (EC3)**: Dada a função de uma única variável $g(x) = 2^{-2((x-0.1)/0.9)^2}\text{sen}(5\pi x)^6$, apresentada na Figura 1, tente determinar seu ótimo global utilizando o algoritmo genético padrão. A variável *x* está definida no intervalo [0,1], $x \in [0,1]$. (Exercício 2, Seção 8.2, pg. 37, do Tutorial 1.)

Escolha como representação para a variável *x* uma cadeia binária *m* ($m = \langle m_l,..., m_2, m_1\rangle$) de comprimento $l = 22$, correspondente a uma precisão de seis casas decimais.

- O mapeamento de uma cadeia binária para um número real pode ser feito em dois passos:
  - Converta a cadeia binária $m = \langle m_l,..., m_2, m_1\rangle$ da base 2 para a base 10:

$$\left(\langle m_l,...,m_2,m_1\rangle\right)_2 = \left(\sum_{i=0}^{l-1} m_i.2^i\right)_{10} = x'$$





  - Encontre o correspondente valor real ($\Re$) para *x*: $x = x_{\min} + x'.\frac{x_{\max} - x_{\min}}{2^l - 1}$, onde $x_{\min} = 0$ e $x_{\max} = 1$, $x \in [x_{\min}, x_{\max}]$ (domínio da variável).

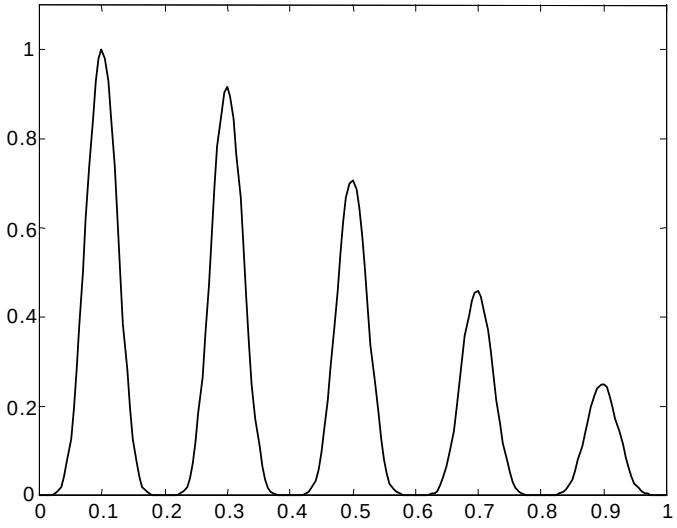



**Figura 3:** Função $g(x) = 2^{-2((x-0.1)/0.9)^2}\sin(5\pi x)^6$ a ser maximizada.

**Função de Avaliação**: eval($v$) = $f(x)$

**Operadores Genéticos**: Crossover simples, mutação pontual

**População Inicial** - Aleatória com 50 indivíduos

**Parâmetros:**

- probabilidade de crossover $p_c$ = 0,5 e probabilidade de mutação $p_m$ = 0,01

**EXERCÍCIO COMPUTACIONAL 4 (EC4)**: Resolver o Exercício 3 da Seção 8.4 (pg. 37) do Tutorial 1.

O problema do caixeiro viajante (*travelling salesman problem – TSP*) é de natureza combinatorial e aparece em diversas aplicações, do projeto de circuitos integrados à entrega de *fast food*. Em sua forma mais simples, o caixeiro deve visitar cada cidade em um dado território somente uma vez, e depois retornar a cidade de origem. Dado o custo da viagem entre cada uma das cidades, qual o itinerário que resulta no custo mínimo total?





Implemente um algoritmo genético para resolver o problema do caixeiro ilustrado na Figura 4.

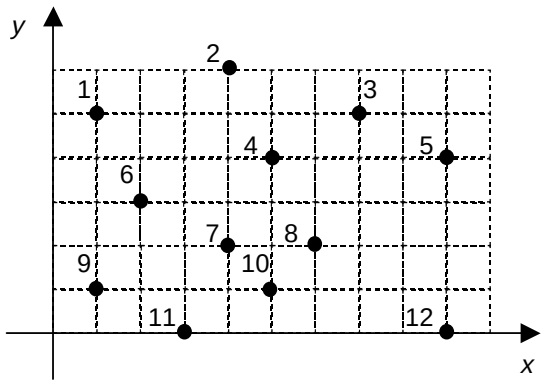


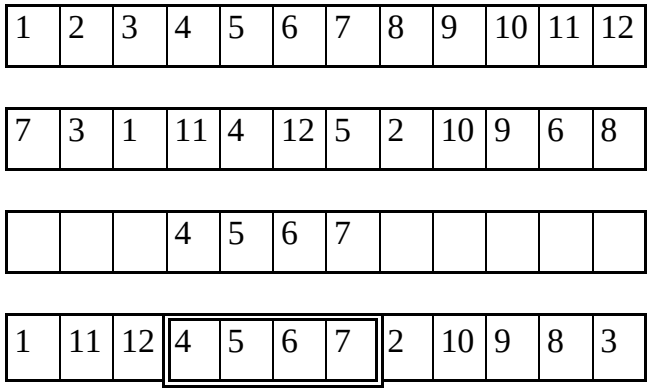

**Figura 4:** TSP simples com 12 cidades.

**Codificação** - arranjo de números inteiros: ⟨1,3,4,5,6,8,9,12,11,10,7,2⟩

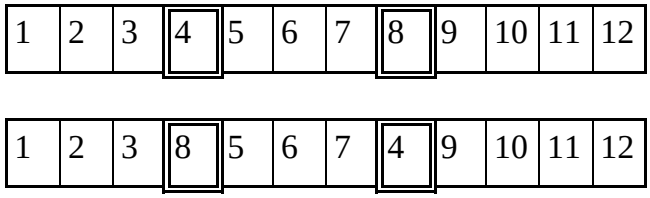
**Função de *fitness*** - soma dos custos de movimentação entre cada par de cidades.

## Operadores

- Crossover OX:



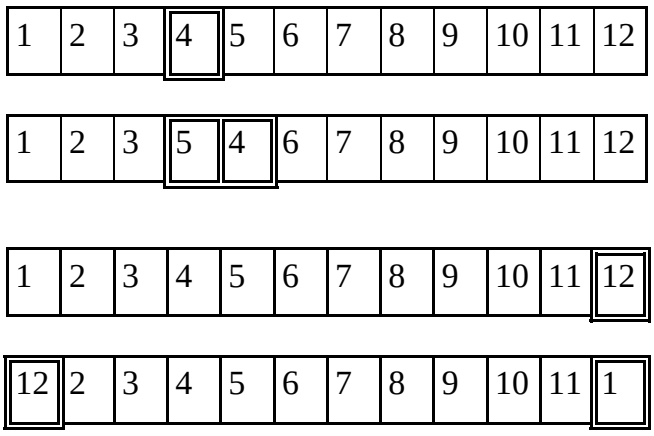

- Mutação Inversiva








- Variação de Mutação Inversiva




**EXERCÍCIO COMPUTACIONAL 5 (EC5)**: Verifique o comportamento do algoritmo implementado no exercício 4 quando as seguintes modificações são inseridas independentemente e em conjunto:

- Elitismo na seleção: mantenha sempre o melhor indivíduo na população.
- Crossover de todos os indivíduos com o melhor indivíduo.

## 5 Teoria dos Esquemas (*schemata theory*)

- A teoria dos esquemas foi proposta por HOLLAND (1975; 1992) para tentar explicar por que os algoritmos genéticos funcionam.
- Nesta seção, apresentaremos os principais resultados da teoria dos esquemas: o teorema de crescimento dos esquemas e a hipótese dos blocos construtivos.
- Um *esquema* é uma representação capaz de descrever diversos cromossomos simultaneamente. Um esquema é construído inserindo um caractere *don't care* ($*$) no alfabeto dos genes, indicando que aquele gene representa qualquer alelo.
  - Por exemplo, o esquema [1 $*$ 0 1 0 0 1] representa os cromossomos [1 **0** 0 1 0 0 1] e [1 **1** 0 1 0 0 1]. O esquema [1 $*$ 0 $*$ 1 1 0] representa quatro cromossomos: [1 **0** 0 **0** 1 1 0], [1 **0** 0 **1** 1 1 0], [1 **1** 0 **0** 1 1 0] e [1 **1** 0 **1** 1 1 0].
- Obviamente, o esquema [1 1 1 0 0 1 0] representa apenas um cromossomo, enquanto que o esquema [$*$ $*$ $*$ $*$ $*$ $*$ $*$] representa todos os cromossomos de comprimento 7.





- Observe que cada esquema representa $2^r$ cromossomos, onde $r$ é o número de caracteres *don't care* "$*$" presentes no esquema. Por outro lado, cada cromossomo de comprimento $m$ é representado por $2^m$ esquemas.
  - Por exemplo, considere o cromossomo [0 1 0 0 1 0 0]. Este cromossomo é representado pelos seguintes $2^7$ esquemas:

[0 1 0 0 1 0 0]
[* 1 0 0 1 0 0]
[0 * 0 0 1 0 0]
⋮
[0 1 0 0 1 0 *]
[* * 0 0 1 0 0]
[* 1 * 0 1 0 0]
⋮
[0 1 0 0 1 * *]
[* * * 0 1 0 0]
⋮
[* * * * * * *]

- Considerando cromossomos de comprimento *m*, há um total de $3^m$ possíveis esquemas. Numa população de tamanho *n*, entre $2^m$ e $n \cdot 2^m$ diferentes esquemas podem ser representados.
- A *ordem* de um esquema *S*, $o(S)$, é definida como o número de 0's e 1's presentes no esquema, isto é, o número de *posições fixas* (caracteres diferentes de *don't care*) presentes no esquema. A ordem de um esquema define sua especificidade, de modo que quanto maior a ordem, mais específico é o esquema.
- O *comprimento definitório* de um esquema *S*, denotado por $\delta(S)$, é a maior distância entre posições fixas de um cromossomo. O comprimento definitório define o nível de compactação da informação contida no esquema.
- O *fitness* de um esquema *S* na geração *t*, eval(*S*, *t*), é definido como a média dos *fitness* de todos os cromossomos na população representados pelo esquema *S*. Assuma que há *p* cromossomos $\left\{\mathbf{x}_{i_1}^t, \ldots, \mathbf{x}_{i_p}^t\right\}$ representados pelo esquema $S_i$ na geração *t*. Então:





$$\text{eval}(S_i, t) = \frac{1}{p} \sum_{j=1}^{p} \text{eval}\left(\mathbf{x}_{i_j}^t\right),$$

onde $\text{eval}\left(\mathbf{x}_{i_j}^t\right)$ é o *fitness* do indivíduo $\mathbf{x}_{i_j}^t$.

- Seja *tam_pop* o tamanho da população. O *fitness* médio da população na geração *t*, $\overline{F}(t)$, é dado por

$$\overline{F}(t) = \frac{1}{tam\_pop} \sum_{i=1}^{tam\_pop} \text{eval}\left(\mathbf{x}_i^t\right).$$

- Sejam $p_c$ e $p_m$ as probabilidades de *crossover* e mutação, respectivamente, e *m* o comprimento dos cromossomos. Seja $\xi(S_i, t)$ o número de cromossomos representados pelo esquema $S_i$ na geração *t*. Pode-se mostrar que (MICHALEWICZ, 1996):

$$\xi(S_i, t+1) \geq \frac{\xi(S_i, t)\text{eval}(S_i, t)}{\overline{F}(t)}\left[1 - p_c \frac{\delta(S_i)}{m-1} - o(S_i)p_m\right]$$

- A equação acima é conhecida como *equação de crescimento reprodutivo do esquema*. Esta equação é deduzida supondo que a função de *fitness* $f(\cdot)$ produz apenas valores positivos. Se a função a ser otimizada produz valores negativos, um mapeamento entre as funções de otimização e de *fitness* é necessário.
- Esta equação de crescimento mostra que a seleção aumenta a amostragem de esquemas cujo *fitness* está acima da média da população, e este aumento é exponencial (MICHALEWICZ, 1996).
- A seleção, por si só, não introduz nenhum novo esquema (não representado na geração inicial em $t = 0$). Esta é a razão da introdução do operador de *crossover*: possibilitar a troca de informação estruturada, ainda que aleatória. Além disso, o operador de mutação introduz uma variabilidade maior na população.
- O efeito (destrutivo) combinado destes operadores não é significativo se o esquema é curto e de ordem baixa. O resultado final da equação de crescimento pode ser formulado como segue:





**Teorema dos Esquemas:** Esquemas com comprimento definitório curto, de ordem baixa, e com *fitness* acima da média, têm um aumento exponencial de sua participação em gerações consecutivas de um algoritmo genético.
Prova: Veja HOLLAND (1975; 1992).

- Uma conseqüência imediata deste teorema é que os algoritmos genéticos tendem a explorar o espaço por meio de esquemas curtos e de baixa ordem que, subseqüentemente, são usados para troca de informação durante o *crossover*.

**Hipótese dos Blocos Construtivos**: Um algoritmo genético busca desempenho quase-ótimo através da justaposição de esquemas curtos, de baixa ordem e alto desempenho, chamados de *blocos construtivos*.

- Em uma população de tamanho *tam_pop*, indivíduos de comprimento *m* processam pelo menos $2^m$ e no máximo $2^{tam\_pop}$ esquemas. Alguns deles são processados de forma útil: são amostrados a uma taxa crescente exponencial (desejável); e outros são quebrados por meio de *crossover* e mutação.

- HOLLAND (1975; 1992) mostrou que, em uma população de tamanho *tam_pop*, pelo menos $tam\_pop^3$ são processados de forma útil. Esta propriedade foi denominada *paralelismo implícito*, pois é obtida sem nenhuma exigência extra de memória e processamento. Entretanto, BERTONI & DORIGO (1993) mostraram que a estimativa $tam\_pop^3$ é válida apenas para o caso particular em que *tam_pop* é proporcional a $2^l$, onde $l = \frac{1}{2} m\varepsilon$ e $\varepsilon$ é a probabilidade de um esquema ser rompido por *crossover*.
- Note entretanto que, em alguns problemas, alguns blocos construtivos (esquemas curtos, de ordem baixa) podem direcionar erroneamente o algoritmo, levando-o a convergir a pontos sub-ótimos. Este fenômeno é conhecido como **decepção**. Assim, a hipótese dos blocos construtivos não fornece uma explicação definitiva do porquê os algoritmos genéticos funcionam. Ela é apenas uma indicação do porquê os algoritmos genéticos funcionam para uma certa classe de problemas.





## 5.1 Deception Problem

- Alguns blocos construtivos podem direcionar erroneamante o GA, levando a uma convergência a pontos sub-ótimos:
  - ⟨1 1 1 * * * * * * * *⟩ – fitness *acima* da média
  - ⟨* * * * * * * * * 1 1⟩ – fitness *acima* da média
  - ⟨1 1 1 * * * * * * 1 1⟩ – fitness *muito menor* que ⟨0 0 0 * * * * * * 0 0⟩
  - solução ótima – ⟨1 1 1 1 1 1 1 1 1 1 1⟩
  - tendência a convergir para pontos como ⟨0 0 0 1 1 1 1 1 1 0 0⟩

- Algumas alternativas foram propostas para combater o problema da decepção (GOLDBERG, 1989; MICHALEWICZ, 1996). A primeira assume que há algum conhecimento a priori da função-objetivo para que seja possível codificá-la de forma apropriada (que forme blocos construtivos "coesos").

- A segunda é a utilização de um operador de *inversão*: selecionam-se 2 pontos em um cromossomo e inverte-se a ordem dos bits entre os pontos selecionados (alterando-se a codificação em conjunto com a operação).
- A terceira opção é utilizar algoritmos genéticos *messy*, que diferem do algoritmo genético clássico de várias maneiras: codificação, operadores, presença de cromossomos de tamanho distinto e fases evolutivas.

# 6 Referências

GOLDBERG, D. E. "Messy Genetic Algorithms: Motivation, Analysis, and First Results", *Complex Systems*, 3: 493-530, 1989.

HOLLAND, J.H. "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975.

HOLLAND, J.H. "Adaptation in Natural and Artificial Systems", 2nd edition, The MIT Press, 1992.

MICHALEWICZ, Z. "Genetic Algorithms + Data Structures = Evolution Programs", 3rd edition, Springer, 1996.