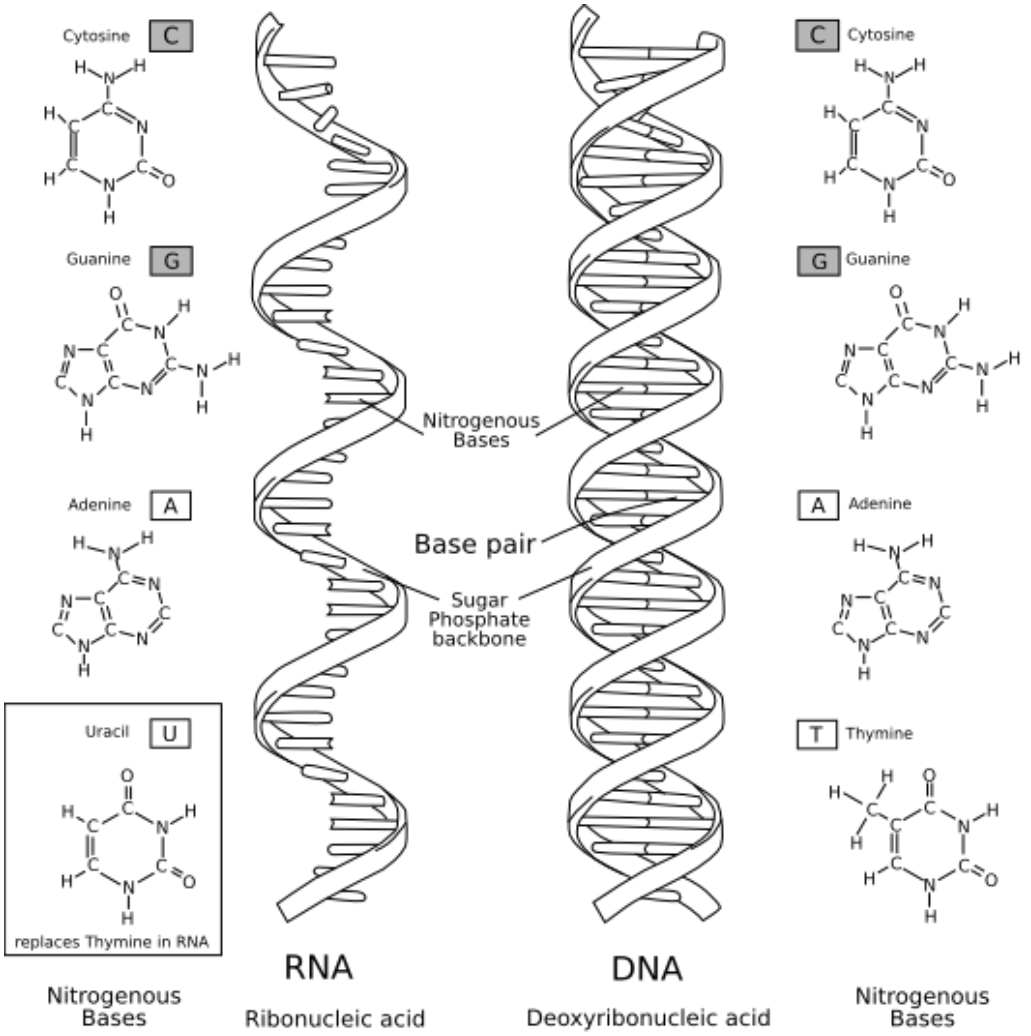
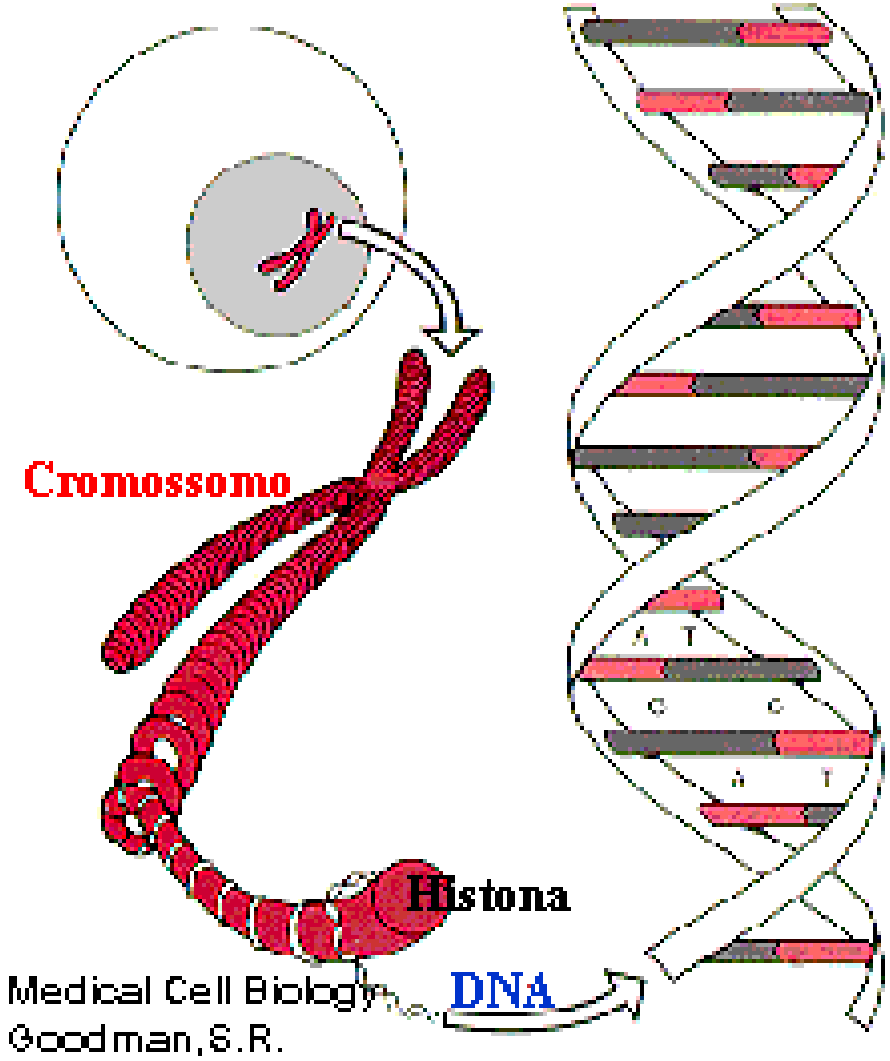
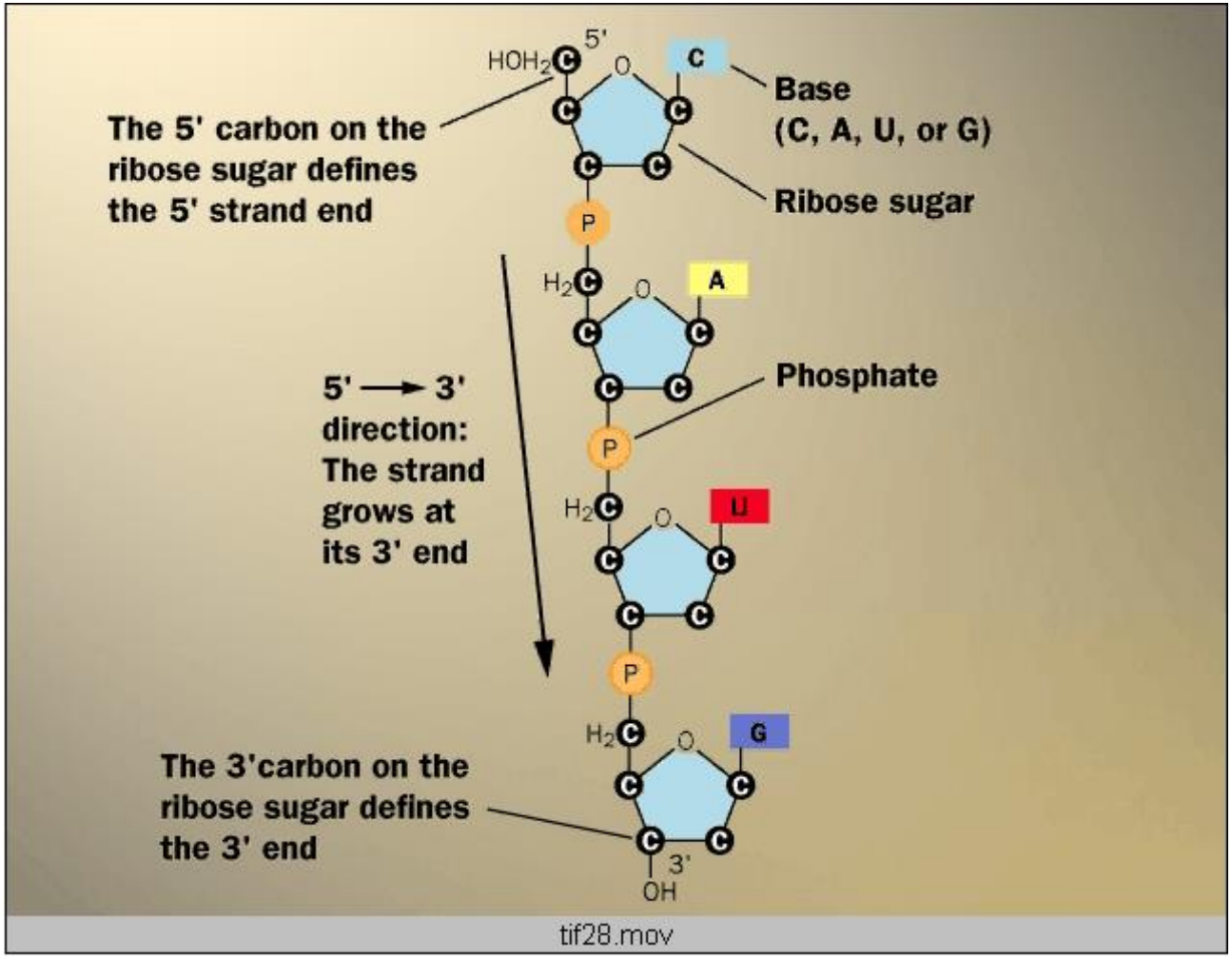
Computação Evolutiva

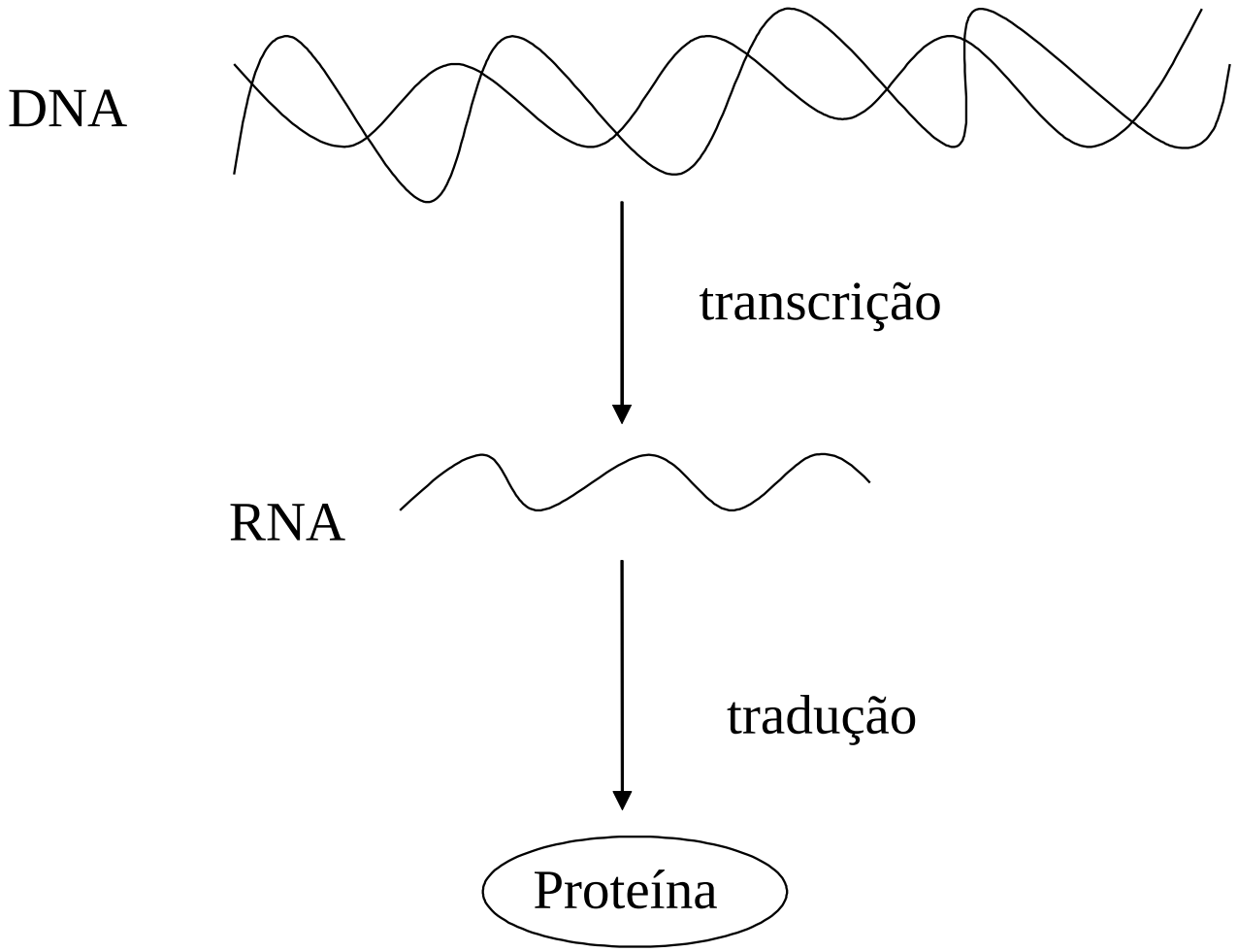
Índice

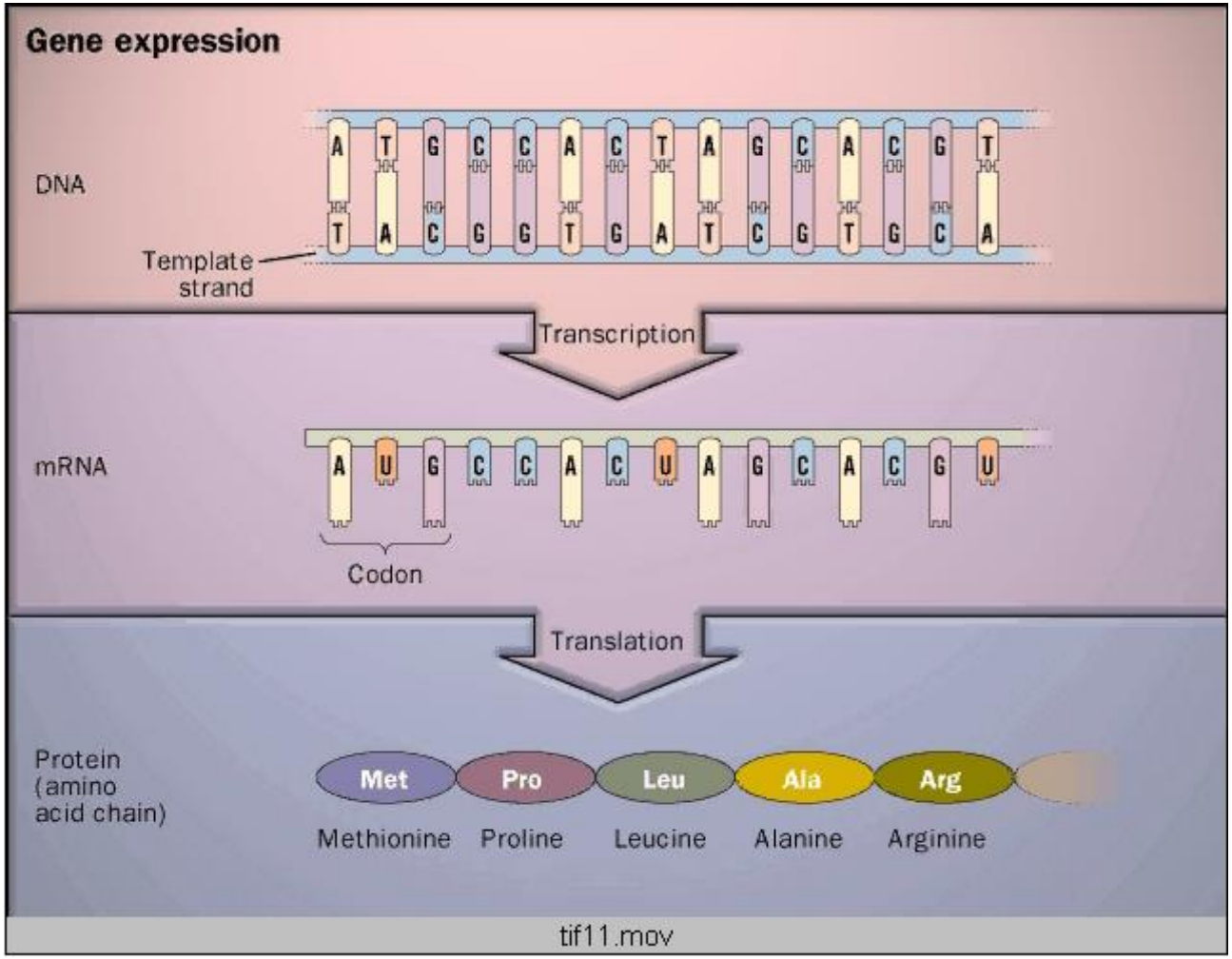
1. Transcrição e tradução

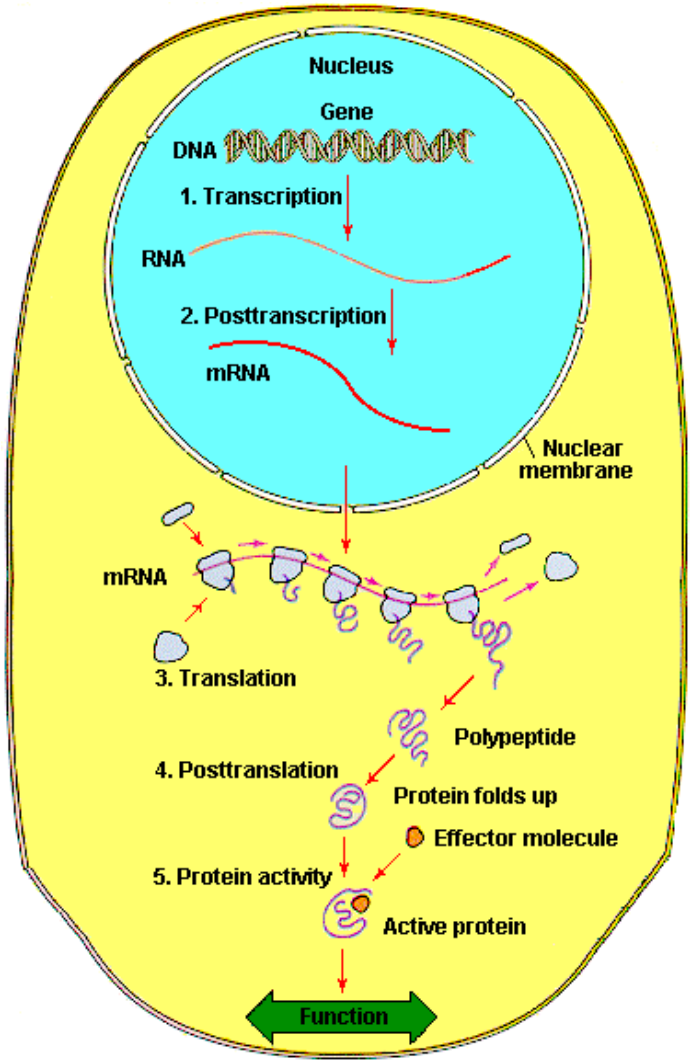



Cromossomo
Histona
DNA
Medical Cell Biology
Goodman, S.R.


The 5' carbon on the ribose sugar defines the 5' strand end
Base (C, A, U, or G)
Ribose sugar
Phosphate
5' → 3' direction: The strand grows at its 3' end
The 3'carbon on the ribose sugar defines the 3' end
5'
3'
OH
C
A
U
G
P
O
tif28.mov


DNA
transcrição
RNA
tradução
Proteína


Gene expression
DNA
Template strand
Transcription
mRNA
Codon
Translation
Protein (amino acid chain)
Met
Pro
Leu
Ala
Arg
Methionine
Proline
Leucine
Alanine
Arginine
tif11.mov



Procarioto:

RNA – proteína

Eucarioto:

pre-mRNA – mRNA maduro – proteína

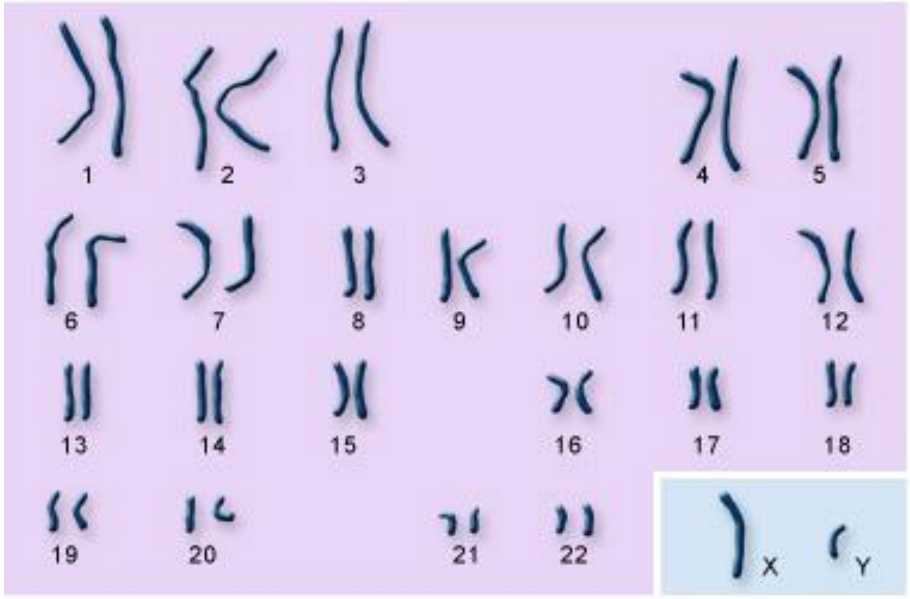
1.1 Aspectos numéricos e curiosidades acerca do genoma

- Procariotos (10^6 a 10^7 pares de bases) × Eucariotos (10^7 a 10^{10} pares de bases)
- O sequenciamento do genoma humano foi concluído em abril/2003 e chegou-se a mais de 3 bilhões de pares de bases. Estima-se que existam entre 20 e 27 mil genes.
- O genoma da levedura Saccharomyces cerevisiae possui 12,4 milhões de pares de bases, ou seja, 0,4% do tamanho do genoma humano, e aprox. 6,3 mil genes.
- O genoma da mosca da fruta Drosophila melanogaster possui aproximadamente 165 milhões de bases e 13,6 mil genes.
- Na levedura, as proteínas têm em média 466 aminoácidos.
- A maior proteína tem 27.000 aminoácidos e é responsável pela elasticidade passiva dos músculos em seres humanos.
- Apenas entre 1 e 1,5% do genoma humano codifica proteína (éxons). O restante corresponde a, por exemplo, sequências regulatórias, introns (25%) e DNA não-codificante.

- Se esticarmos o material genético, cada cromossomo humano terá entre 1,7 e 8,5 cm, com média de 5,0 cm. Logo, o genoma de uma célula humana, quando esticado, atinge aproximadamente 1,15 metros. Como temos aproximadamente 10 trilhões de células em nosso organismo, esticando e empilhando o código genético de todas as células de um ser humano produziria o comprimento de 1,2 bilhões de quilômetros, que corresponde a 4 viagens completas de ida e volta da Terra ao Sol.
- A parte terminal de um cromossomo (para os casos em que os cromossomos não são circulares, como é o caso humano) é chamada de região telomérica (ou telômero). Cada vez que uma célula se reproduz, as células descendentes vão possuir uma região telomérica menor. Já que com uma região telomérica muito curta ou já inexistente a célula perde funcionalidades essenciais, naturalmente as células possuem um número finito de "gerações". Para aqueles que têm como objetivo viver mais tempo e com qualidade, sugere-se evitar a promoção da reprodução de suas células, particularmente aquelas reproduções que podem ser controladas pelo seu estilo de vida.

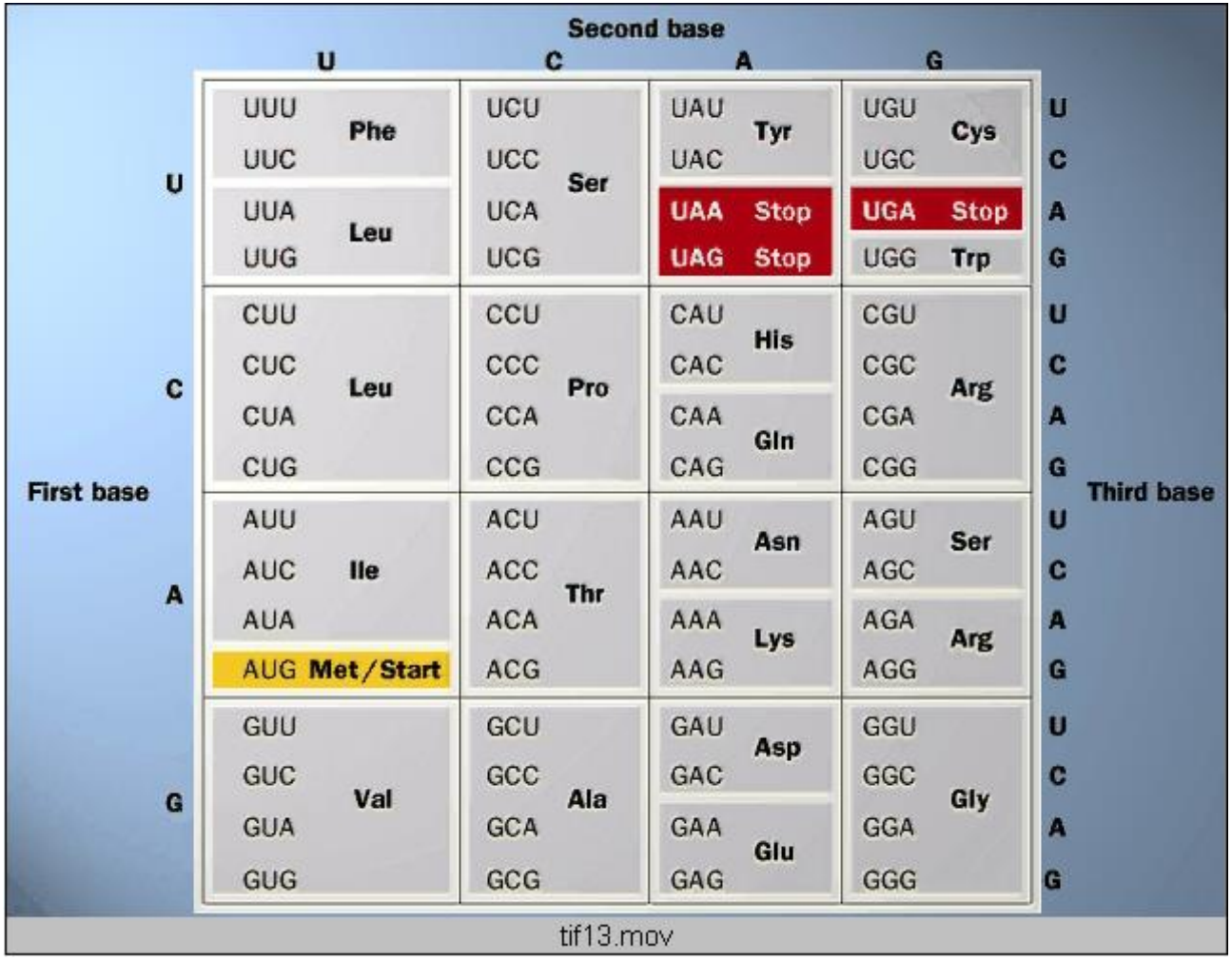
How many genes do other organisms have?

	chromosomes –diploid	base pairs	genome size (#genes)
fruit fly	8	1.65×10^{8}	13,600
Budding yeast	16	12,462,637	6,275
human	46	3.3×10^{9}	~21,000
human mitochondria		16,569	13
rice	24	4.66×10^{8}	46,022 -55,615
dog	78	2.4×10^{9}	~25,000
mouse	40	3.4×10^{9}	~23,000

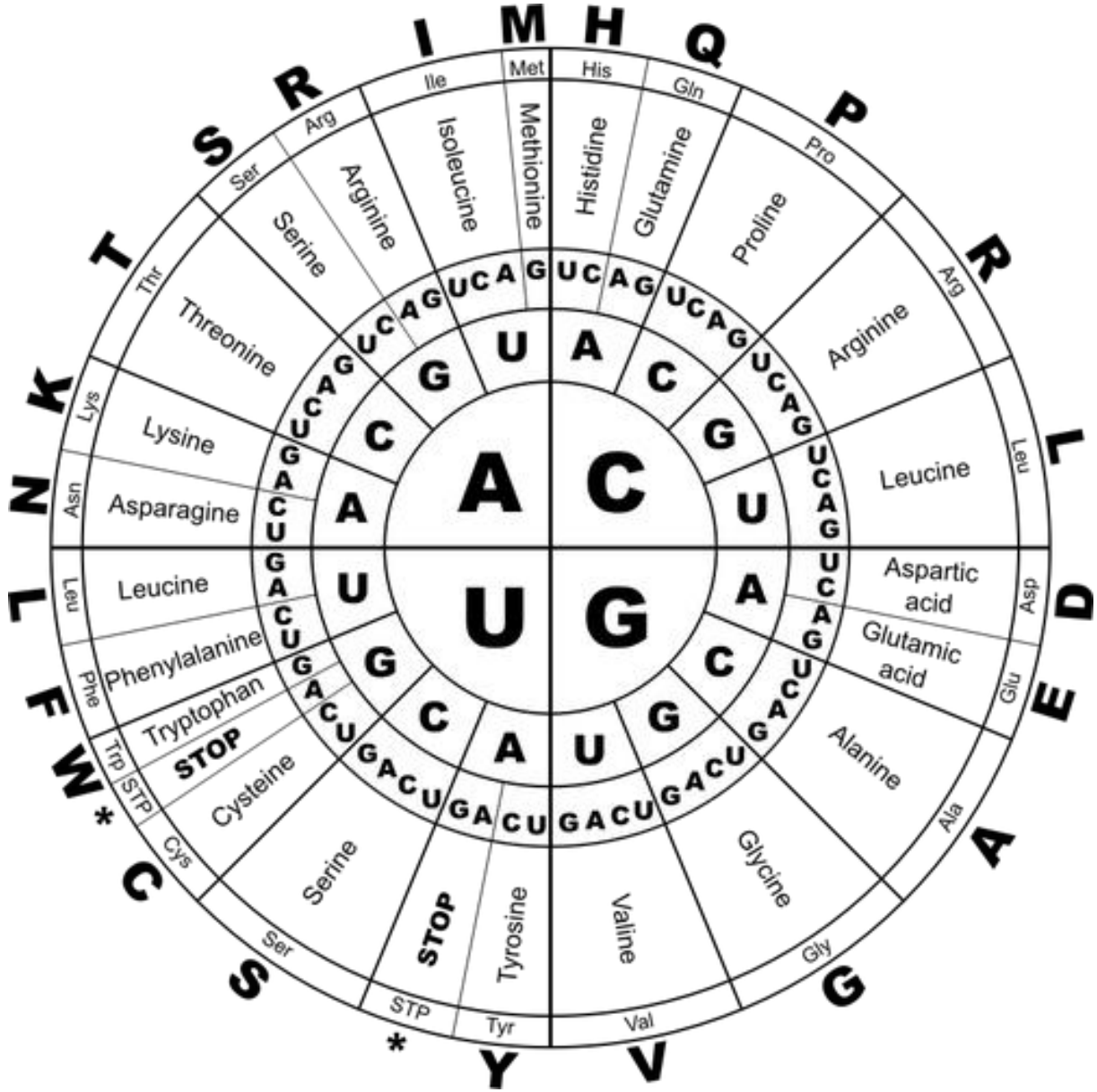


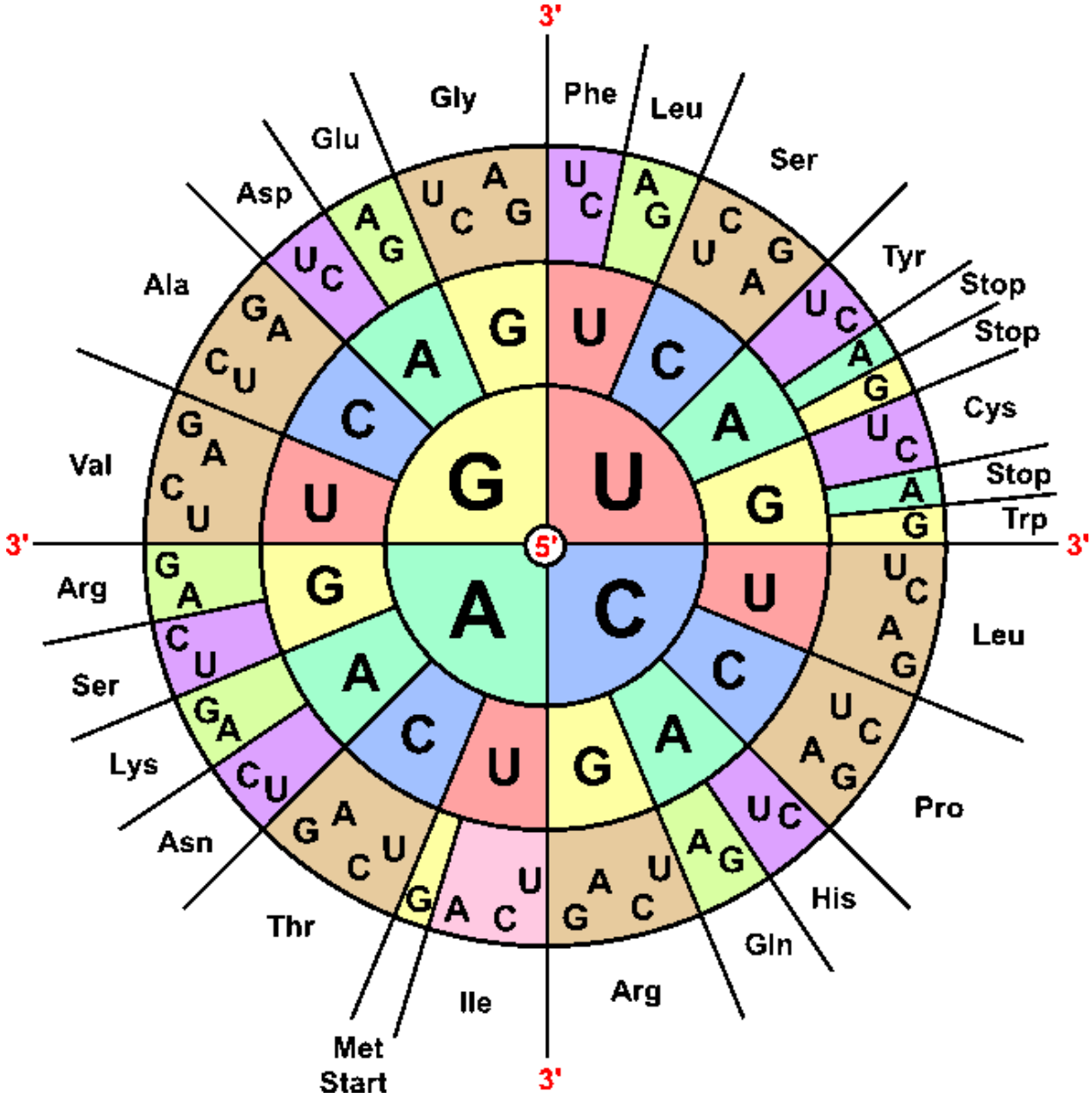
23 pares de cromossomos humanos

1.2 Código genético



4^3 códons $\rightarrow$ 20 aminoácidos

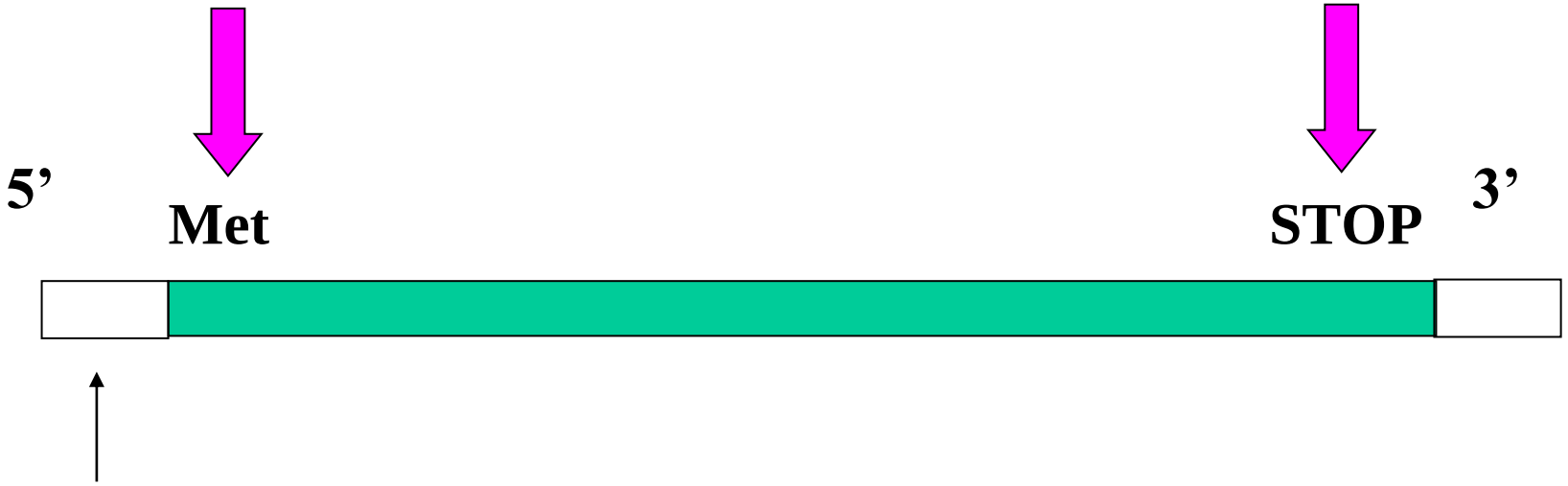

A
C
U
G
Methionine
Isoleucine
Arginine
Serine
Histidine
Glutamine
Proline
Arginine
Leucine
Threonine
Lysine
Asparagine
Leucine
Phenylalanine
Tryptophan
STOP
Cysteine
Aspartic acid
Glutamic acid
Alanine
Glycine
Valine
Tyrosine
STOP
Serine
Met
Ile
Arg
Ser
His
Gln
Pro
Arg
Leu
Thr
Lys
Asn
Leu
Phe
Trp
STP
Cys
Asp
Glu
Ala
Gly
Val
Tyr
STP
Ser


3'
Gly
Phe
Leu
Glu
Ser
Asp
Tyr
Ala
Stop
Stop
Cys
Val
Stop
Trp
3'
5'
3'
Arg
Leu
Ser
Lys
Pro
Asn
His
Thr
Gln
Ile
Arg
Met
Start
3'

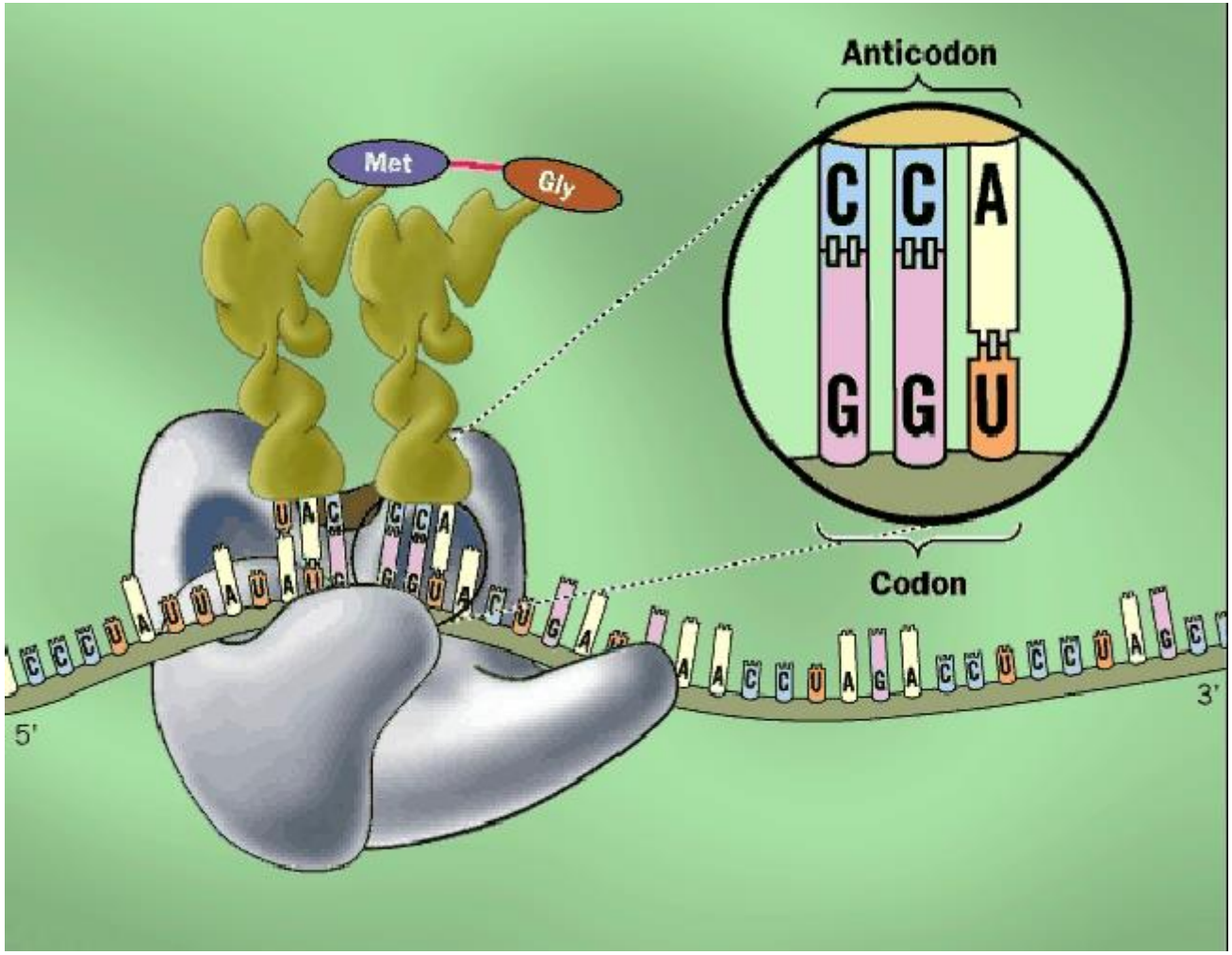
CÓDON PREFERENCIAL

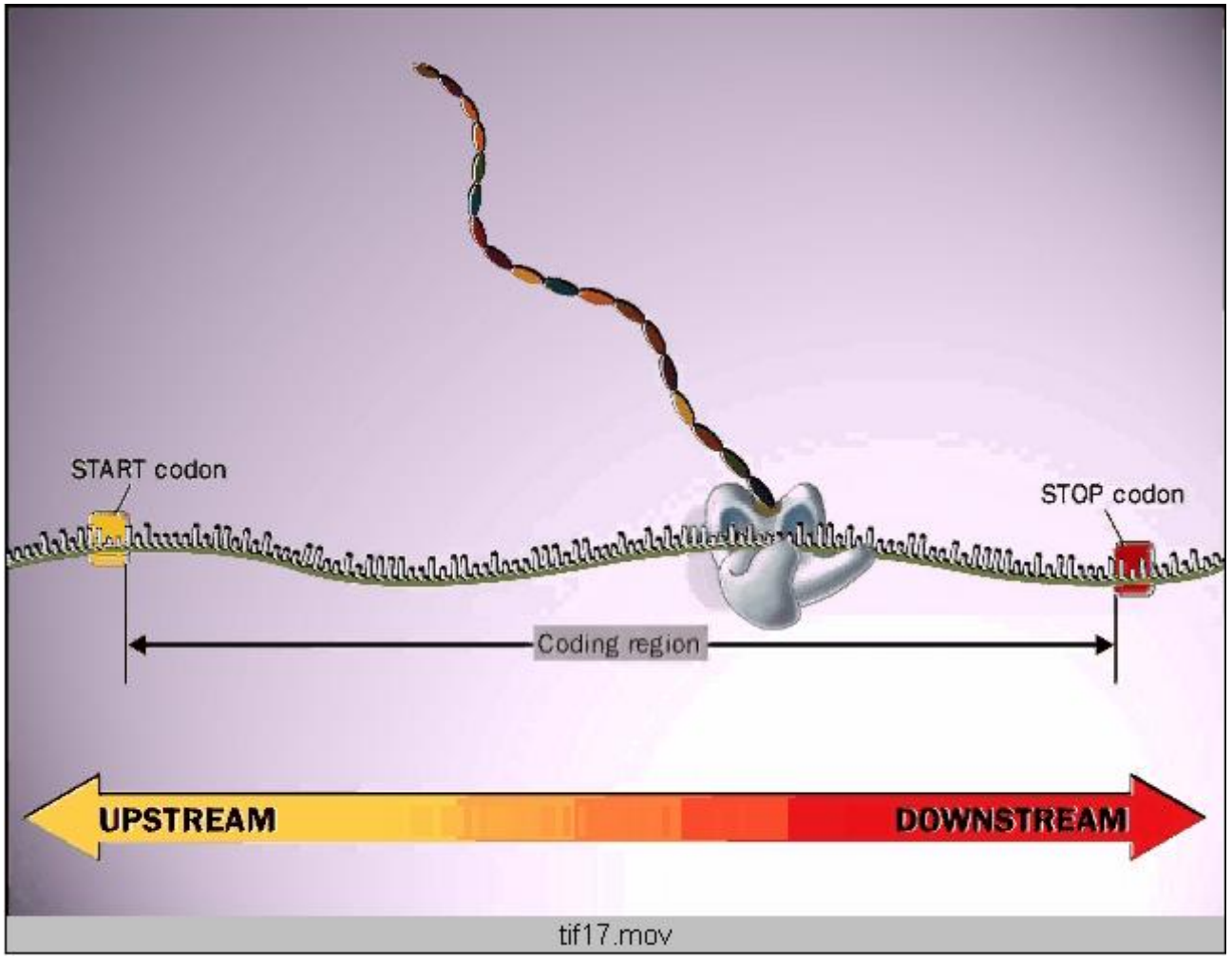
Ex: CUU (65%)
CUC (15%)
CUA (10%) Leucina
CUG (5%)
UUA (3%)
UUG (2%)

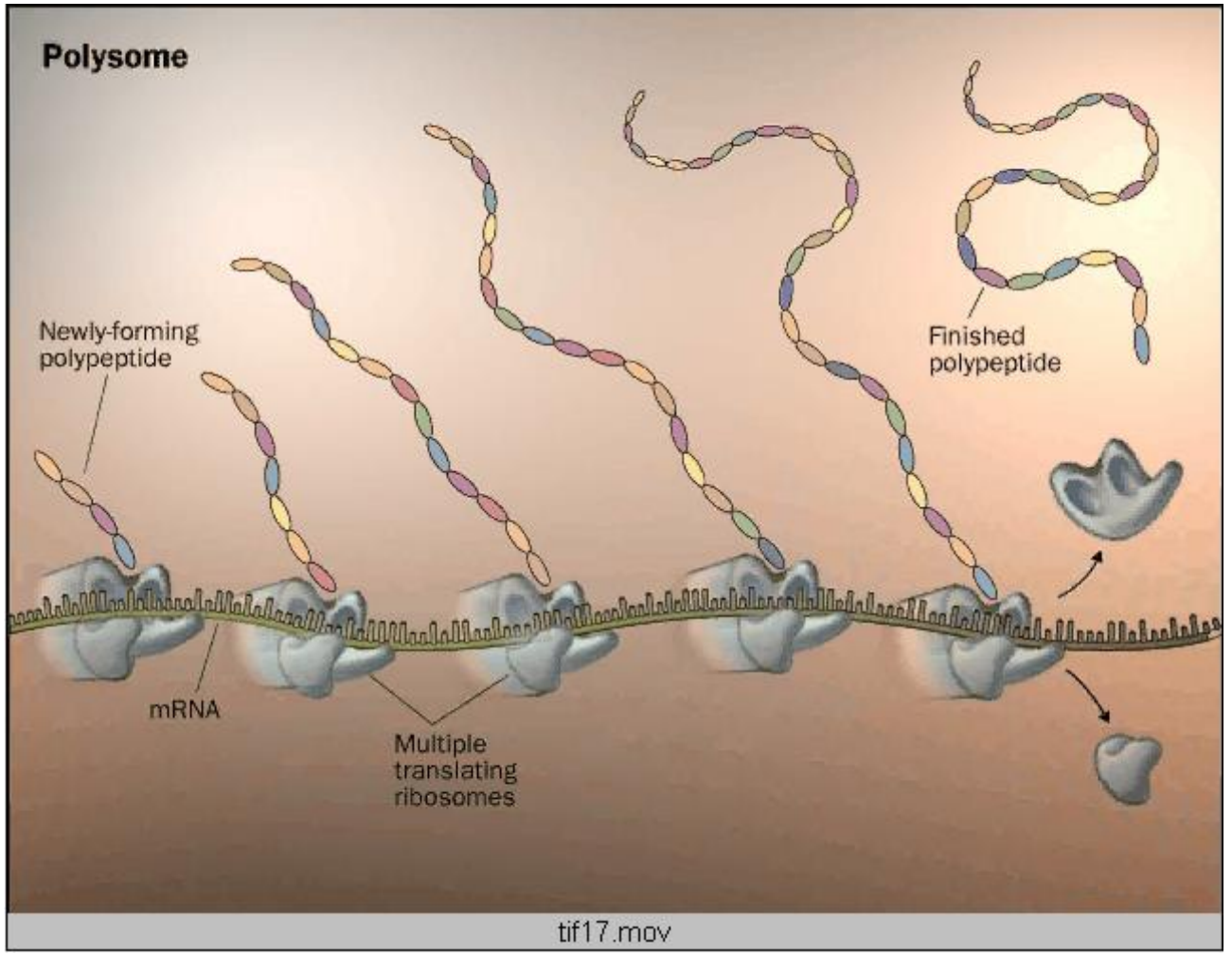
1.3 Estrutura de um gene



1.4 Maquinaria de tradução

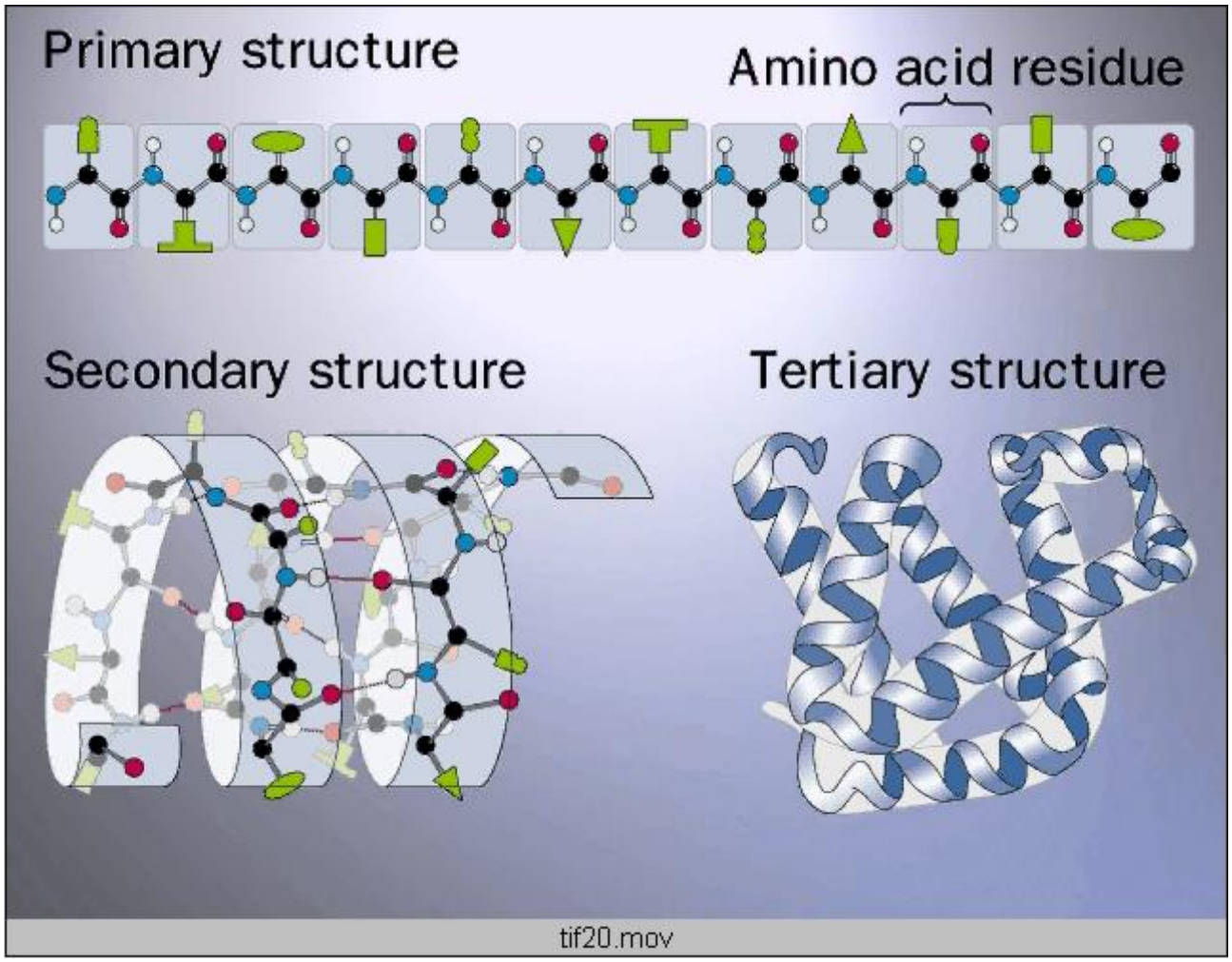


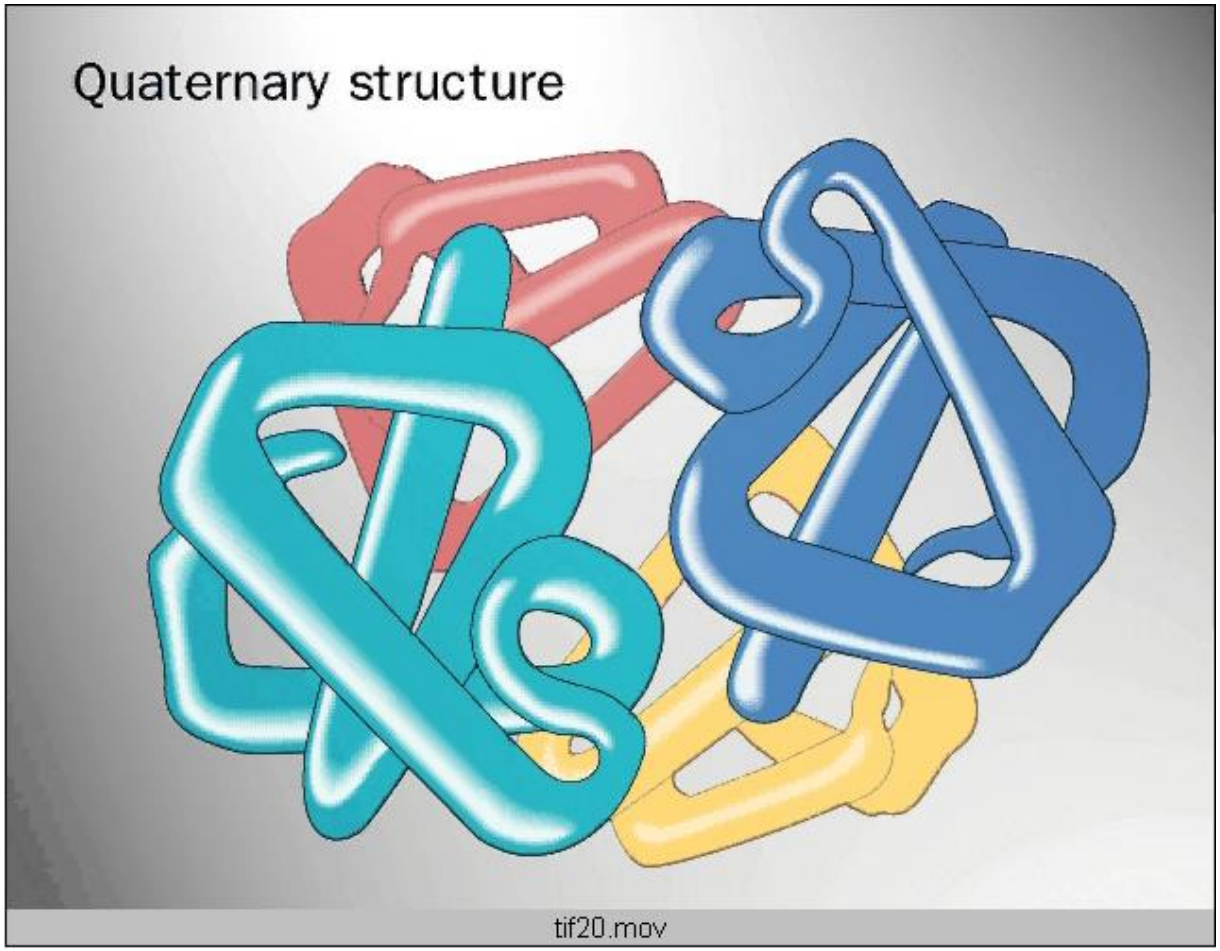

START codon
STOP codon
Coding region
UPSTREAM
DOWNSTREAM
tif17.mov



1 mRNA $\rightarrow$ + de 10 moléculas de proteínas

1.5 Diferentes conformações das proteínas





Para um vídeo didático abordando transcrição e tradução, tem-se:

https://www.youtube.com/watch?v=gG7uCskUOrA

2. Outros exemplos de inspiração na natureza

https://biomimicry.org/biomimicry-examples/

Martim-pescador, seu bico em forma de cunha e a continuação de sua cabeça tornam eficientes seus mergulhos na água (envolve passagem abrupta de um ambiente menos denso para um mais denso).

Trem-bala da linha entre Osaka e Hakata (Japão), a qual é caracterizada pela presença de muitos túneis (envolve passagem abrupta de um ambiente menos denso para um mais denso).

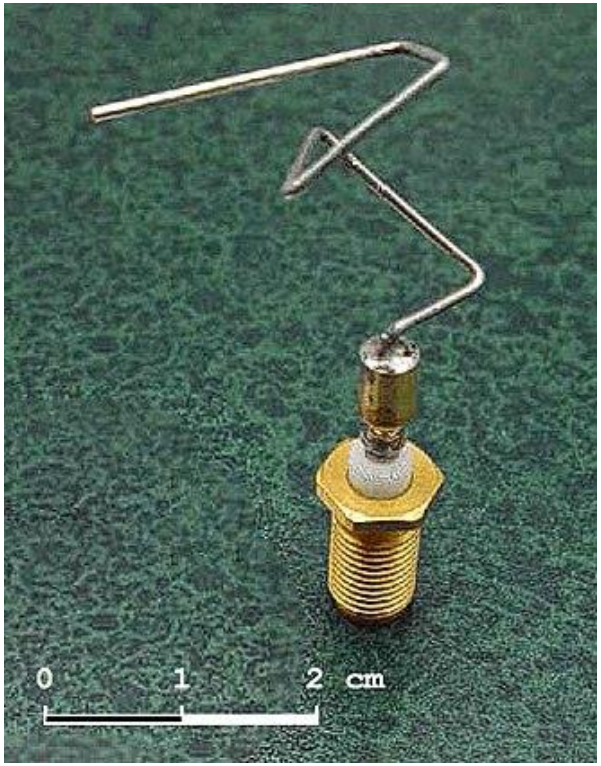
Penetração eficiente em ambientes mais densos com um bico em forma de cunha

Controle climático do interior de edifícios inspirado em ninhos de cupins.

A probóscide de um mosquito inspirando a fabricação de microagulhas de penetração mais eficiente e indolor

3. Alguns exemplos de sucesso da computação evolutiva em aplicações práticas

- Evolução de uma antena que produza o melhor padrão de radiação: *The 2006 NASA ST5 spacecraft antena*.
- https://en.wikipedia.org/wiki/Evolved_antenna



- Evolução de parâmetros de projeto de aeronaves (Multidisciplinary Design Optimization – MDO)
- https://optimization.mccormick.northwestern.edu/index.php/Wing_Shape_Optimization

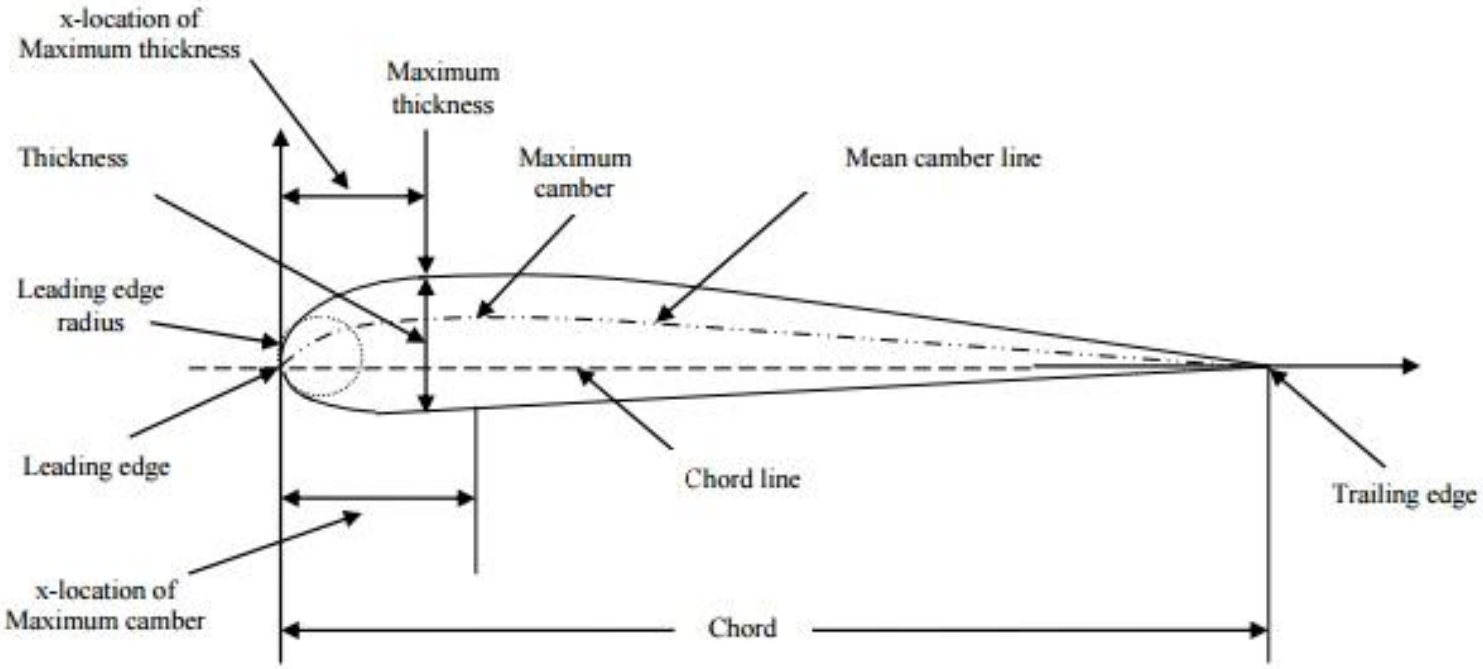
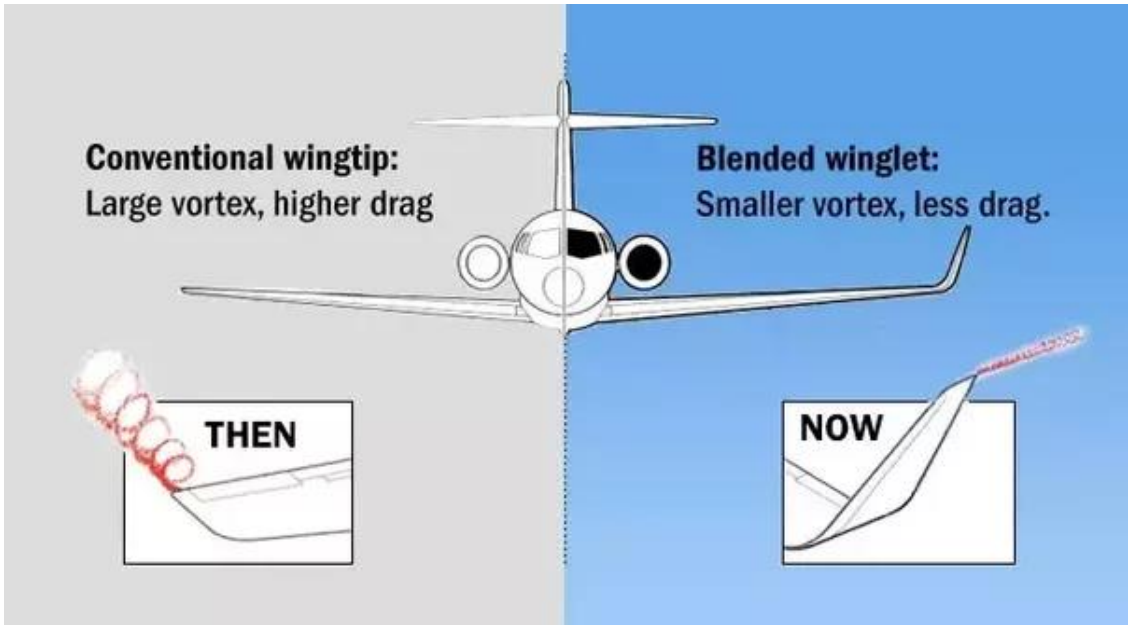


Figure 5.5. Airfoil geometric parameters



http://www.airspacemag.com/flight-today/how-things-work-winglets-2468375/

4. Fundamentos de computação evolutiva

- A computação evolutiva é uma meta-heurística populacional baseada no princípio da seleção natural de Darwin.
- Definição de algoritmos evolutivos: São procedimentos computacionais para a solução de problemas ou modelagem de processos evolutivos, resultantes da aplicação de técnicas heurísticas baseadas na seguinte sequência básica comum: realização de reprodução, imposição de variações aleatórias, promoção de competição e execução de seleção de indivíduos de uma dada população.
- Logo, a evolução é caracterizada basicamente por um processo constituído de 3 passos:

    1. Reprodução com herança genética;
    2. Introdução de variação aleatória em uma população de indivíduos;
    3. Aplicação da "seleção natural" para a produção da próxima geração.

- A variação cria diversidade na população, a diversidade é transmitida por herança e a seleção elimina indivíduos menos aptos, quando comparados aos demais indivíduos da população.
- Quatro operações são fundamentais junto à população de indivíduos: reprodução; variação; atribuição de um valor ou grau de adaptação *relativa* (denominado *fitness*); e seleção de indivíduos que irão compor a próxima geração.
- A disponibilidade de uma quantidade significativa de recursos computacionais é um requisito para viabilizar a implementação computacional de algoritmos evolutivos.
- A seleção natural em si não tem um propósito definido, mas em computador é possível impor um propósito matematicamente definido, por exemplo, para implementar meta-heurísticas de otimização.
- Neste sentido, os algoritmos evolutivos caracterizam-se como meta-heurísticas que abrangem metodologias de busca em espaços de soluções candidatas, capazes de gerenciar operadores computacionais de busca local e de busca global.

- Como praticamente todas as meta-heurísticas, os algoritmos evolutivos geralmente **NÃO** apresentam as seguintes propriedades:
  - ✓ Garantia de obtenção da solução ótima;
  - ✓ Garantia de convergência;
  - ✓ Garantia de custo máximo para se chegar a uma solução.

5. Formalização matemática

- Representação genotípica: Codificação dos candidatos à solução (um subconjunto deles irá compor a população a cada geração). Podem existir vários tipos de representação para o mesmo problema, mas qualquer uma delas deve ser capaz de representar todas as soluções-candidatas, mesmo que nem sempre de forma única.
- Representação fenotípica: Interpretação da representação genotípica.
- Espaço de busca: Tem como elementos todos os possíveis candidatos à solução do problema e é definido a partir da representação genotípica. Dependendo da existência ou não de restrições, pode conter regiões factíveis e infactíveis.

- Função de adaptação ou *fitness*: Atribui a cada elemento do espaço de busca um valor de adaptação, que será usado como medida *relativa* de desempenho. Representa a pressão do ambiente sobre o fenótipo dos indivíduos.
- Operadores de inicialização: Produzem a primeira geração de indivíduos (população inicial), tomando elementos do espaço de busca.
- Operadores genéticos: Implementam o mecanismo de introdução de variabilidade aleatória no genótipo da população.
- Operadores de seleção: Implementam o mecanismo de "seleção natural".
- Índice de diversidade: Geralmente representa a distância média entre os indivíduos da população corrente. Pode ser medido no genótipo, no fenótipo ou levando-se em conta apenas o *fitness* (medida aproximada).

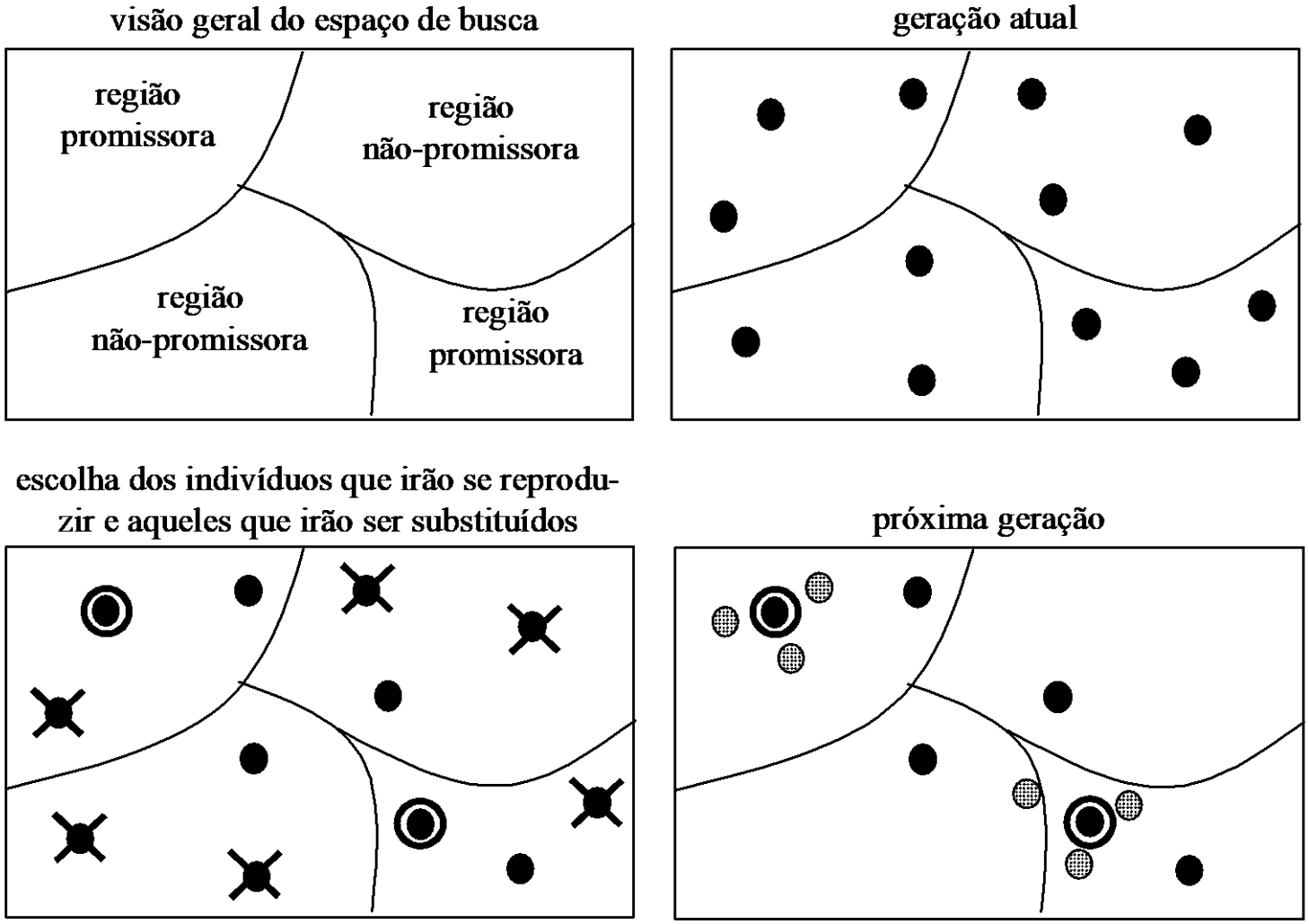
- A codificação genética (representação genotípica) e a definição da função de adaptação ou *fitness* são as etapas mais críticas, embora o desempenho efetivo do processo evolutivo dependa das decisões tomadas em todas as etapas de definição do algoritmo.

- É necessário atribuir valores aos parâmetros envolvidos: tamanho da população; probabilidade de aplicação dos operadores genéticos; argumentos dos operadores de seleção; e argumentos do critério de parada.

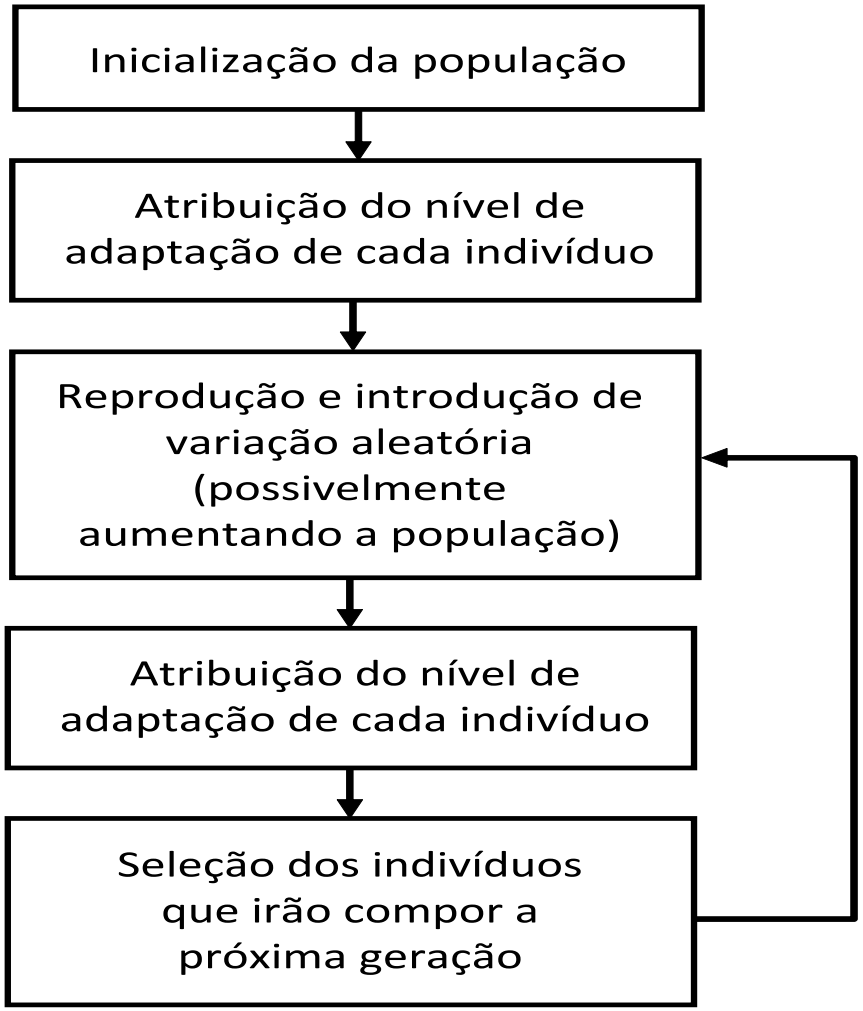
Termos Biológicos	**Termos Computacionais**
Cromossomo	Indivíduo
Gene	Caractere ou atributo
Alelo	Valor do atributo
Lócus	Posição do atributo no cromossomo
Genótipo	Vetor de atributos que representa o indivíduo
Fenótipo	Interpretação do vetor de atributos

- Genótipo: representa o conjunto específico de genes do genoma. Neste caso, indivíduos com o mesmo genoma são ditos terem o mesmo genótipo.
- Fenótipo: é a manifestação do genótipo no comportamento, fisiologia e morfologia do indivíduo, como um produto de sua interação com o ambiente.
- A seleção natural opera somente na expressão fenotípica do genótipo.

6. Visão pictórica da força evolutiva



7. Fluxograma de um algoritmo evolutivo



8. Pseudo-código de um algoritmo evolutivo

Procedimento [*P*] = algoritmo_evolutivo(*N*, *pc, pm*)
  P'' ← inicializa(*N*)
  fit ← avalia(*P''*)
  t ← 1
  Enquanto condição_de_parada for FALSO **faça**,
    P ← seleciona(*P''*, *fit*)
    P' ← reproduz(*P*, *fit, pc*)
    P'' ← varia(*P', pm*)
    fit ← avalia(*P''*)
    t ← t + 1
  Fim Enquanto
Fim Procedimento

- Quais são as diferenças entre este pseudo-código e o fluxograma do slide anterior?

9. As várias faces dos algoritmos evolutivos

- Anos 50 e 60: Cientistas da computação já estudavam sistemas evolutivos com a ideia de que o mecanismo de evolução poderia ser utilizado como uma ferramenta de otimização para problemas de engenharia.

  - FRASER (1959) – "Simulation of Genetic Systems by Automatic Digital Computers" – Australian Journal of Biological Science, 10:484-499.

  - FRIEDBERG (1958) – "A Learning Machine: part I" – IBM Journal, pp. 2-13, Jan.

  - ANDERSON (1953) – "Recent Advances in Finding Best Operating Conditions" – Journal of American Statistic Association, 48, pp. 789-798.

  - BREMERMANN (1962) – "Optimization Through Evolution and Recombination" – in M.C. Yovits, G.T. Jacobi and D.G. Goldstein, editors – *Self-organizing Systems*, Spartan, Washington, D.C., pp. 93-106.

- RECHENBERG (1973) introduziu, nos anos 60, as ***estratégias evolutivas***, as quais foram utilizadas para otimizar parâmetros de valor real em sistemas dinâmicos. As estratégias evolutivas foram aperfeiçoadas por SCHWEFEL (1975; 1977).
- FOGEL, OWENS e WALSH (1966) desenvolveram a ***programação evolutiva***, método em que os candidatos à solução de um dado problema são representados por máquinas de estado finito, as quais evoluem pela mutação aleatória de seus diagramas de transição de estados, seguida pela seleção da mais bem adaptada. Uma formulação mais ampla da programação evolutiva pode ser encontrada em FOGEL (1999).
- Os ***algoritmos genéticos*** foram propostos por Holland nos anos 60, tendo sido desenvolvidos até meados dos anos 70 por seu grupo de pesquisa na Universidade de Michigan. Em contraste com as estratégias evolutivas e a programação evolutiva, o objetivo original de Holland não foi o de desenvolver algoritmos para a solução de problemas específicos, mas sim estudar formalmente os fenômenos de adaptação, naturais ou artificiais, com o propósito de importar estes mecanismos de adaptação

para ambientes computacionais. HOLLAND (1975/1992) apresentou os algoritmos genéticos como uma abstração da evolução biológica, tendo como inovações significativas a utilização conjunta de operadores de recombinação e inversão (além de operadores de mutação) e de um número elevado de indivíduos em cada geração.

- HOLLAND (1975/1992) também apresentou os ***sistemas classificadores***, especificamente implementados para operarem em ambientes não-estacionários, como exigido no caso de agentes autônomos.
- Já no início dos anos 90, KOZA (1992) estendeu as técnicas de algoritmo genético para o espaço de programas computacionais, resultando na ***programação genética***.
  - Em programação genética, os indivíduos que constituem a população sujeita ao processo evolutivo, ao invés de apresentarem cadeias cromossômicas de comprimento fixo, são programas que devem ser executados. A programação genética representa uma iniciativa de se desenvolver métodos para a geração automática de programas computacionais genéricos.

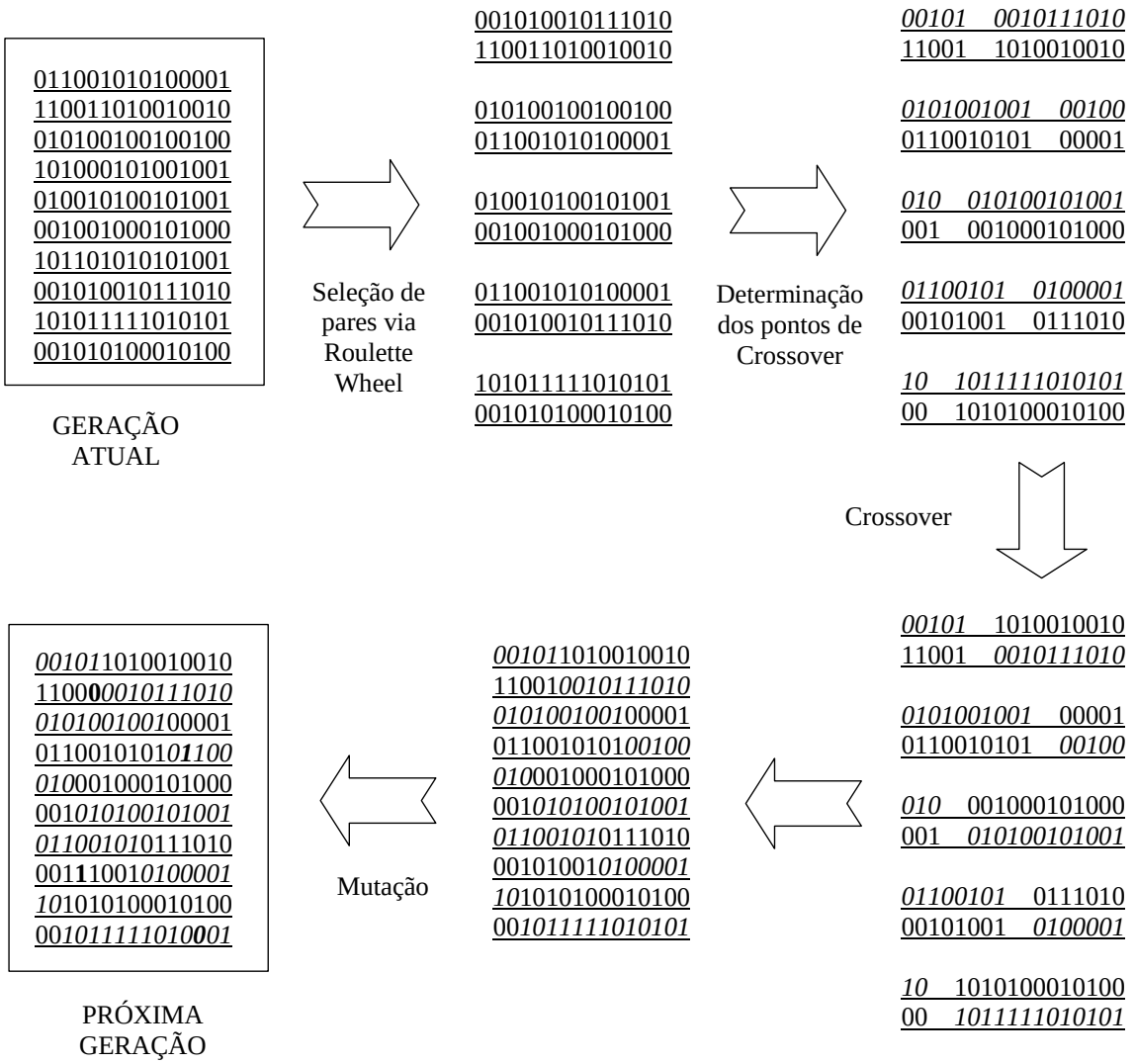
10. Pseudo-código de uma estratégia evolutiva

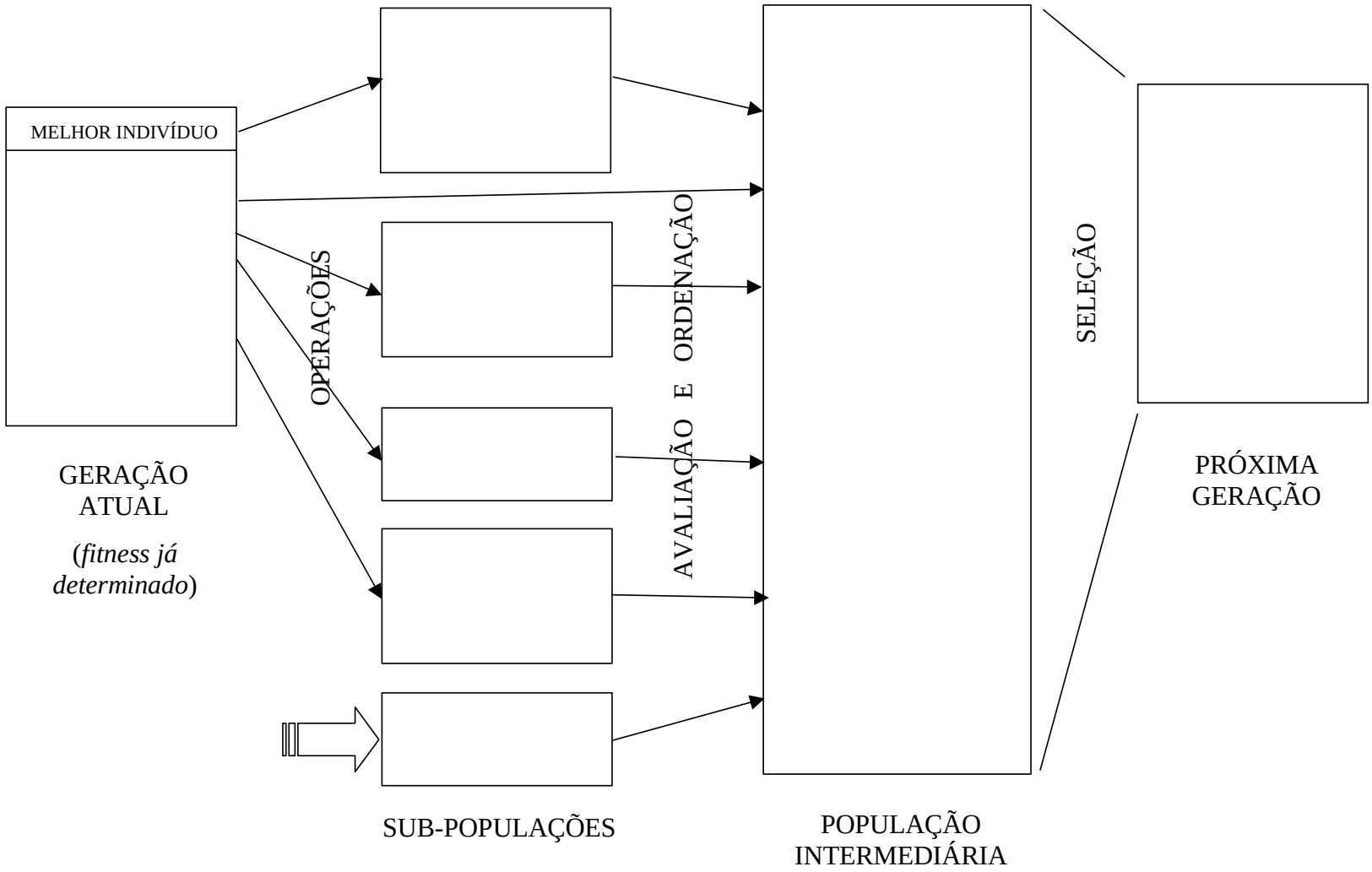
- Voltada para busca em espaços contínuos

Procedimento [*P*] = estrategia_evolutiva(μ,λ)
  inicialize μ indivíduos do tipo: $\boldsymbol{v}_i = (\boldsymbol{x}_i, \boldsymbol{\sigma}_i, \boldsymbol{\theta}_i)$, $i = 1, ..., \mu$
  avalie os indivíduos
  $t \leftarrow 1$
  Enquanto condição_de_parada for FALSA **faça**,
    gere λ indivíduos (clonagem + mutação) a partir dos μ indivíduos
    avalie os indivíduos gerados
    selecione os μ melhores indivíduos de λ ou $\mu+\lambda$
  Fim Enquanto
Fim Procedimento

- Sugere-se o emprego da seguinte relação: $1 \leq \mu \leq \lambda$
- Repare que cada indivíduo contém instruções sobre como aplicar a mutação aos seus descendentes e estes devem herdar estas instruções, com alguma variabilidade. A mutação emprega uma distribuição normal multivariada.

11. Um fluxograma de um algoritmo genético





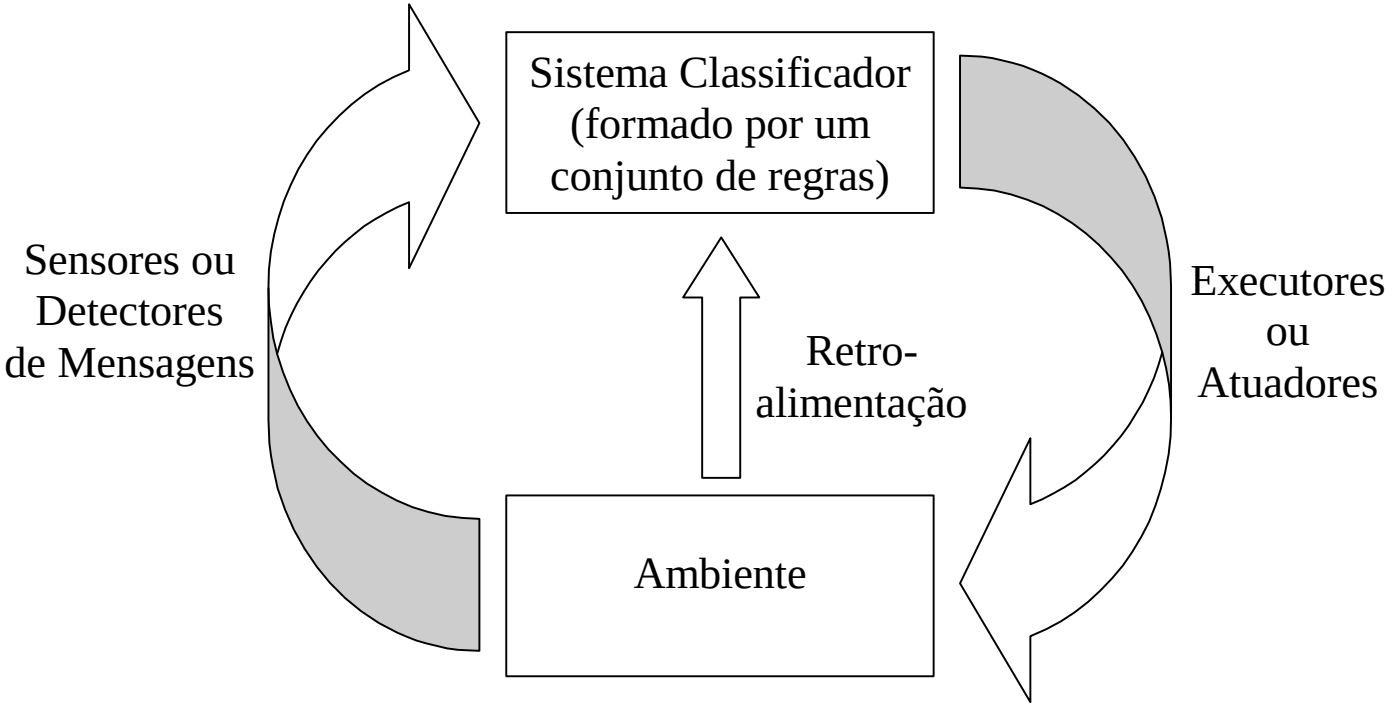
Uma proposta de extensão de um algoritmo genético

12. Sistemas classificadores

- Sistemas Classificadores representam metodologias para criação e atualização evolutiva de regras (denominadas classificadores) em um sistema de tomada de decisão.
- Dadas as características de um ambiente em um determinado instante e levando-se em conta a "energia" de cada classificador (regra), alguns classificadores podem ser ativados. Eles codificam alternativas de ações específicas, as quais são submetidas a um processo de competição para selecionar aquela que será executada.
- Dependendo do efeito de cada ação (ou sequência de ações) no atendimento dos objetivos (os quais podem estar implícitos), os classificadores responsáveis pelas ações serão recompensados ou punidos (ganhando ou perdendo "energia").
- Periodicamente, o elenco de classificadores é submetido a um processo evolutivo, que toma basicamente como medida de *fitness* a "energia" dos classificadores.
- As aplicações de Sistemas Classificadores se estendem desde a síntese de

controladores autônomos para robôs, passando por mineração de dados e otimização de formas geométricas de instrumentos industriais.

- A interação com o ambiente ocorre através da troca de mensagens. As mensagens provenientes do ambiente procuram retratar o seu estado atual. Já as mensagens advindas do Sistema Classificador retratam ações a serem executadas.



12.1 Algoritmo simplificado de geração de um sistema classificador

Baseado em: Vargas, P.A. "Sistemas Classificadores para Redução de Perdas em Redes de Distribuição de Energia Elétrica", Tese de Mestrado, FEEC – Unicamp, 2000.

INÍCIO DO PROCESSO EVOLUTIVO

INÍCIO DE UMA ÉPOCA DE ITERAÇÕES

Passo 1 : Receber mensagem do ambiente.
Passo 2 : Codificar mensagem.
Passo 3 : Selecionar classificadores que se identificam com a mensagem.
Passo 4 : Iniciar "Competição" :
- Calcular "bid" de cada competidor.
- Apontar vencedor.
- Cobrar taxas dos participantes e do vencedor.

Passo 5 : Atuar sobre o ambiente.
Passo 6 : Receber mensagem (retroalimentação) do ambiente.
Passo 7 : Codificar mensagem.
Passo 8 : Recompensar ou punir o classificador vencedor.
Passo 9 : Cobrar taxa de vida de todos os indivíduos.
Passo 10 : Caso não seja fim de uma "Época", retornar ao **Passo 3.**

FIM DE UMA ÉPOCA DE ITERAÇÕES

Passo 11 : Selecionar indivíduos mais aptos para aplicação do algoritmo genético.
Passo 12 : Aplicar os operadores de crossover e mutação para gerar os filhos.
Passo 13 : Selecionar os indivíduos mais fracos da população.
Passo 14 : Inserir os filhos gerados no **Passo 12** na população, substituindo os indivíduos selecionados no **Passo 13**.
Passo 15 : Caso não seja fim do processo evolutivo, voltar ao **Passo 1.**

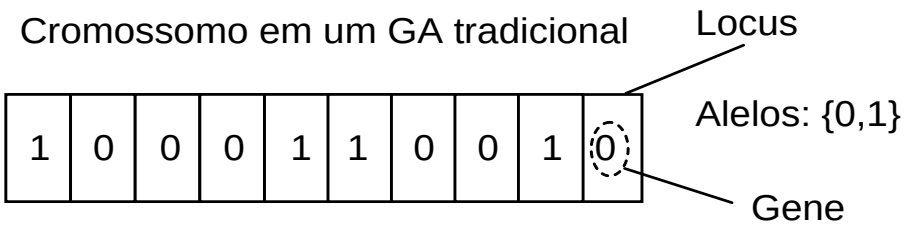
FIM DO PROCESSO EVOLUTIVO

13. Codificação e operadores genéticos

- Na grande maioria dos casos, os indivíduos da população de um algoritmo evolutivo, os quais representam soluções candidatas (isso na abordagem Pittsburgh, em que cada indivíduo corresponde a uma proposta de solução para o problema), são representados computacionalmente por uma única lista ou vetor de atributos de tamanho fixo. Logo, esta lista de atributos representa o código genético do indivíduo (genótipo), caracteriza unicamente o fenótipo e é geralmente denominada de cromossomo.
- Em alguns casos, a lista de atributos não tem tamanho fixo e em outros casos o código genético requer estruturas de dados mais elaboradas, como uma matriz ou uma árvore de atributos.
- Esses atributos podem ser de três tipos: binários, inteiros ou reais (em ponto flutuante). E num mesmo cromossomo pode aparecer um único tipo de atributo ou então mais de um tipo simultaneamente.
- **Para cada tipo de atributo, existem operadores genéticos específicos.**

- Normalmente, os operadores genéticos correspondem às operações de mutação gênica e recombinação gênica (tendo o crossover como um caso particular), que atuam sobre os cromossomos da população.
- Certos problemas admitem múltiplas codificações alternativas e a escolha de uma delas deve levar em conta a competência em explorar o espaço de busca associado.
- De fato, ao definir a codificação do indivíduo (solução candidata), fica completamente caracterizado o espaço de busca e fica estabelecida a relação de vizinhança entre as soluções candidatas. Como foi mencionado no slide anterior, para cada proposta de codificação existem operadores genéticos específicos.
- Para poder abordar um problema via computação evolutiva, é necessário definir:
    1. Uma codificação para as soluções candidatas;
    2. Uma função de avaliação (função de *fitness*);
    3. Operadores genéticos;
    4. Mecanismos de seleção;
    5. Taxas e parâmetros do algoritmo;
    6. Critérios de parada.

- Caso tradicional: Cadeia binária de comprimento fixo, supondo que cada indivíduo é representado por um único cromossomo.



13.1 Operadores de mutação para codificação binária e em ponto flutuante

- 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 locus **na população *P*** (repare que não está sendo considerado aqui apenas um indivíduo, mas todos os indivíduos ao mesmo tempo, enfileirados e com índices sequenciais) é operado da seguinte forma:
  - Os números $r,\ldots,u$, indicando as posições que irão sofrer mutação, são determinados aleatoriamente de forma que cada posição possui uma pequena probabilidade p_m 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 determinados aleatoriamente, partindo do conjunto de alelos para cada gene.
- Estudos empíricos e teóricos sugerem que:
  - Um valor inicial maior para a mutação deve ser adotado e decrescido geometricamente ao longo das gerações (FOGARTY, 1989);
  - Um limite inferior $p_m = 1/dim\_espaço\_de\_busca$ para a taxa de mutação pode ser empregado (MÜHLENBEIN, 1992; BÄCK, 1993).

- No caso de cadeias binárias, se uma posição escolhida para mutação possui alelo ‘0’ então ela se torna ‘1’, e vice-versa.
- No caso de cadeias com valores em ponto flutuante, é geralmente inserida uma perturbação com distribuição normal e média zero na posição escolhida para mutação. O desvio-padrão da distribuição normal deve iniciar maior e ir caindo ao longo das gerações, para tornar a busca mais refinada conforme a população se aproxima de ótimos locais.
- Este é o caso da chamada *mutação não-uniforme* (MICHALEWICZ, 1996; MICHALEWICZ & SCHOENAUER, 1996). Pode-se defini-la da seguinte forma: Seja $\mathbf{x}^t = [x_1 \ldots x_n]$ um cromossomo e suponha que o elemento x_k foi selecionado para mutação; o cromossomo resultante será $\mathbf{x}^{t+1} = [x_1 \ldots x'_k \ldots x_n]$, onde

$$x'_k = \begin{cases} x_k + \Delta(t, a - x_k), & \text{com } 50\% \text{ de probabilidade} \\ x_k - \Delta(t, x_k - b), & \text{com } 50\% \text{ de probabilidade} \end{cases}$$

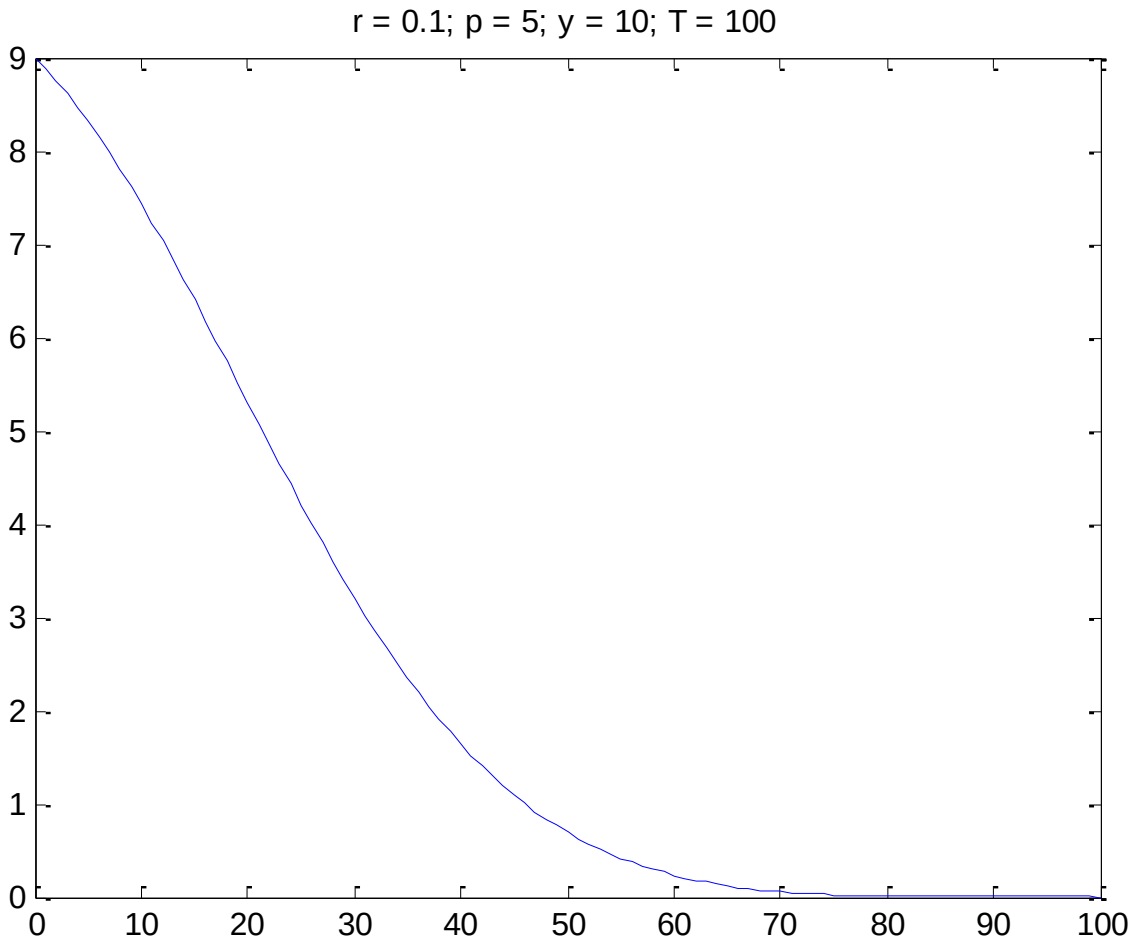
onde a e b são os limites inferiores e superiores da variável x_k. A função $\Delta(t, y)$ retorna um valor no intervalo $[0, y]$ tal que a probabilidade de $\Delta(t, y)$ ser próximo de zero aumenta à medida que t aumenta.

- Esta propriedade faz com que este operador inicialmente explore o espaço de busca de forma mais ampla (quando t é pequeno) e de forma mais refinada (localmente) em gerações avançadas (quando t é grande);
- MICHALEWICZ (1996) propõe a seguinte função:

$$\Delta(t, y) = y \cdot \left(1 - r^{(1-t/T)^p}\right)$$

onde r é um número aleatório no intervalo [0, 1], T é o número máximo de gerações e p é um parâmetro que determina o grau de dependência do número de iterações (valor proposto por MICHALEWICZ (1996): $p = 5$).

- A figura a seguir ilustra o comportamento desta função para valores específicos de seus parâmetros.


r = 0.1; p = 5; y = 10; T = 100
9
8
7
6
5
4
3
2
1
0
0
10
20
30
40
50
60
70
80
90
100

13.2 Distribuição normal a partir de uma distribuição uniforme

- Geralmente, as linguagens de programação disponibilizam geradores de números pseudo-aleatórios com distribuição uniforme.
- Como é menos frequente a disponibilidade de geradores de números pseudo-aleatórios com distribuição normal, é possível implementá-los a partir da transformação de Box-Muller (BOX & MULLER, 1958).
- Dispondo de um gerador com distribuição uniforme, uma distribuição normal de média μ e variância σ^2, $x \sim N(\mu, \sigma^2)$, pode ser obtida na forma:

$$x = \mu + \sigma\sqrt{-2\log_e(u_1)}\cos(2\pi u_2)$$

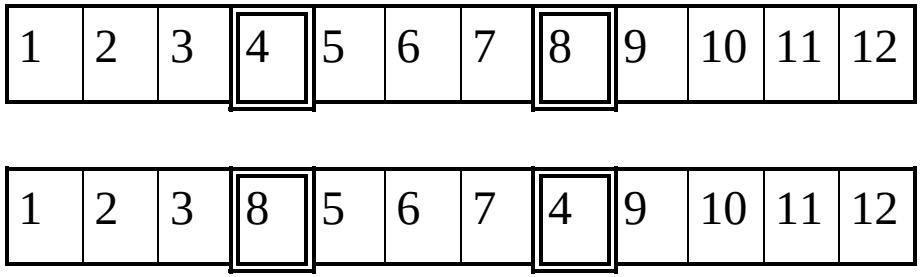
ou então

$$x = \mu + \sigma\sqrt{-2\log_e(u_1)}\,\mathrm{sen}(2\pi u_2)$$

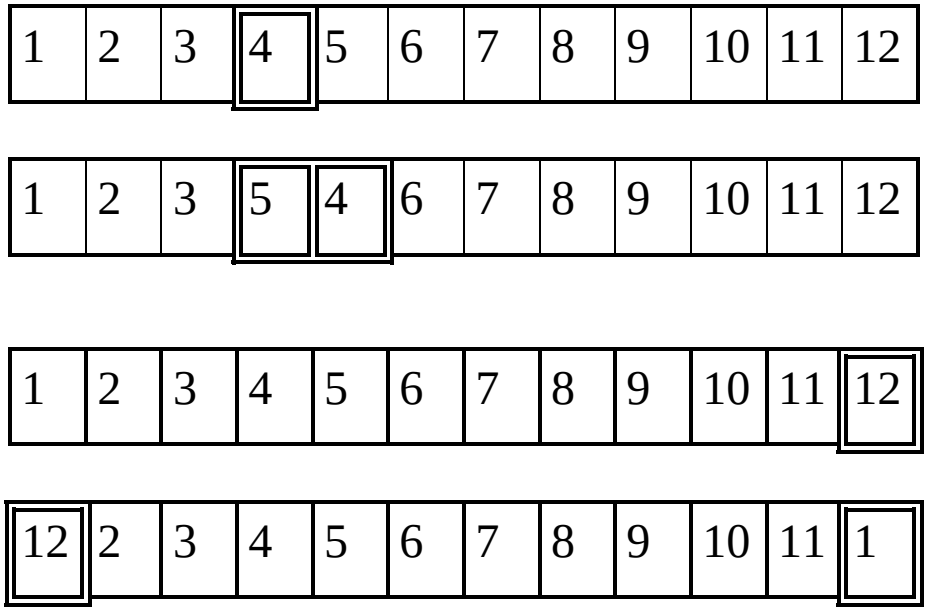
com $u_1, u_2 \sim U(0,1)$.

13.3 Operadores de mutação para permutações (codificação inteira)

- Em problemas como o caixeiro viajante, a codificação inteira utilizada para representar a sequência de cidades leva diretamente a um problema de permutação de índices.
- Nesses casos, deve-se buscar operadores de mutação específicos, que troquem genes de posição, de modo a manter a solução candidata como uma permutação dos mesmos índices que existiam antes da aplicação do operador.
- Exemplos:
  - Mutação Inversiva



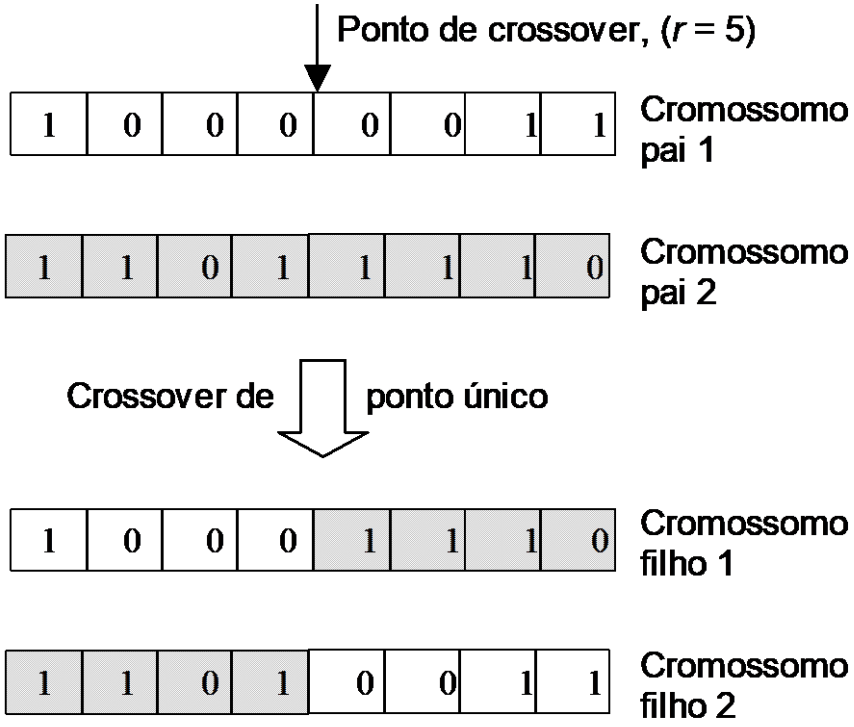
- Variação de Mutação Inversiva



13.4 Operadores de recombinação para codificação binária

- Nos sistemas biológicos, a recombinação (ou 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 tradicional (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 p_c.
  - 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 cromossomos gerados $\boldsymbol{x'}$ e $\boldsymbol{y'}$ são os descendentes de $\boldsymbol{x}$ e $\boldsymbol{y}$.
- Este processo é ilustrado na figura a seguir, sendo denominado de crossover de um ponto.
- É possível estender este operador para considerar múltiplos pontos de corte.



- O operador de recombinação mais utilizado em codificação binária é o **crossover uniforme**, que toma um gene de cada progenitor com igual probabilidade, para formar um novo cromossomo.

13.5 Operadores de recombinação para codificação em ponto flutuante

- O operador mais utilizado para codificação em ponto flutuante é o crossover aritmético. Ele realiza uma combinação convexa dos valores dos genes dos progenitores para formar os descendentes.
- Portanto, dado um valor aleatório $a \in [0,+1]$ e dados dois cromossomos $\boldsymbol{x}$ e $\boldsymbol{y}$, o descendente após o crossover aritmético é dado por: $a\boldsymbol{x} + (1-a)\boldsymbol{y}$.
- Existe uma variante em que é escolhido um valor de a para cada gene do cromossomo e não para todo o cromossomo, como acima.
- Também é possível fazer com que o valor de a pertença ao intervalo $a \in [-\varepsilon,+1+\varepsilon]$, com $\varepsilon>0$ pequeno, permitindo que o descendente não esteja restrito à região fechada que é delimitada pela combinação convexa de $\boldsymbol{x}$ e $\boldsymbol{y}$.
- Mais opções: *Crossover heurístico* (WRIGHT, 1994); *crossover geométrico, crossover esférico* e *crossover simplex* descritos em MICHALEWICZ & SCHOENAUER (1996). Veja também BÄCK *et al.* (2000a).

13.6 Operador de recombinação para permutações (codificação inteira)

- Novamente por causa do problema de permutação de índices vinculado, por exemplo, ao problema do caixeiro viajante, deve-se buscar operadores de recombinação específicos, de modo a manter a solução candidata como uma permutação dos mesmos índices que existiam antes da aplicação do operador.
  - Exemplo: Crossover OX

1	2	3	4	5	6	7	8	9	10	11	12

7	3	1	11	4	12	5	2	10	9	6	8

			4	5	6	7					

1	11	12	4	5	6	7	2	10	9	8	3

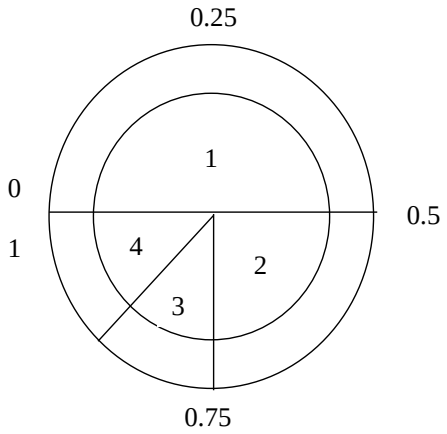
14. Operadores de seleção

- É sugerido que o grau de adaptação a ser atribuído a cada indivíduo da população (solução candidata) seja não-negativo e esteja restrito a um intervalo finito de valores.
- Os operadores de seleção podem ser empregados em diversas etapas de um algoritmo evolutivo, e as principais propostas são:
  - ✓ Algoritmo da roleta (roulette wheel): seleção proporcional ao grau de adaptação (*fitness*);
  - ✓ Torneio de p jogadores ($p \geq 2$): ajuda a preservar a diversidade para valores pequenos de p. Um aumento de p aumenta a pressão seletiva;
  - ✓ *Rank*: utiliza a roleta, mas adota apenas a ordem dos indivíduos, de acordo com o grau de adaptação, para definir a probabilidade de escolha, que é fixada a priori;
  - ✓ Seleção biclassista ou elitista: Na seleção bi-classista, são escolhidos os b% melhores indivíduos e os w% piores indivíduos da população. O restante (100–(b+w))% é selecionado aleatoriamente, com ou sem reposição.

14.1 O algoritmo da roleta

- A seleção no GA tradicional 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.

14.2 Seleção Por Torneio

- É um dos mais refinados e bem-sucedidos processos de seleção, por permitir ajustar a pressão seletiva.
- Para se selecionar N indivíduos, realizam-se N torneios envolvendo p indivíduos em cada torneio, escolhidos sem levar em conta o fitness e com reposição (um indivíduo pode aparecer mais de uma vez num mesmo torneio). Vence cada torneio aquele que apresentar o maior fitness (comparado ao de seu(s) oponente(s) no torneio). Repare que a definição do vencedor do torneio é determinística.
- Para propósitos práticos, $p \geq 10$ conduz a uma forte pressão seletiva, enquanto valores de p entre 3 e 5 levam a uma fraca pressão seletiva.
    - Para $q = 1$, nenhuma seleção está sendo feita.
    - Para $q = 2$, tem-se o chamado torneio binário.

15. Operadores de busca local

- Qualquer algoritmo evolutivo pode ter seu desempenho melhorado com a introdução de operadores de busca local, os quais geralmente não apresentam motivação biológica e são específicos para cada problema.
- Particularmente no caso de espaços discretos, a implementação da busca local requer a definição de uma vizinhança. Quanto maior a vizinhança, maior tende a ser o custo da busca local.
- Evidentemente, a busca local só será útil se apresentar uma boa relação de custo/desempenho.
- Tipos de algoritmos de busca local em espaços discretos:
  - ✓ *First improvement*: interrompe a busca ao obter a primeira melhora;
  - ✓ *Best improvement*: consulta toda a vizinhança e fica com a melhor solução;
  - ✓ *Best improvement with limited resources*: o mesmo que o anterior, mas restringe o uso de recursos computacionais (pode não consultar toda a vizinhança).

16. Efeitos da mutação e do tamanho da população

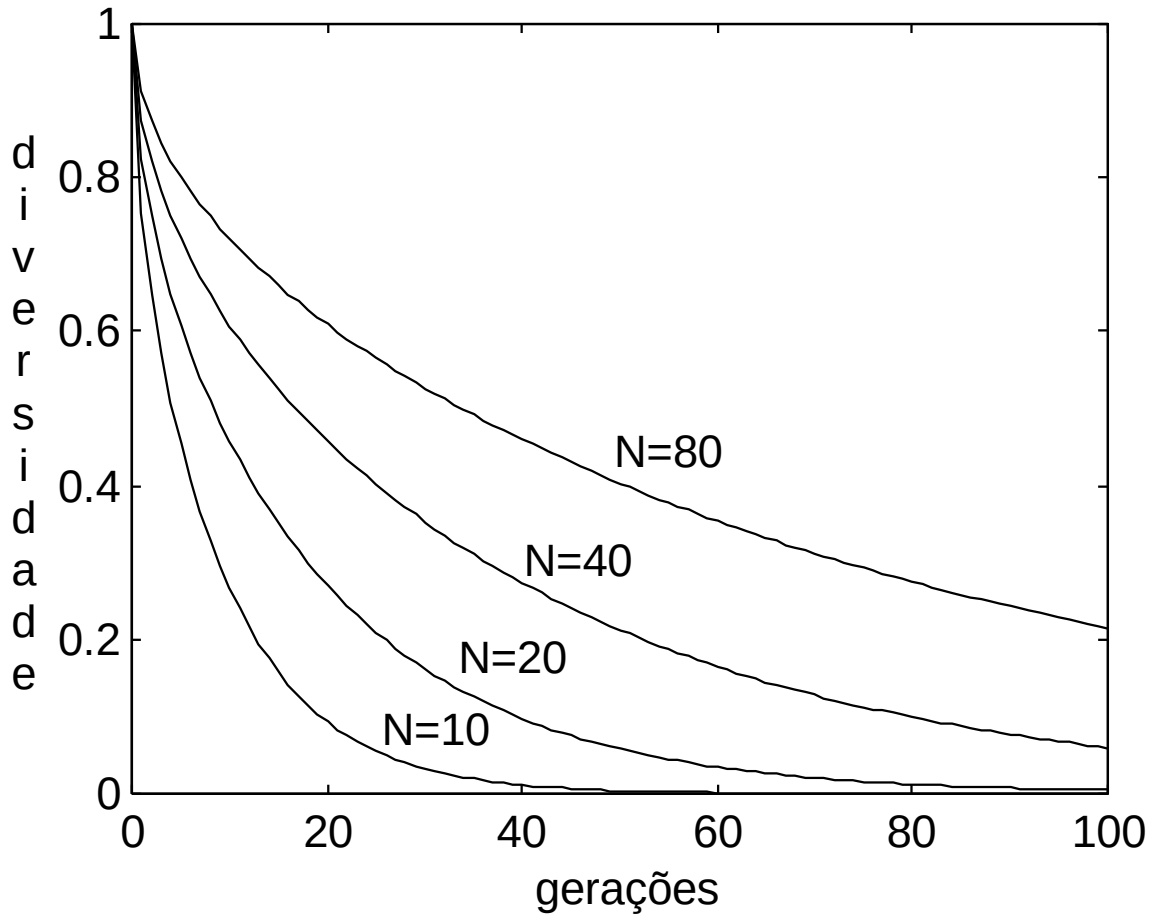


Figura 1 – Deriva genética (*genetic drift*): queda da diversidade mesmo **na ausência de pressão seletiva** (N é o tamanho da população e não há mutação).

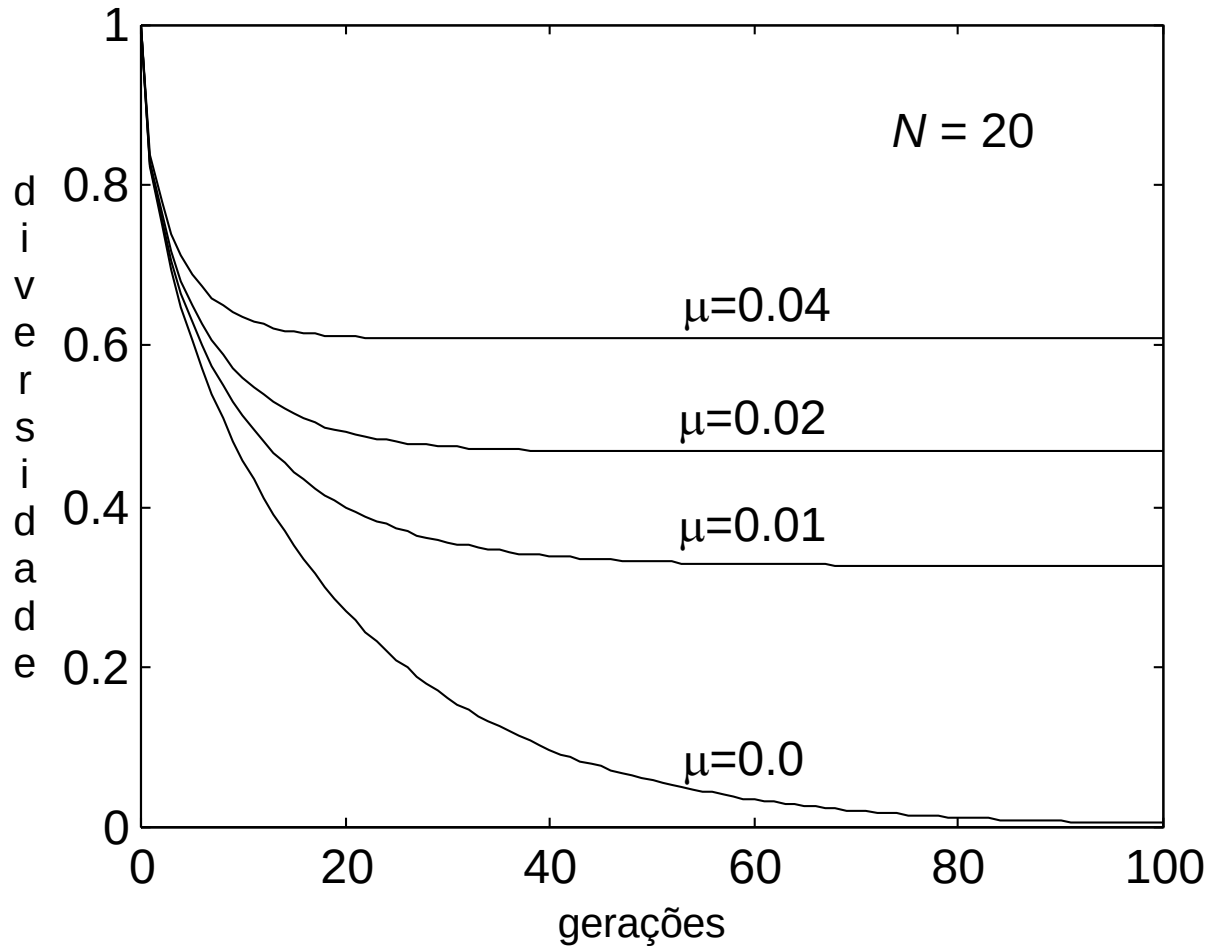


Figura 2 – Deriva genética (*genetic drift*): queda da diversidade mesmo **na ausência de pressão seletiva**, para diferentes taxas de mutação.

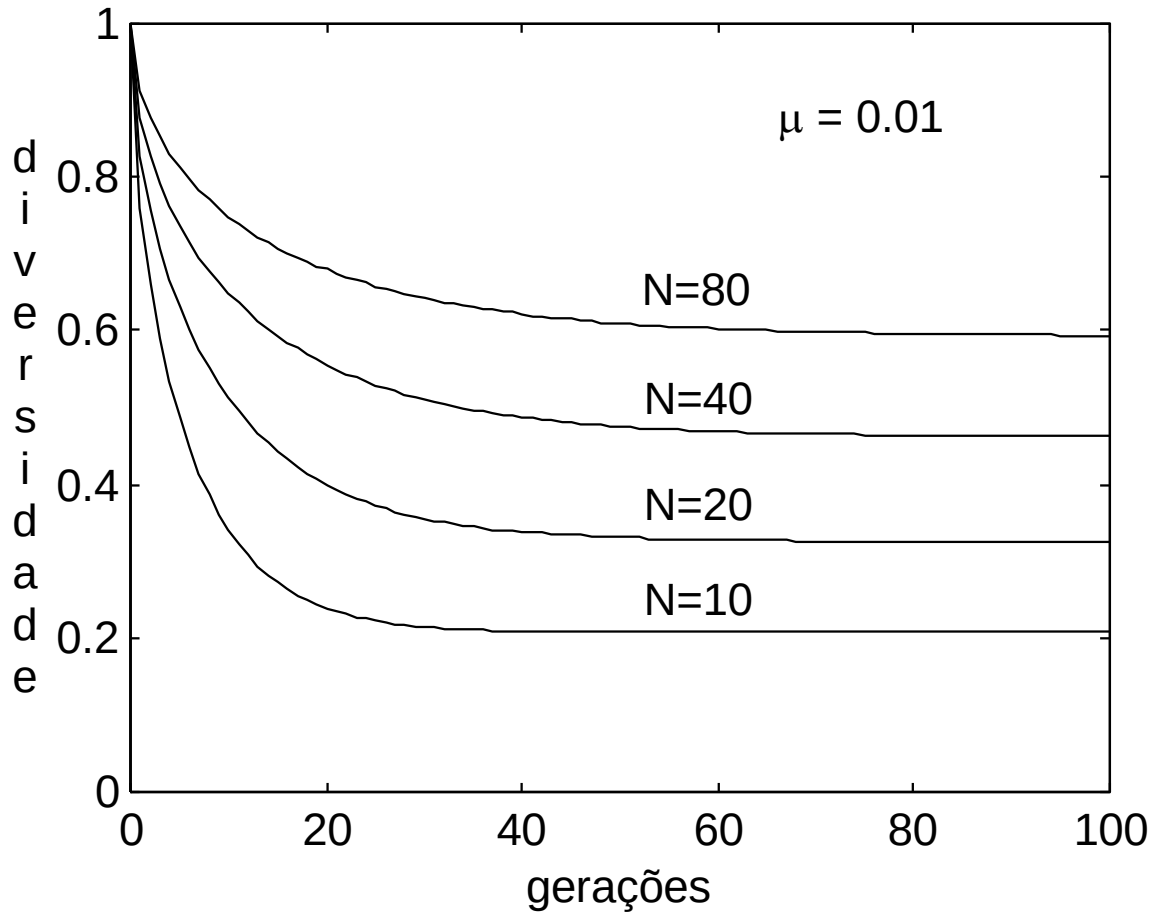


Figura 3 – Deriva genética (*genetic drift*): queda da diversidade mesmo **na ausência de pressão seletiva**, para diferentes tamanhos de população e com taxa de mutação fixa.

17. Exemplos de aplicação: caso discreto

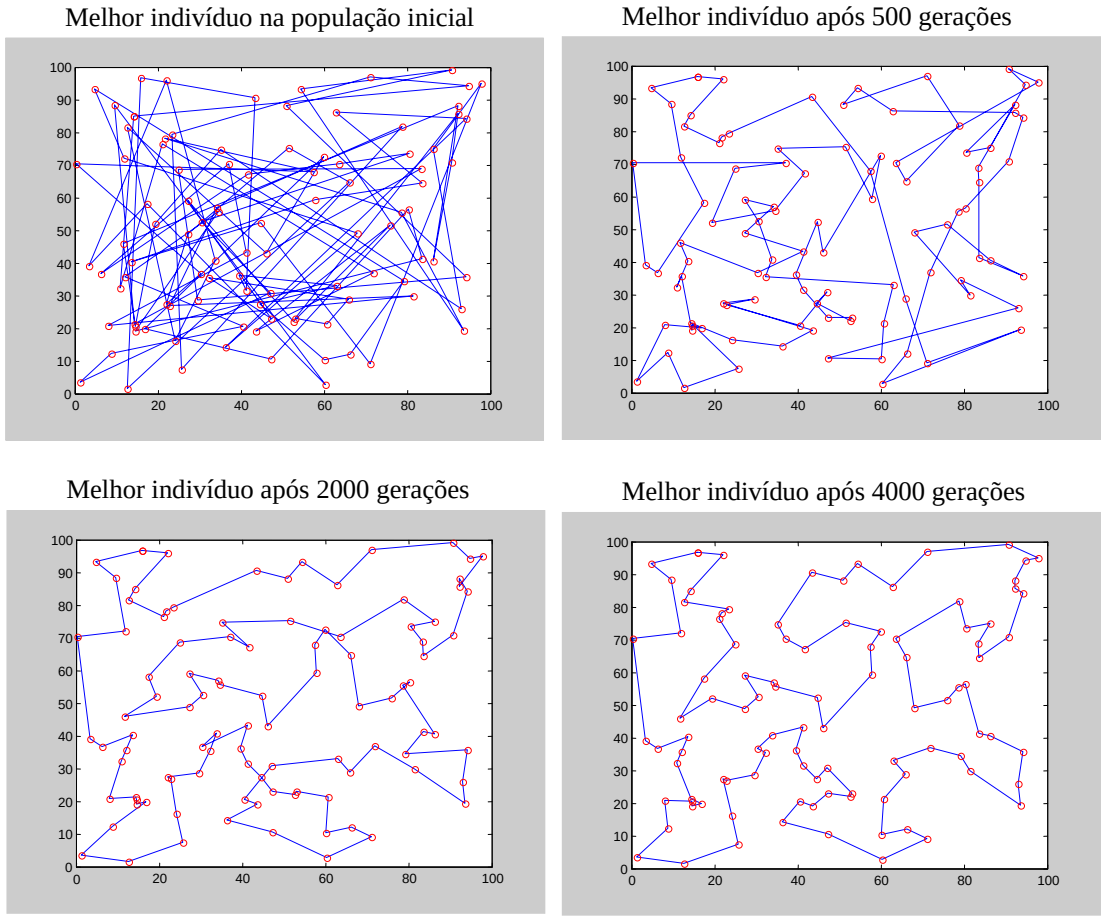
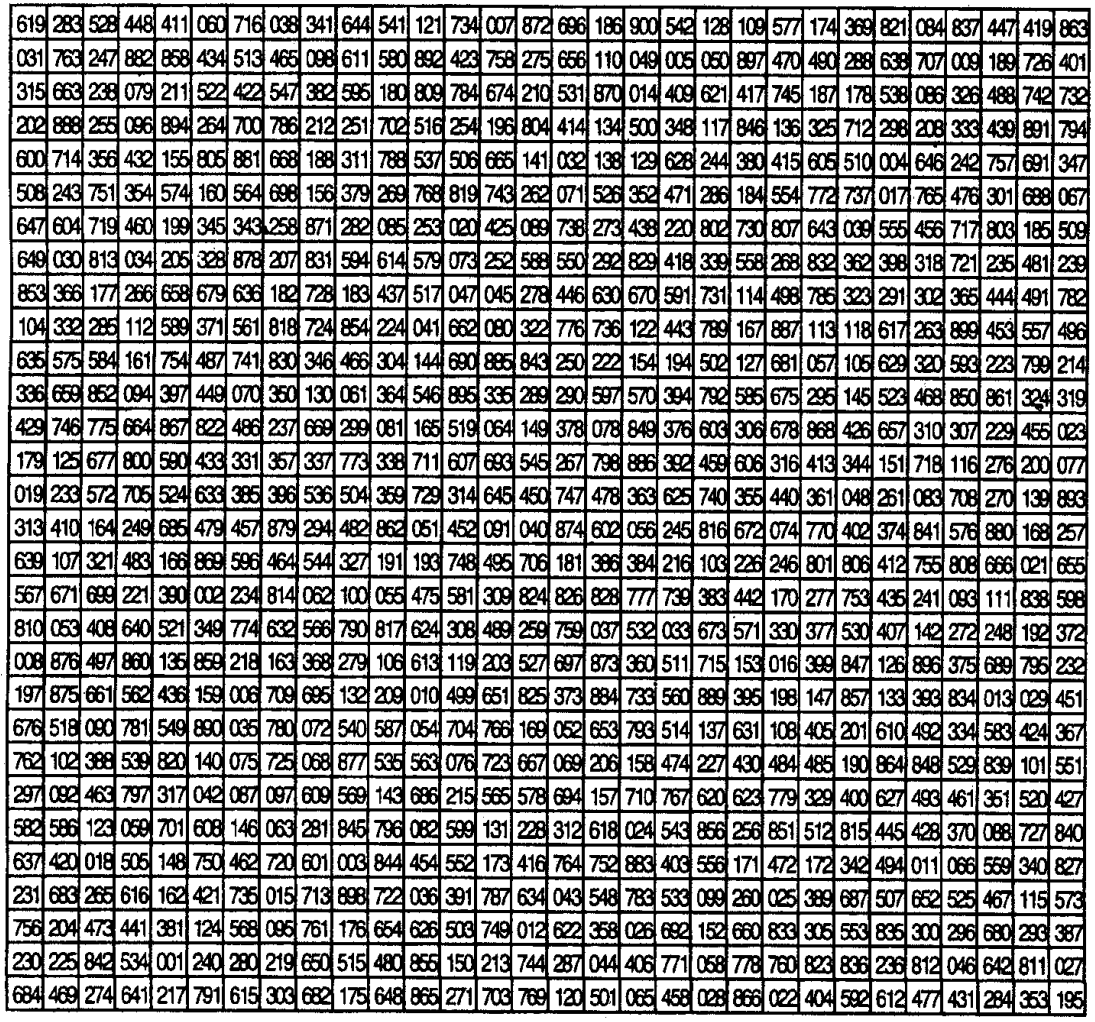


Figura 4 – Solução de um problema de caixeiro viajante



619	283	528	448	411	060	716	038	341	644	541	121	734	007	872	696	186	900	542	128	109	577	174	369	821	084	837	447	419	863
031	763	247	882	858	434	513	465	098	611	580	892	423	758	275	656	110	049	005	050	897	470	490	288	638	707	009	189	726	401
315	663	238	079	211	522	422	547	382	595	180	809	784	674	210	531	870	014	409	621	417	745	187	178	538	086	326	488	742	732
202	898	255	096	894	264	700	786	212	251	702	516	254	196	804	414	134	500	348	117	846	136	325	712	298	208	333	439	891	794
600	714	356	432	155	805	881	668	188	311	788	537	506	665	141	032	138	129	628	244	380	415	605	510	004	646	242	757	691	347
508	243	751	354	574	160	564	698	156	379	269	768	819	743	262	071	526	352	471	286	184	554	772	737	017	765	476	301	698	067
647	604	719	460	199	345	343	258	871	282	085	253	020	425	089	738	273	438	220	802	730	807	643	039	555	456	717	803	185	509
649	030	813	034	205	328	878	207	831	594	614	579	073	252	588	550	292	829	418	339	558	268	832	362	398	318	721	235	481	239
853	366	177	266	658	679	636	182	728	183	437	517	047	045	278	446	630	670	591	731	114	498	785	323	291	302	365	444	491	782
104	332	285	112	589	371	561	818	724	854	224	041	662	080	322	776	736	122	443	789	167	887	113	118	617	263	899	453	557	496
635	575	584	161	754	487	741	830	346	466	304	144	690	885	843	250	222	154	194	502	127	681	057	105	629	320	593	223	799	214
336	659	852	094	397	449	070	350	130	061	364	546	895	335	289	290	597	570	394	792	586	675	295	145	523	468	850	861	324	319
429	746	775	664	867	822	486	237	669	299	081	165	519	064	149	378	078	849	376	603	306	678	868	426	657	310	307	229	455	023
179	125	677	800	590	433	331	357	337	773	338	711	607	693	545	267	798	886	392	459	606	316	413	344	151	718	116	276	200	077
019	233	572	705	524	633	385	396	536	504	359	729	314	645	450	747	478	363	625	740	355	440	361	048	261	083	708	270	139	893
313	410	164	249	685	479	457	879	294	482	862	051	452	091	040	874	602	056	245	816	672	074	770	402	374	841	576	880	168	257
639	107	321	483	166	869	596	464	544	327	191	193	748	495	706	181	386	384	216	103	226	246	801	806	412	755	808	666	021	655
567	671	699	221	390	002	234	814	062	100	055	475	581	309	824	826	828	777	739	383	442	170	277	753	435	241	093	111	838	598
810	053	408	640	521	349	774	632	566	790	817	624	308	489	259	759	037	532	033	673	571	330	377	530	407	142	272	248	192	372
008	876	497	860	135	859	218	163	368	279	106	613	119	203	527	697	873	360	511	715	153	016	399	847	126	896	375	689	795	232
197	875	661	562	436	159	006	709	695	132	209	010	499	651	825	373	884	733	560	889	395	198	147	857	133	393	834	013	029	451
676	518	090	781	549	890	035	780	072	540	587	054	704	766	169	052	653	793	514	137	631	108	405	201	610	492	334	583	424	367
762	102	388	539	820	140	075	725	068	877	535	563	076	723	667	069	206	158	474	227	430	484	485	190	864	848	529	839	101	551
297	092	463	797	317	042	087	097	609	569	143	686	215	565	578	694	157	710	767	620	623	779	329	400	627	493	461	351	520	427
582	586	123	059	701	608	146	063	281	845	796	082	599	131	228	312	618	024	543	856	256	851	512	815	445	428	370	088	727	840
637	420	018	505	148	750	462	720	601	003	844	454	552	173	416	764	752	883	403	556	171	472	172	342	494	011	066	559	340	827
231	683	265	616	162	421	735	015	713	898	722	036	391	787	634	043	548	783	533	099	260	025	389	687	507	652	525	467	115	573
756	204	473	441	381	124	568	095	761	176	654	626	503	749	012	622	358	026	692	152	660	833	305	553	835	300	296	680	293	387
230	225	842	534	001	240	280	219	650	515	480	855	150	213	744	287	044	406	771	058	778	760	823	836	236	812	046	642	811	027
684	469	274	641	217	791	615	303	682	175	648	865	271	703	769	120	501	065	458	028	866	022	404	592	612	477	431	284	353	195

Figura 5 – Solução do problema do quadrado mágico (apenas como motivação, pois já existem algoritmos mais eficientes, que exploram particularidades do problema)

18. Exemplo de aplicação: caso contínuo

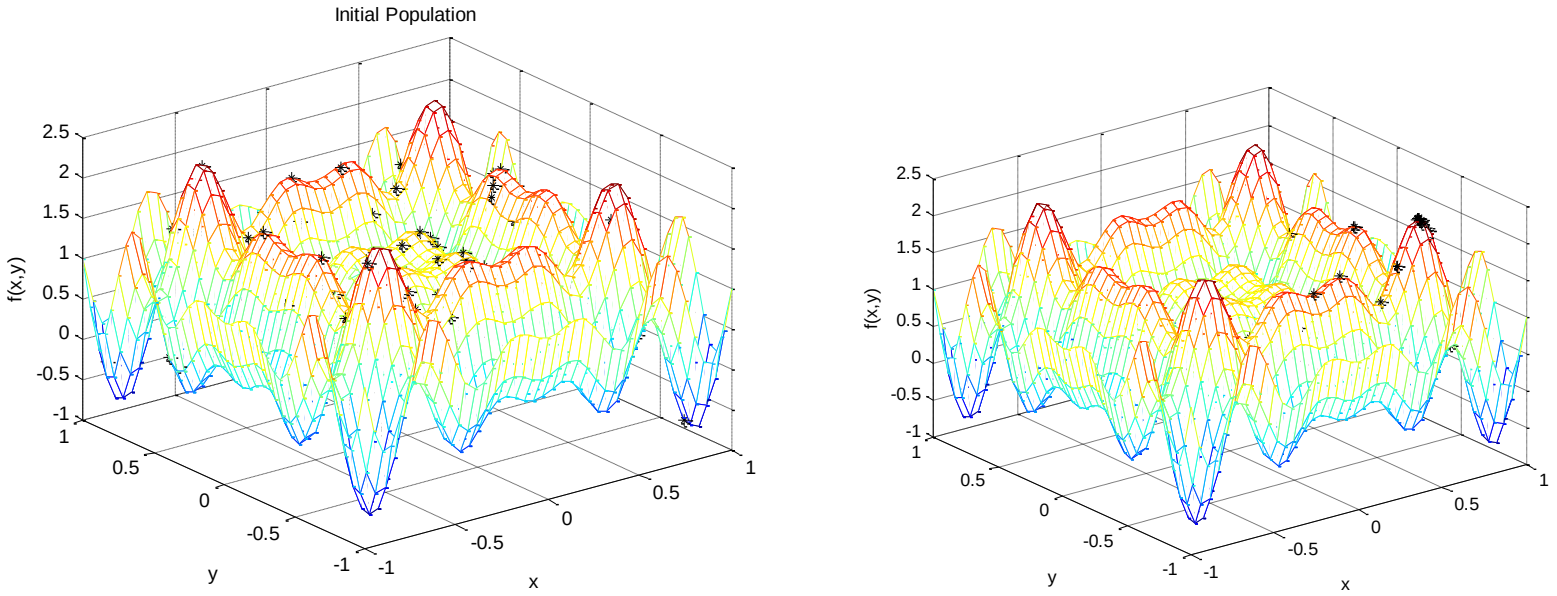


Figura 6 – Superfície de otimização multimodal
População inicial (à esquerda) e final (à direita)

- Embora haja 4 picos de igual qualidade, o algoritmo genético tende a deslocar toda a população para apenas um deles (portanto, não faz busca multimodal).

19. Exemplo de codificação em matriz

Prado, O. G. "Computação Evolutiva Empregada na Reconstrução de Árvores Filogenéticas", Dissertação de Mestrado, FEEC/Unicamp, 2001.

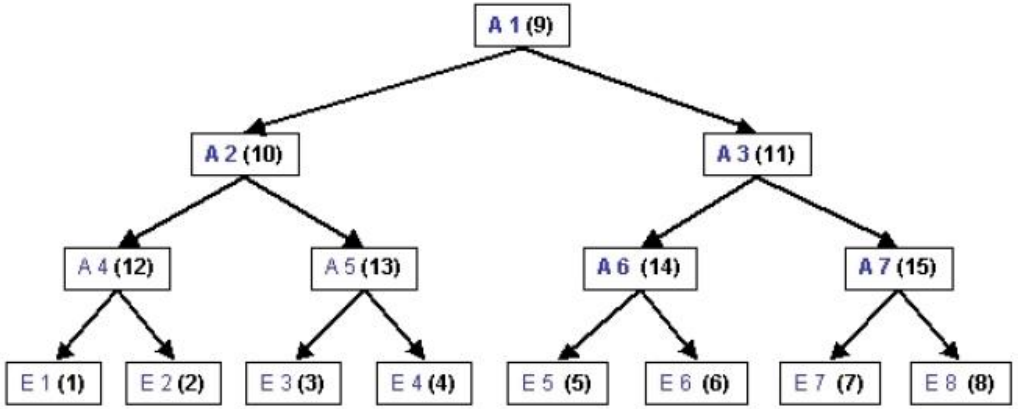


Figura 2.7 Uma possível representação gráfica da árvore filogenética de oito espécies, onde "E n" significa "espécie n" e "A n" significa "Ancestral n".

Filhos

Pais

	E1 (1)	E2 (2)	E3 (3)	E4 (4)	E5 (5)	E6 (6)	E7 (7)	E8 (8)	A1 (9)	A2 (10)	A3 (11)	A4 (12)	A5 (13)	A6 (14)	A7 (15)
A 1 (9)	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
A 2 (10)	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
A 3 (11)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
A 4 (12)	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A 5 (13)	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
A 6 (14)	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
A 7 (15)	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Figura 2.8 Representação na forma de matriz de adjacências da árvore da Figura 2.7, onde "E n" significa "espécie n" e "A n" significa "Ancestral n".

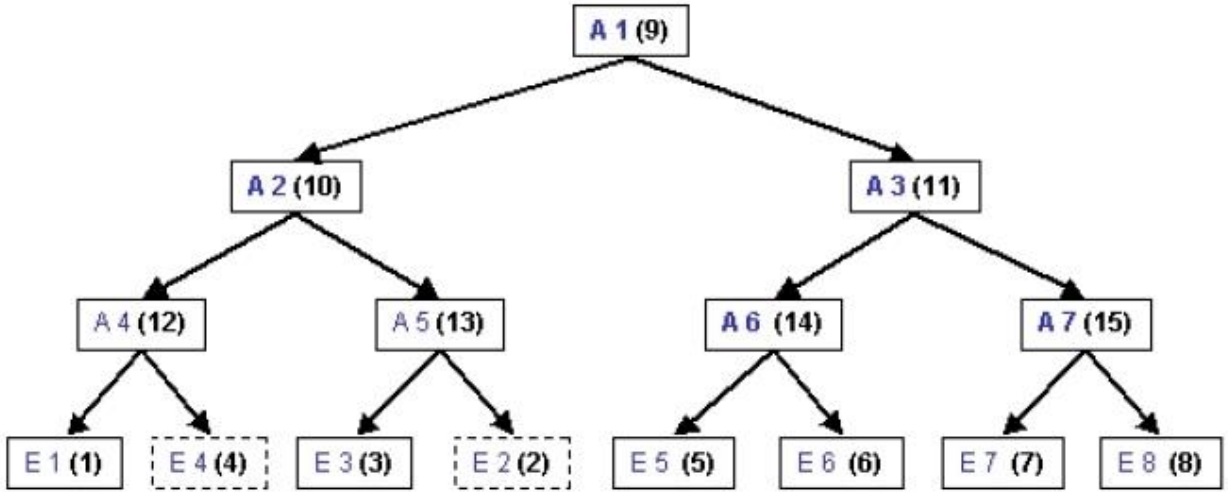


Figura 2.9 Representação gráfica da árvore da Figura 2.7 após aplicação do operador de mutação para troca de colunas à esquerda da raiz, ou seja, troca de espécies. Neste caso foram trocadas as colunas 2 e 4.

Filhos

Pais

	E1 (1)	E2 (2)	E3 (3)	E4 (4)	E5 (5)	E6 (6)	E7 (7)	E8 (8)	A1 (9)	A2 (10)	A3 (11)	A4 (12)	A5 (13)	A6 (14)	A7 (15)
A 1 (9)	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
A 2 (10)	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
A 3 (11)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
A 4 (12)	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
A 5 (13)	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A 6 (14)	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
A 7 (15)	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Figura 2.10 Representação na forma de matriz de adjacências da árvore da Figura 2.9, onde "E n" significa "espécie n" e "A n" significa "Ancestral n".

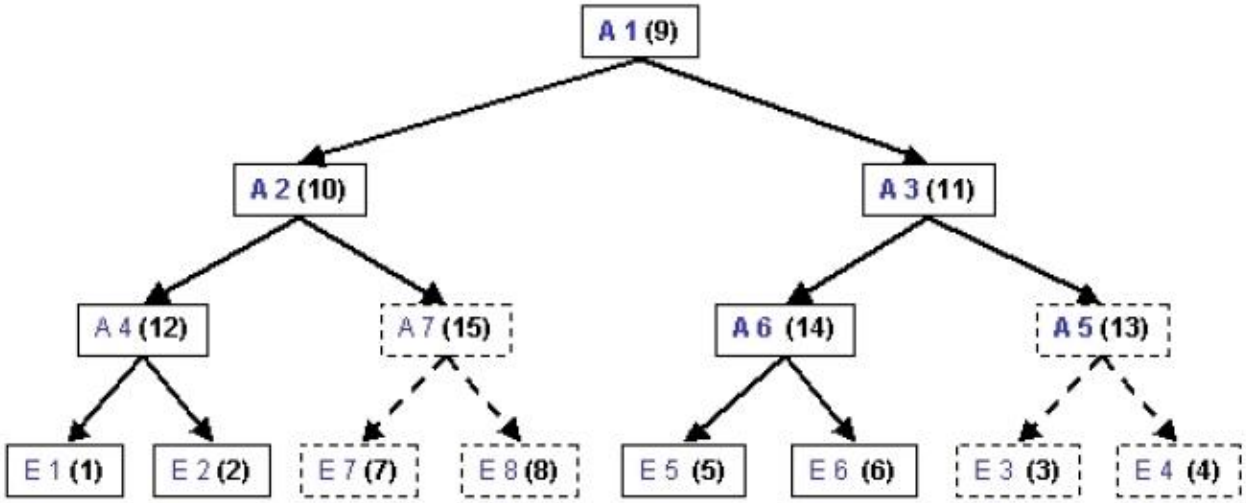


Figura 2.11 Representação gráfica da árvore da Figura 2.7 após aplicação do operador de mutação para troca de colunas à direita da raiz, ou seja, troca de ramos. Neste caso foram trocadas as colunas 13 e 15.

Filhos

Pais

	E1 (1)	E2 (2)	E3 (3)	E4 (4)	E5 (5)	E6 (6)	E7 (7)	E8 (8)	A1 (9)	A2 (10)	A3 (11)	A4 (12)	A5 (13)	A6 (14)	A7 (15)
A1 (9)	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
A2 (10)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
A3 (11)	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
A4 (12)	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A5 (13)	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
A6 (14)	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
A7 (15)	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Figura 2.12 Representação na forma de matriz de adjacências da árvore da Figura 2.11, onde "E n" significa "espécie n" e "A n" significa "Ancestral n".

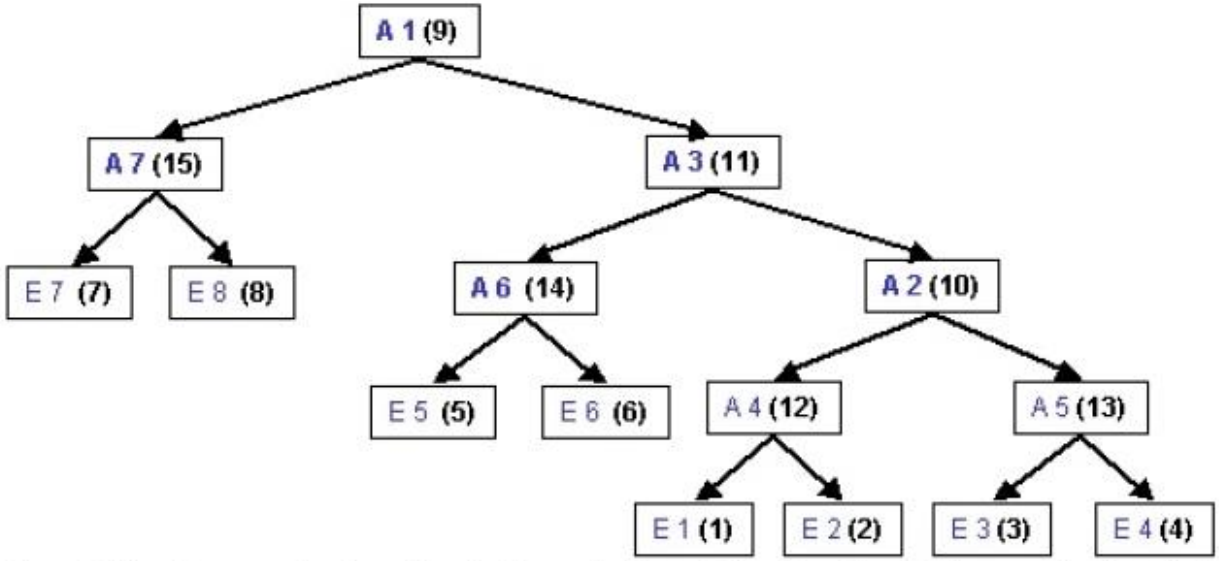


Figura 2.11a Representação gráfica da árvore da Figura 2.7 após aplicação do operador de mutação para troca de colunas à direita da raiz, ou seja, troca de ramos. Neste caso foram trocadas as colunas 10 e 15.

	E1 (1)	E2 (2)	E3 (3)	E4 (4)	E5 (5)	E6 (6)	E7 (7)	E8 (8)	A1 (9)	A2 (10)	A3 (11)	A4 (12)	A5 (13)	A6 (14)	A7 (15)
A1 (9)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
A2 (10)	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
A3 (11)	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
A4 (12)	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A5 (13)	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
A6 (14)	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
A7 (15)	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Figura 2.12a Representação na forma de matriz de adjacências da árvore da Figura 2.11a, onde "E n" significa "espécie n" e "A n" significa "Ancestral n".

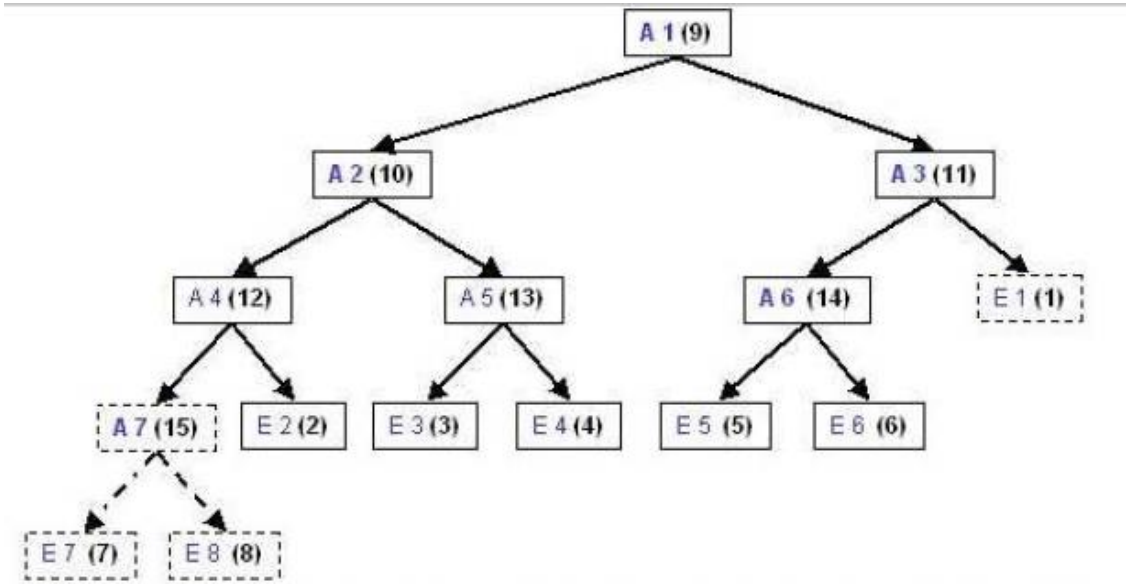


Figura 2.13 Representação gráfica da árvore da Figura 2.7 após aplicação do operador de mutação para troca de uma coluna à direita da raiz por outra à sua esquerda, ou seja, troca de ramo por folha. Neste caso foram trocadas as colunas 1 e 15.

Filhos / Pais

	E1 (1)	E2 (2)	E3 (3)	E4 (4)	E5 (5)	E6 (6)	E7 (7)	E8 (8)	A1 (9)	A2 (10)	A3 (11)	A4 (12)	A5 (13)	A6 (14)	A7 (15)
A1 (9)	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
A2 (10)	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
A3 (11)	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
A4 (12)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
A5 (13)	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
A6 (14)	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
A7 (15)	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Figura 2.14 Representação na forma de matriz de adjacências da árvore da Figura 2.11, onde "E n" significa "espécie n" e "A n" significa "Ancestral n".

20. Dimensão do espaço de busca e custo da busca por evolução

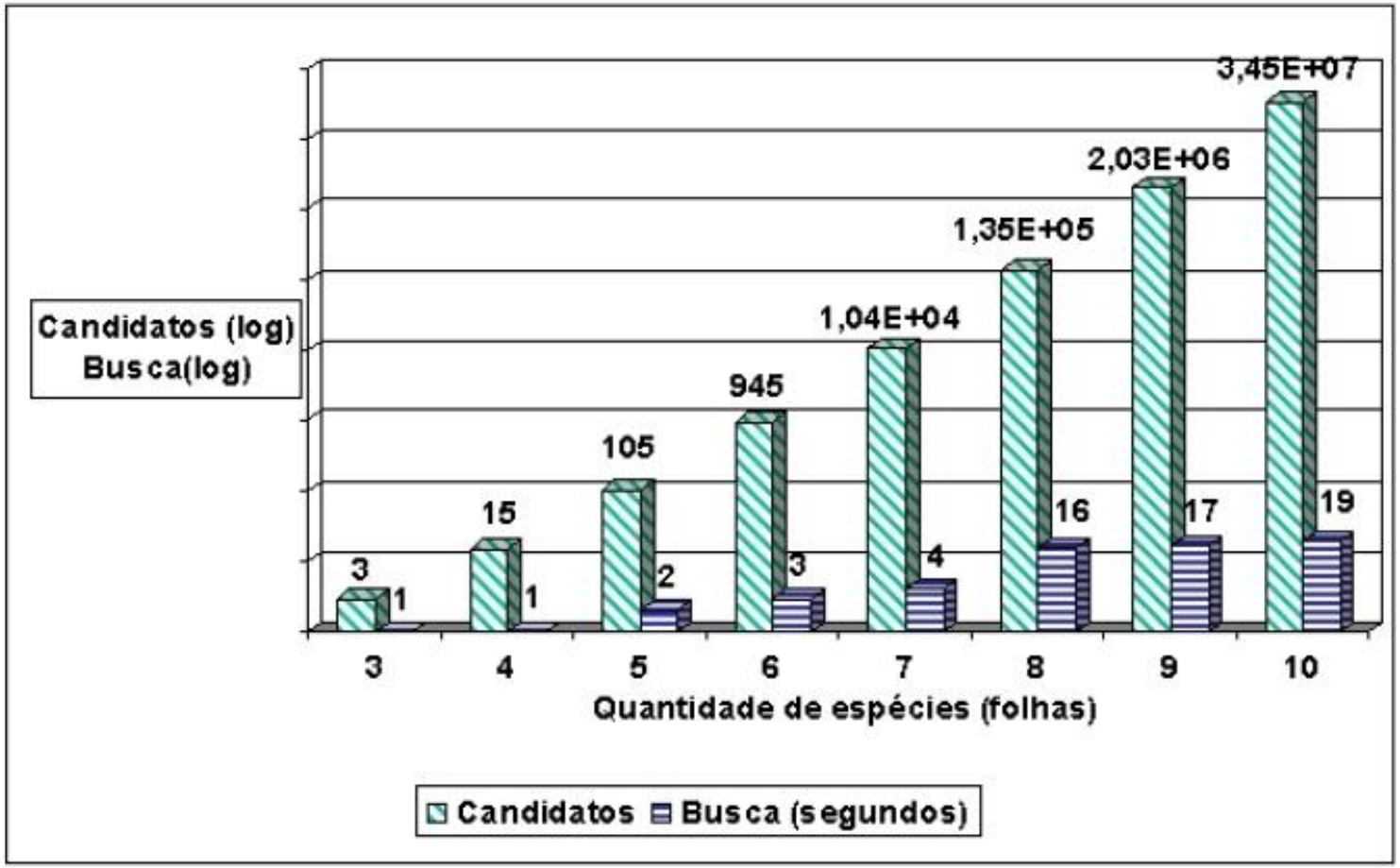


Figura 5.6a Relação entre quantidade de candidatos existentes no espaço de busca e tempo necessário para encontrar uma boa árvore filogenética dentro do intervalo de 1 a 10 seqüências de bases.

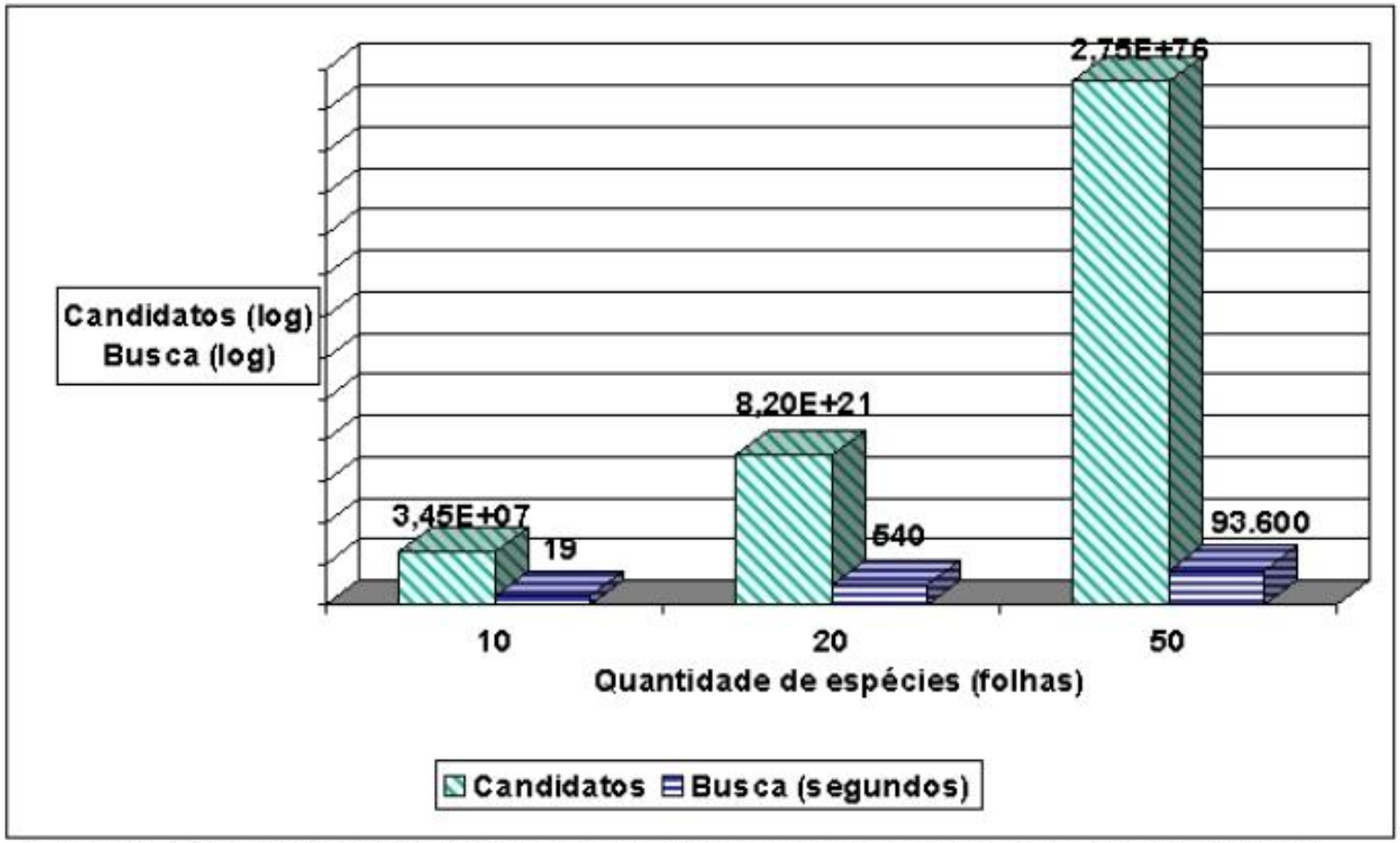


Figura 5.6b Relação entre quantidade de candidatos existentes no espaço de busca e tempo necessário para encontrar uma boa árvore filogenética dentro do intervalo de 10 a 50 seqüências de bases.

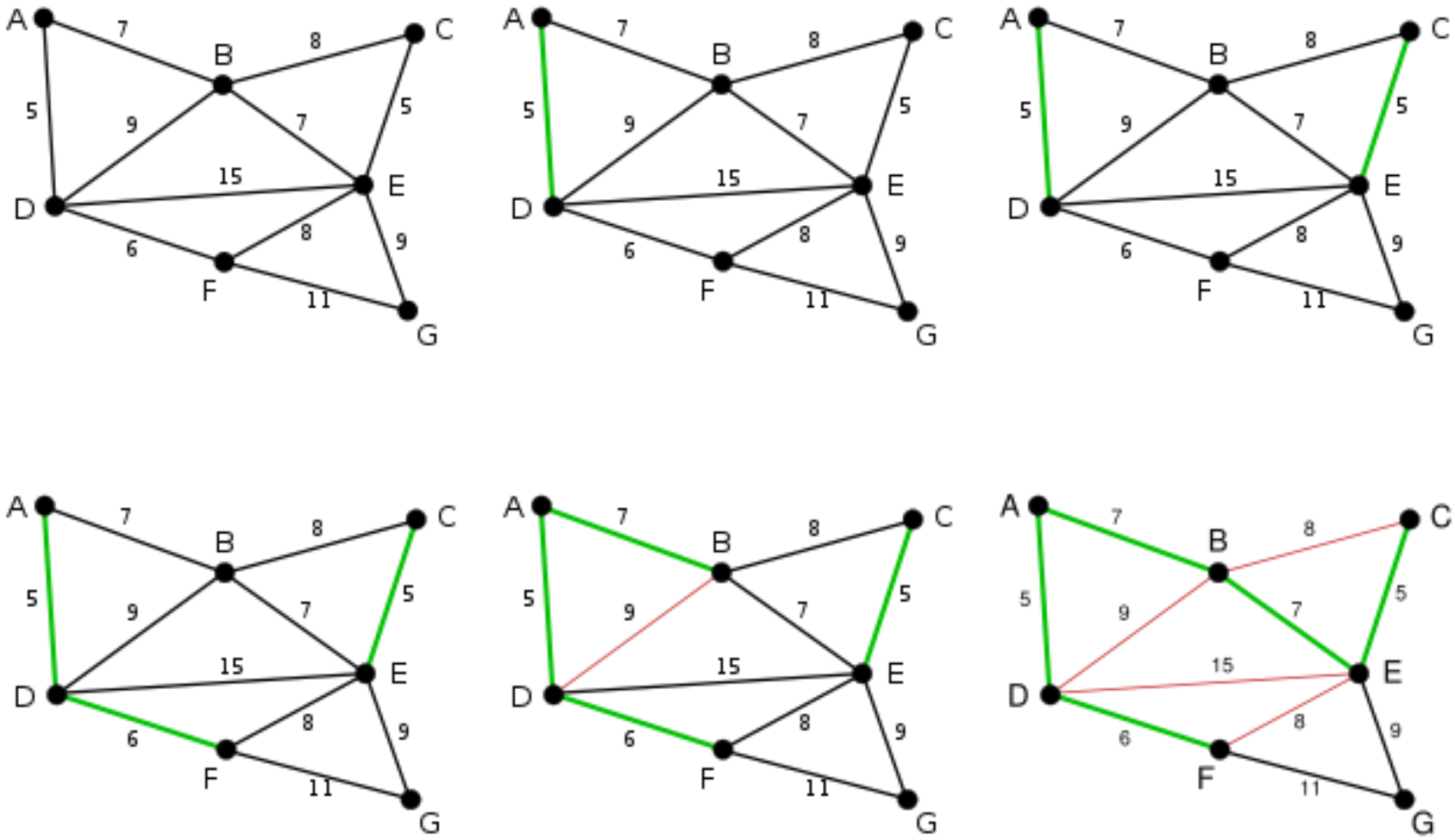
21. Uma proposta que distancia o fenótipo do genótipo

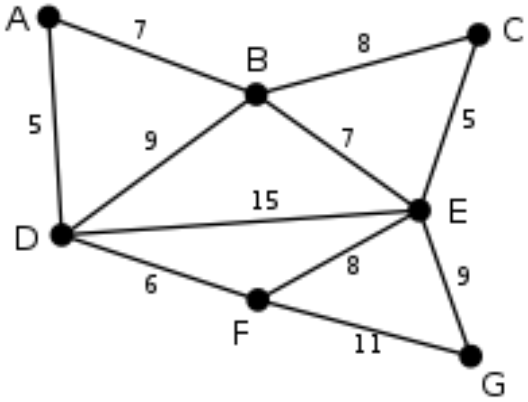
- Em certas circunstâncias, é interessante trabalhar com dois espaços distintos: o espaço genotípico e o espaço fenotípico. Para tanto, é necessário dispor de um mapeamento que associe cada ponto no espaço genotípico a um ponto do espaço fenotípico, e deve-se garantir que todo ponto do espaço fenotípico tenha ao menos uma representação no espaço genotípico.
- Algumas circunstâncias que justificam esta abordagem são as seguintes:
  - ✓ Dificuldade de se realizar a busca diretamente no espaço fenotípico, devido à existência de soluções infactíveis e outras restrições.
  - ✓ Disponibilidade de um mapeamento "barato" computacionalmente e que atenda aos requisitos dispostos acima.
  - ✓ Existência de operadores mais eficientes para realizar a busca no espaço genotípico que no espaço fenotípico. Em muitos casos, inclusive, o espaço genotípico é contínuo e o fenotípico é discreto.

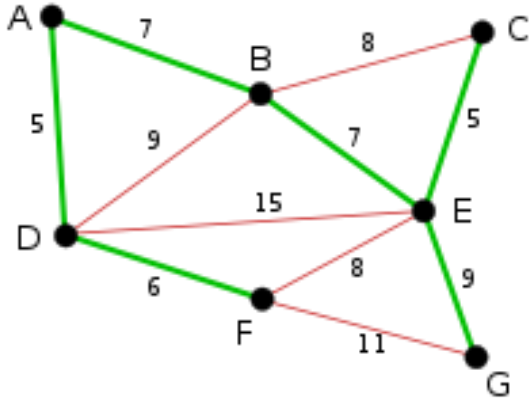
- Um exemplo muito utilizado e de bastante sucesso é a busca de uma árvore que maximiza algum critério de desempenho e que liga um conjunto de nós (vértices).
- Em lugar de buscar no espaço das árvores é possível realizar a busca no espaço de grafos completos (ou ao menos conexos) com arestas ponderadas (descritos por matrizes de adjacências) e empregar algoritmos de *minimum spanning trees*, como Kruskal e Prim, para mapear entre os dois espaços.
- O algoritmo de Kruskal garante obter a árvore geradora mínima (*minimum spanning tree*) (existe um caminho único entre quaisquer dois vértices do grafo e a soma das arestas que compõem a árvore é mínima) para qualquer grafo conexo com arestas ponderadas, e foi proposto em 1956. Caso o grafo não seja conexo, ele produz uma floresta geradora mínima (*minimum spanning forest*), ou seja, haverá uma árvore geradora mínima para cada vértice conectado. Ele é um algoritmo guloso que opera da seguinte forma:

- ✓ Crie uma floresta *F* (conjunto de árvores), onde cada vértice do grafo é uma árvore isolada;
- ✓ Crie um conjunto *S* contendo todos os ramos do grafo;
- ✓ Enquanto *S* não for vazio, faça:
  - Remova o ramo com o menor comprimento de *S* (se houver empate, escolha um deles, aleatoriamente);
  - Se este ramo conecta duas árvores diferentes, então adicione este ramo à floresta *F*, combinando duas árvores em uma única árvore;
  - Se este ramo não conectar duas árvores diferentes, simplesmente descarte-o.

- Quando este algoritmo termina, e sendo o grafo conexo, a floresta *F* terá apenas uma árvore, que é a árvore geradora mínima do grafo. Se o grafo não for conexo, resultará uma floresta.

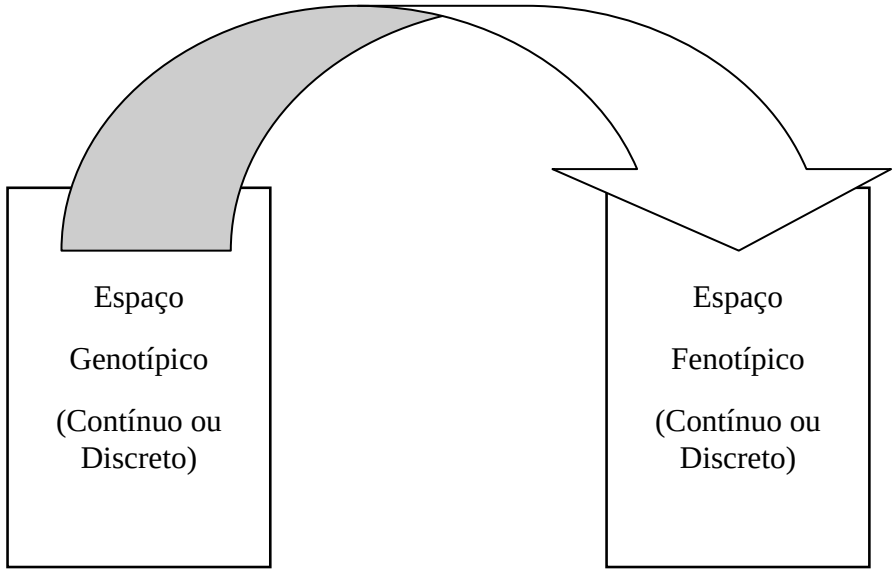

A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11





Mapeamento

Mecanismo de construção da solução



É aqui que se dá a busca, via aplicação dos operadores genéticos

É aqui que os indivíduos são avaliados

22. Cardinalidade de espaços de busca discretos

- A cardinalidade de um conjunto contável de elementos é dada pelo número de elementos do conjunto.

Caso de estudo 1

Problema de bipartição, em que um conjunto de N números deve ser dividido em 2 subconjuntos de modo a produzir 2 somas que, entre si, tenham distância mínima.

Solução: Cada número pode estar em um dos 2 subconjuntos. Cardinalidade: $\boxed{2^N}$.

Caso de estudo 2

Problema de tripartição, em que um conjunto de N números deve ser dividido em 3 subconjuntos de modo a produzir 3 somas que, entre si, tenham distância mínima.

Solução: Cada número pode estar em um dos 3 subconjuntos. Cardinalidade: $\boxed{3^N}$.

Caso de estudo 3

Problema de programação genética restrita a árvores binárias com N folhas, N–1 nós internos, T topologias distintas, S símbolos não-terminais candidatos e P símbolos terminais candidatos.

Solução:

Cada folha pode ter um de P símbolos.

Cada nó interno pode ter um de S símbolos.

Cada topologia tem N folhas e N–1 nós internos.

Cardinalidade: $\boxed{T * S^{N-1} * P^{N}}$.

Caso de estudo 4

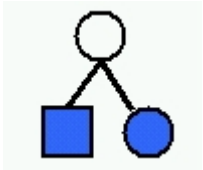
Qual é o número de topologias possíveis para árvores binárias com raiz e N folhas?

Solução:

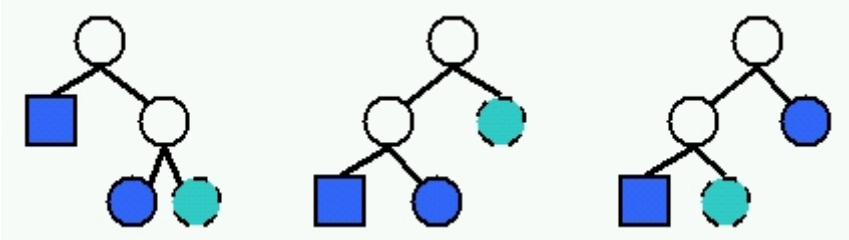
O número de nós internos de árvores binárias com N folhas é $N–1$.

O número total de nós de árvores binárias com N folhas é $2N–1$.

Para 2 nós-folha, existe apenas uma topologia possível:



Para considerar o terceiro nó, este pode se combinar com qualquer um dos três nós já existentes, requerendo sempre a criação de um nó interno adicional:



Para considerar o quarto nó, este pode se combinar com qualquer um dos cinco nós já existentes, e assim por diante, até o N-ésimo nó-folha, o qual poderá se combinar com cada um dos $2(N–1)–1 = 2N–3$ nós. Assim, o número total de topologias possíveis é:

$$1*3*5*\cdots*(2N-5)*(2N-3)=\prod_{i=2}^{N}(2i-3)$$

Multiplicando e dividindo a expressão acima por:

$$\prod_{i=2}^{N-1}(2i-2)=2*4*6*\cdots*(2(N-2)-2)*(2(N-1)-2)$$

resulta:

$$\frac{1*2*3*4*5*6*\cdots*(2N-6)*(2N-5)*(2N-4)*(2N-3)}{2*4*6*\cdots*(2N-6)*(2N-4)}$$

O numerador já pode ser convertido em fatorial. Para o denominador, repare que existem N–2 fatores pares no produto, sendo que é possível dividir todos por 2 e multiplicar por 2^{N-2}, produzindo, finalmente:

$$\boxed{\frac{(2N-3)!}{2^{N-2}(N-2)!}}$$

Caso de estudo 5

Qual é o número de topologias possíveis para árvores binárias sem raiz e *N* folhas?

Solução:

Como a raiz pode se combinar com qualquer nó de uma árvore sem raiz, então pode-se concluir que o número de topologias para árvores binárias sem raiz e *N* folhas é igual ao caso com raiz, mas considerando um nó a menos. Assim resulta:

$$\frac{(2N-5)!}{2^{N-3}(N-3)!}$$

Caso de estudo 6

Busca de uma máquina de estado finito com *N* estados, estado inicial a determinar, *P* símbolos de entrada e *Q* símbolos de saída.

Solução:

Cada estado pode ser o estado inicial.

Para cada estado:

- ✓ Para cada símbolo de entrada, existem N próximos estados possíveis;
- ✓ Para cada próximo estado, existem Q símbolos de saída possíveis.

Cardinalidade: $\boxed{N * \left(N^P * Q^P\right)^N}$.

Solução alternativa:

Cada estado pode ser o estado inicial.

De cada um dos N estados saem P arcos, que podem chegar a qualquer um dos N estados. Logo, o no. de topologias distintas para a máquina de estado finito é N^{P*N}.

O no. de arcos é $P * N$ e cada arco pode ter associado a si um dentre os Q símbolos de saída. Logo, o no. de conjuntos distintos de arcos para uma mesma topologia é Q^{P*N}.

Cardinalidade: $\boxed{N * N^{P*N} * Q^{P*N}}$.

Caso de estudo 7

Busca de uma solução para o problema do caixeiro viajante com *N* cidades. Não importa a cidade inicial e nem o sentido de percurso.

Solução:

Número de permutações das *N* cidades: $N!$.

Como qualquer cidade pode ser a cidade de origem, então existem *N* soluções idênticas para cada percurso possível, o que reduz a cardinalidade do espaço de busca a: $\frac{N!}{N} = (N-1)!$.

Como o sentido de percurso não importa, então ainda existem 2 soluções idênticas para cada percurso possível, o que reduz a cardinalidade do espaço de busca a: $\frac{(N-1)!}{2}$.

23. Probabilidade de ocorrência de eventos

Na inicialização de um cromossomo binário de N bits, sabe-se que cada bit é escolhido por um gerador aleatório de bits que obedece à seguinte regra de probabilidade: p para o bit 1 e $1-p$ para o bit 0. Qual é a probabilidade do cromossomo ter R bits com o valor 1?

Solução:

É evidente que deve-se considerar que $R \leq N$.

Probabilidade do gerador produzir R bits com o valor 1 e $N-R$ bits com o valor 0:

$$(p)^R(1-p)^{N-R}$$

Número de combinações possíveis de ocorrência dos R bits 1 em N posições:

$$\binom{N}{R} = \frac{N!}{R!(N-R)!}$$

Logo, a resposta é dada na forma:

$$\boxed{\binom{N}{R}(p)^R(1-p)^{N-R}}$$

24. Tabela de contagem de coleções

- Tamanho do universo: n
- Tamanho da coleção: k
- Quantas coleções de k elementos existem, sabendo que existem n elementos candidatos?

	Repetição permitida	Repetição proibida
Ordenado	n^k	$(n)_k = \frac{n!}{(n-k)!}$
Não-ordenado	$\left(\!\binom{n}{k}\!\right) = \binom{n+k-1}{k}$	$\binom{n}{k} = \frac{n!}{k!\cdot(n-k)!}$

- A última coluna desta tabela mostra a diferença entre o número de permutações e de combinações de n objetos tomados k de cada vez: $C(n,k) = \frac{P(n,k)}{k!}$.

25. Multiconjuntos

- Dado um conjunto, um elemento pertence ou não a ele. Um mesmo elemento não pode figurar duas vezes em um conjunto.
- Um multiconjunto é uma generalização de um conjunto em que, além de não haver ordem entre seus elementos (como num conjunto), os elementos podem aparecer em multiplicidade.
- Exemplo: Apresente todos os multiconjuntos possíveis de 3 elementos, onde os elementos são escolhidos do conjunto {1, 2, 3}.

$$\langle 1\ \ 1\ \ 1\rangle, \langle 1\ \ 1\ \ 2\rangle, \langle 1\ \ 1\ \ 3\rangle, \langle 1\ \ 2\ \ 2\rangle, \langle 1\ \ 2\ \ 3\rangle$$
$$\langle 1\ \ 3\ \ 3\rangle, \langle 2\ \ 2\ \ 2\rangle, \langle 2\ \ 2\ \ 3\rangle, \langle 3\ \ 3\ \ 2\rangle, \langle 3\ \ 3\ \ 3\rangle$$

- O número de multiconjuntos de k elementos que podem ser formados pelos inteiros de 1 a n é dado por:

$$\left(\!\binom{n}{k}\!\right) = \binom{n+k-1}{k}$$

- Exemplo em algoritmos evolutivos: No processo de seleção por torneio, considerando 4 participantes e 20 indivíduos na população, quantas são as possibilidades de grupos?

$$\left(\!\!\binom{20}{4}\!\!\right) = \binom{20+4-1}{4} = \binom{23}{4} = 8.855$$

25.1 Permutação de multiconjuntos

- As permutações de um multiconjunto são também conhecidas como arranjos com repetição de elementos. Que fique claro que o multiconjunto não tem ordem, mas os seus elementos podem ser dispostos em uma lista (arranjo ordenado de elementos).
- Sem repetição de elementos, o número de arranjos possíveis de n elementos é $n!$.
- Dado um multiconjunto composto por k elementos distintos, sendo $n_1, n_2, \cdots, n_k$ as multiplicidades respectivas desses elementos, então o número de permutações (listas distintas) que se pode obter a partir desses elementos é dada por:

$$\frac{(n_1 + n_2 + \cdots + n_k)!}{(n_1)! \cdot (n_2)! \cdots (n_k)!}$$

- Exemplo: Se um multiconjunto tem 8 elementos ao todo, com 3 repetições de um elemento e 2 repetições de um outro elemento, com os demais elementos não se repetindo, então o número total de arranjos diferentes é dado por:

$$\frac{8!}{3!\cdot 2!} = 3.360$$

- Isso é utilizado para se calcular o número de anagramas de uma palavra, que é dado pelo número de palavras diferentes que podem ser formadas com as letras de uma dada palavra. O exemplo numérico acima pode ser aplicado à palavra PAPAGAIO. É claro que poucas das 3.360 palavras diferentes fazem algum sentido em alguma língua.

26. Número de Stirling de tipo 2

- O número de Stirling de tipo 2 realiza a contagem do número de **diferentes partições de um conjunto** de *p* elementos em *k* conjuntos não-vazios e não-ordenados. Ele é dado pela seguinte fórmula (com versões alternativas):

$$\left\{ {n \atop k} \right\} = \frac{1}{k!}\sum_{j=0}^{k}(-1)^j \binom{k}{j}(k-j)^n = \sum_{j=1}^{k}(-1)^{k-j}\frac{j^{n-1}}{(j-1)!\cdot(k-j)!} = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$$

- Para chegar a esta fórmula, James Stirling (1692-1770) aplicou o princípio da inclusão-exclusão, usado para contar o número de elementos resultante da união de conjuntos que possivelmente apresentam elementos em comum.
- Para dois conjuntos *A* e *B*, sabe-se que:

$$n(A \cup B) = n(A) + n(B) - n(A \cap B)$$

- Para três conjuntos *A*, *B* e *C*, sabe-se que:

$$n(A \cup B \cup C) = n(A) + n(B) + n(C) - n(A \cap B) - n(A \cap C) - n(B \cap C) + n(A \cap B \cap C)$$

27. Uma aproximação para a função fatorial

- James Stirling (1692-1770) provou que:

$$\lim_{n\to\infty} \frac{n!}{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n} = 1$$

- Logo, para valores elevados de *n*, vale a aproximação:

$$n! \cong \sqrt{2\pi n}\left(\frac{n}{e}\right)^n = \sqrt{2\pi n} \cdot n^n \cdot e^{-n}$$

28. Referências bibliográficas

BÄCK, T. "Evolutionary Algorithms in Theory and Practice", Oxford University Press, 1996.

BÄCK, T. "Optimal Mutation Rates in Genetic Search", *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 2-8, 1993.

BÄCK, T., FOGEL, D.B. & MICHALEWICZ, Z. (eds.) "Handbook of Evolutionary Computation", Institute of Physics Publishing and Oxford University Press, 1997.

BÄCK, T., FOGEL, D.B. & MICHALEWICZ, Z. (eds.) "Evolutionary Computation 1: Basic Algorithms and Operators", Institute of Physics Publishing, 2000a.

BÄCK, T., FOGEL, D.B. & MICHALEWICZ, Z. (eds.) "Evolutionary Computation 2: Advanced Algorithms and Operators", Institute of Physics Publishing, 2000b.

BOOKER, L.B., GOLDBERG, D.E. & HOLLAND, J.H. "Classifier Systems and Genetic Algorithms", *Artificial Intelligence*, vol. 40, pp. 235-282, 1989.

BOX, G.E.P. & MULLER, M.E. "A note on the generation of random normal deviates", The Annals of Mathematical Statistics, vol. 29, no. 2, pp. 610-611, 1958.

DARWIN, C. "The Origin of Species", John Murray, 1859 (Penguin Classics, 1985).

DAVIS, L. (ed.) "Handbook of Genetic Algorithms", Van Nostrand Reinhold, 1991.

DE JONG, K.A. "Evolutionary Computation", The MIT Press, 2002.

EIBEN, A.E. & SMITH, J.E. "Introduction to Evolutionary Computing", Natural Computing Series, Springer, 2nd edition, 2015.

FOGARTY, T.C. "Varying the Probability of Mutation in the Genetic Algorithm", *Proceedings of the 3rd International Conference on Genetic Algorithms,* pp. 104-109, 1989.

FOGEL, D.B. "An Introduction to Simulated Evolutionary Computation", *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3-14, 1994.

FOGEL, D.B. (ed.) "Evolutionary Computation: The Fossil Record", The IEEE Press, 1998.

FOGEL, D.B. "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence", 2nd edition, The IEEE Press, 1999.

GASPAR-CUNHA, A., ANTUNES, C.H. & TAKAHASHI, R. (Eds.) "Manual de Computação Evolutiva e Meta-Heurística", Editora da UFMG, 2013.

GOLDBERG, D.E. "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, 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.

KINNEAR, K.E. (ed.) "Advances in Genetic Programming", The MIT Press, 1994.

KOZA, J.R. "Genetic Programming: On the Programming of Computers by means of Natural Selection", The MIT Press, 1992.

KOZA, J.R., BENNET III, F.H., ANDRE, D., KEANE, M.A. & DUNLAP, F. "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming", IEEE Transactions on Evolutionary Computation, vol. 1, no. 2, pp. 109-128, 1997.

LINDEN, R. "Algoritmos Genéticos: Uma importante ferramenta de Inteligência Computacional", Brasport Livros e Multimídia Ltda., 2a. edição, 2008.

LIPSCHUTZ, S. & LIPSON, M. "Discrete Mathematics", 2nd. Edition, Schaum's Outline Series, 1997.

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

MICHALEWICZ, Z. & FOGEL, D. B. "How to solve it: Modern Heuristics", Springer-Verlag, 2000.

MICHALEWICZ, Z. & SCHOENAUER, M. "Evolutionary Algorithms for Constrained Parameter Optimization Problems", *Evolutionary Computation*, vol. 4, no. 1, pp. 1-32, 1996.

MITCHELL, M. "An Introduction to Genetic Algorithms", The MIT Press, 1996.

MÜHLENBEIN, H. "How Genetic Algorithms Really Work: I Mutation and Hillclimbing", Proceedings of the Conference on Parallel Problem Solving from Nature (*PPSN*), vol. 2, pp. 15-25, 1992.

SCHEINERMAN, E.R. "Mathematics: A Discrete Introduction", Brooks Cole, 2nd. Edition, 2005.

SCHWEFEL, H.-P. "Evolution and Optimum Seeking", Sixth-Generation Computer Technology Series, Wiley, 1995.

WRIGHT, A.H. "Genetic Algorithms for Real Parameter Optimization", Foundations of Genetic Algorithms, G. Rawlins (ed.), Morgan Kaufmann, pp. 205-218, 1994.