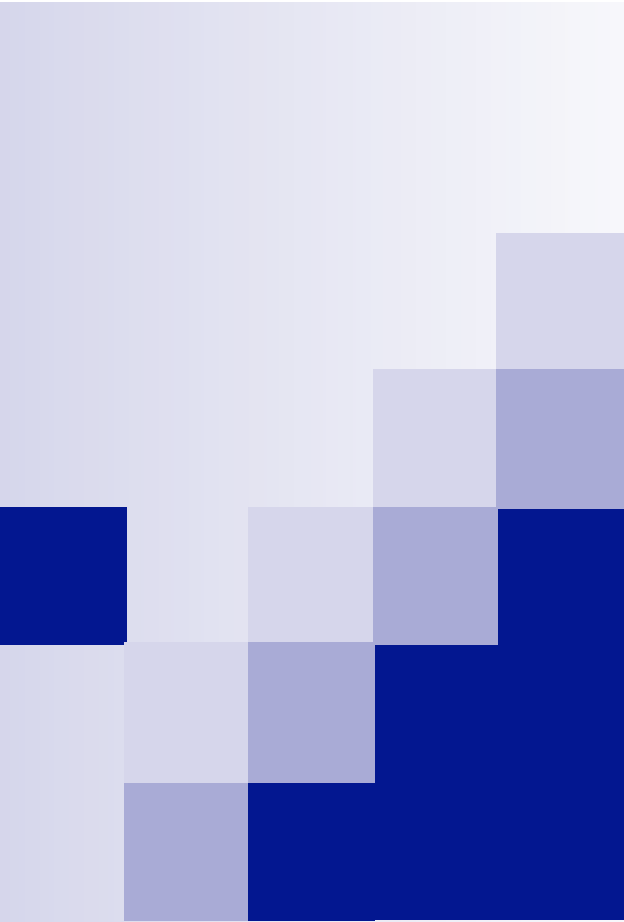
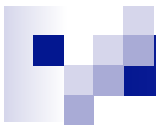




Algoritmos Genéticos

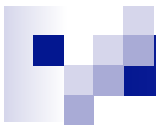
Cap 4.4

Parcialmente adaptado de
<http://aima.eecs.berkeley.edu>



O que são algoritmos genéticos

- Técnicas de optimização estocásticas inspiradas no processo de evolução por selecção natural
- Começam com uma população de indivíduos (ou cromossomas) gerados aleatoriamente
- Fazem evoluir em simultâneo a população (soluções alternativas) por intermédio de operadores de selecção, reprodução e mutação
- Os operadores genéticos permitem manter a diversidade da população e evitam que os AG convirjam prematuramente para um máximo local
- Com uma modelação correcta do problema, a qualidade média das gerações sucessivas será aumentada, em particular a do seu melhor indivíduo

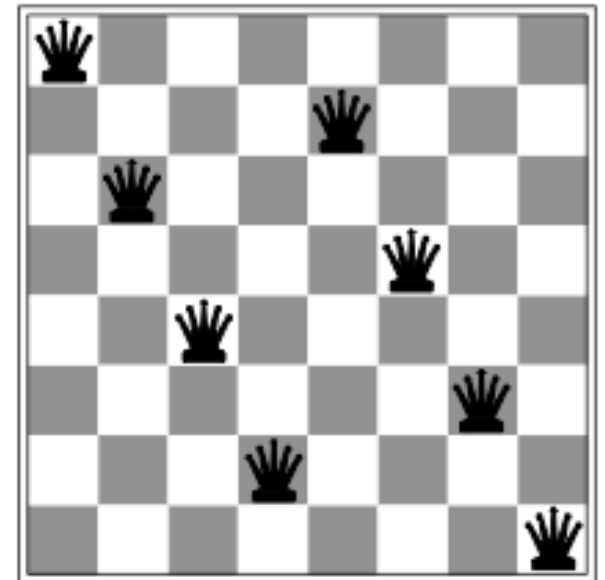


Conceitos básicos de AGs

- A população é constituída por um número geralmente fixo de indivíduos
- Cada cromossoma (ou indivíduo) é representado por uma cadeia de símbolos de um alfabeto finito, normalmente uma cadeia binária de 0s e 1s.
- Os cromossomas são constituídos por genes e os valores que cada gene pode tomar são os alelos.
- A qualidade de cada indivíduo é avaliada por uma função de mérito (*fitness function*) que influencia a probabilidade de selecção dos indivíduos para reprodução
- A reprodução combina dois indivíduos para gerar novo(s) indivíduo(s)
- Os filhos podem sofrer mutações genéticas aleatórias no seu cromossoma antes de serem colocados na população (geração seguinte)
- Uma geração pode ser substituída integralmente pelos seus descendentes ou manter alguns dos seus indivíduos mais aptos.

As rainhas novamente

- Podemos representar um estado por um cromossoma constituído por N-genes, em que cada alelo corresponde à posição ocupado pela rainha respectiva:



Alfabeto	Cromossoma							
{0,1}	111	101	011	001	110	100	010	000
{1,2,3,4,5,6,7,8}	8	6	4	2	7	5	3	1

Funcionamento do AG

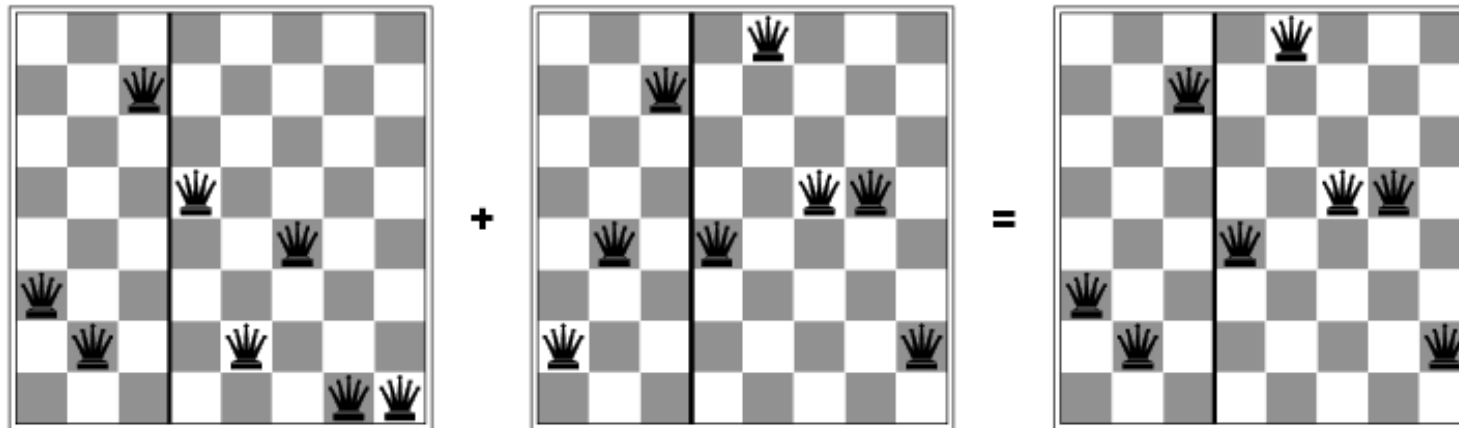


- Função de mérito: número de pares de rainhas que não se atacam (min = 0, max = $8 \times 7/2 = 28$)

- O indivíduo j é seleccionado para reprodução com probabilidade $P(x_j) = \frac{f(x_j)}{\sum_{i=1}^k f(x_i)}$

- Neste exemplo,
 - $24/(24+23+20+11) = 31\%$
 - $23/(24+23+20+11) = 29\%$ etc

Operação de recombinação



O locus (ponto de corte) é escolhido aleatoriamente:

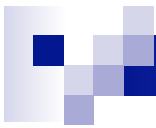
Progenitores	Descendência
32752411	32748552
24748552	24752411

1 ponto de corte

Progenitores	Descendência
32752411	22758451
24748552	34742512

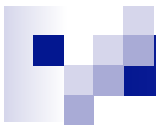
Uniforme com máscara 01110101

Outros métodos permitem a reprodução com N pontos de corte, mas normalmente melhores resultados são obtidos com a recombinação uniforme, em que é criada aleatoriamente uma máscara para copiar o material genético.



Operação de mutação

- Alteração aleatória de um valor dos genes, com uma probabilidade baixa (0.001 valor típico)
- Evita que o algoritmo genético fique preso em máximos locais



Maximização de funções

- Como fazer para encontrar o máximo da função $f(x) = x \cdot \sin(10\pi x) + 1.0$ com precisão de 6 casas decimais no intervalo $[-1..2]$?
- A função de mérito é simplesmente a função que pretendemos maximizar
- Serão necessários 22 bits:
 - $2097152 = 2^{21} < 3\,000\,000 < 2^{22} = 4194304$
- O valor correspondente de x é obtido por simples aritmética a partir do número em decimal representado pelo indivíduo i :
 - $x = -1.0 + \text{decimal}(i) \cdot (1 - (-2)) / (2^{22} - 1) = -1.0 + \text{decimal}(i) \cdot 3 / (4194303)$
- Por exemplo, o cromossoma 100010111011010100011 representa o valor de $x=0,637197$
 - $x = -1.0 + (2\,288\,967) \cdot 3 / (4194303) = 0,637197$
- Podem experimentar o algoritmo com a aplicação disponibilizada em <http://www.obitko.com/tutorials/genetic-algorithms/example-3d-function.php>



Uma implementação de AG

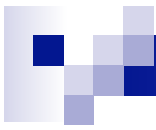
```
function GENETIC-ALGORITHM(problem, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
           FITNESS-FN, a function that measures the fitness of an individual
  repeat
    new_population  $\leftarrow$  empty set
    loop for i from 1 to SIZE(population) do
      x  $\leftarrow$  RANDOM-SELECTION(population, FITNESS-FN)
      y  $\leftarrow$  RANDOM-SELECTION(population, FITNESS-FN)
      child  $\leftarrow$  REPRODUCE(x, y)
      if (random number  $\leq$  mutation probability) then child  $\leftarrow$  MUTATE(child)
      add child to new_population
    population  $\leftarrow$  new_population
  until some individual is fit enough or enough time has elapsed
  return best individual in population, according to FITNESS-FN
```

```
function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals
  n  $\leftarrow$  LENGTH(x)
  c  $\leftarrow$  random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c+1, n))
```



Outra implementação de AG

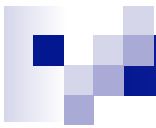
```
function GENETIC-ALGORITHM(problem, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
           FITNESS-FN, a function that measures the fitness of an individual
  repeat
    new_population  $\leftarrow$  empty set
    loop for i from 1 to SIZE(population)/2 do
      x  $\leftarrow$  RANDOM-SELECTION(population, FITNESS-FN)
      y  $\leftarrow$  RANDOM-SELECTION(population, FITNESS-FN)
      if (random number  $\leq$  crossover probability) then
        child1, child2  $\leftarrow$  REPRODUCE(x, y)
      else
        child1, child2  $\leftarrow$  x, y
      endif
      if (random number  $\leq$  mutation probability) then child1  $\leftarrow$  MUTATE(child1)
      if (random number  $\leq$  mutation probability) then child2  $\leftarrow$  MUTATE(child2)
      add child1 and child2 to new_population
    population  $\leftarrow$  new_population
  until some individual is fit enough or enough time has elapsed
  return best individual in population, according to FITNESS-FN
```



Valores empíricos

- Para grandes populações (100 indivíduos)
 - Probabilidade de recombinação 0.6
 - Probabilidade de mutação 0.001

- Para populações pequenas (30 indivíduos)
 - Probabilidade de recombinação 0.9
 - Probabilidade de mutação 0.01



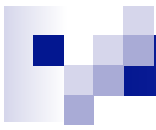
O teorema do esquema

- Um esquema é um padrão da forma $H=(1 * 1 0 * 1 *)$ representando um conjunto de cromossomos (* - don't care)
- O número total de símbolos fixo 0 ou 1 é a ordem do esquema, $O(H)=4$ para o caso anterior
- O comprimento definidor é a distância entre a menor e maior posição definidas no esquema, exemplo $d(H) = 5$.
- **Teorema do Esquema**: o número de instâncias de um determinado esquema cresce exponencialmente ao longo do tempo se estes possuírem três características:
 - ☐ Qualidade acima da média
 - ☐ Baixa ordem
 - ☐ Baixo comprimento definidor
- Os algoritmos genéticos apresentam **paralelismo implícito**: uma população de n indivíduos processa n^3 esquemas.



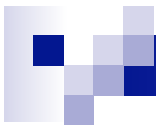
Outros operadores de selecção

- amostragem universal estocástica
 - Mecanismo para obter de uma vez só **todos** os indivíduos para reprodução.
- por posição
 - Ordenam-se os indivíduos por posição, utilizando esta posição para atribuir as probabilidades de selecção.
- por torneio
 - Seja k um parâmetro entre 0 e 1. Seleccionam-se dois indivíduos aleatoriamente da população. Gera-se um número aleatório r . Se $r < k$ fica-se com o melhor indivíduo; caso contrário selecciona-se para reprodução o pior deles.
- estado estável
 - Substitui-se uma percentagem dos piores indivíduos
- por truncatura
 - Só se deixa reproduzir uma percentagem dos melhores indivíduos.
- Elitista
 - Retem-se sempre um pequeno número dos melhores indivíduos.



O caixeiro viajante

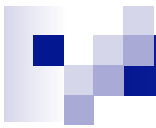
- O problema do caixeiro viajante requer cuidados especiais para a sua resolução com algoritmos genéticos
 - Representação dos cromossomas
 - Operador de recombinação
 - Operador de mutação



Cromossoma para o TSP

- A representação natural
recorre a uma lista
ordenada de cidades para
representar o circuito

1	8	3	4	6	2	7	9	8
---	---	---	---	---	---	---	---	---

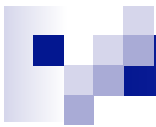


Problemas a resolver

- A dificuldade essencial é o operador de recombinação dado que qualquer dos métodos vistos anteriormente podem criar indivíduos inviáveis (abortos...)
- Duas aproximações possíveis:
 - Reparar os indivíduos
 - Garantir que a recombinação só gera indivíduos válidos

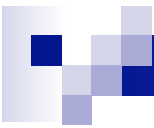
Operador de recombinação PMX

- Partially mapped crossover (PMX)
- Escolhe uma subsequência a partir de um dos pais e preserva a ordem, na medida das possibilidades, do outro progenitor.
- Seleccionam-se dois pontos de corte
 - $p1=1\ 2\ 3\ |\ 4\ 5\ 6\ 7\ |\ 8\ 9$
 - $p2=4\ 5\ 2\ |\ 1\ 8\ 7\ 6\ |\ 9\ 3$
- A selecção induz o conjunto de equivalências $4 \leftrightarrow 1$, $5 \leftrightarrow 8$, $6 \leftrightarrow 7$ e $7 \leftrightarrow 6$.
- Os segmentos entre os pontos são trocados
 - $f1=x\ x\ x\ |\ 1\ 8\ 7\ 6\ |\ x\ x$
 - $f2=x\ x\ x\ |\ 4\ 5\ 6\ 7\ |\ x\ x$
- Copiam-se os restantes valores que não introduzam repetições
 - $f1=x\ 2\ 3\ |\ 1\ 8\ 7\ 6\ |\ x\ 9$
 - $f2=x\ x\ 2\ |\ 4\ 5\ 6\ 7\ |\ 9\ 3$
- Recorre-se às equivalências para substituir os restantes valores
 - $f1=4\ 2\ 3\ |\ 1\ 8\ 7\ 6\ |\ 5\ 9$ (porque $4 \leftrightarrow 1$ e $5 \leftrightarrow 8$)
 - $f2=1\ 8\ 2\ |\ 4\ 5\ 6\ 7\ |\ 9\ 3$ (porque $4 \leftrightarrow 1$ e $5 \leftrightarrow 8$)



Operador de recombinação OX

- Order crossover (OX)
- Selecciona-se uma subsequência e mantém-se a ordem do outro progenitor.
- Seleccionam-se dois pontos de corte
 - p1=1 2 3 | 4 5 6 7 | 8 9
 - p2=4 5 2 | 1 8 7 6 | 9 3
- Copiam-se os segmentos para os descendentes
 - f1=x x x | 4 5 6 7 | x x
 - f2=x x x | 1 8 7 6 | x x
- Copiam-se os valores do outro progenitor sequencialmente a partir do segundo ponto de corte, evitando repetições de cidades
 - f1=2 1 8 | 4 5 6 7 | 9 3
 - f2=3 4 5 | 1 8 7 6 | 9 2



Outras variantes do OX

- Seleccionam-se dois pontos de corte
 - p1=1 2 3 | 4 5 6 7 | 8 9
 - p2=4 5 2 | 1 8 7 6 | 9 3
- Copiam-se os segmentos para os descendentes
 - f1=x x x | 4 5 6 7 | x x
 - f2=x x x | 1 8 7 6 | x x
- Em vez de se começar a partir do segundo ponto de corte começa-se pela ordem do outro progenitor desde o início
 - f1=2 1 8 | 4 5 6 7 | 9 3
 - f2=2 3 4 | 1 8 7 6 | 5 9
- Em vez de utilizar uma subsequência, pode-se gerar uma máscara aleatória para seleccionar as cidades a manter para a geração seguinte.



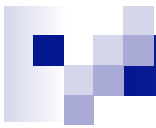
Operador de recombinação CX

- Cycle crossover (CX) – não necessita de pontos de corte
 - p1=1 2 3 4 5 6 7 8 9
 - p2=4 1 2 8 7 6 9 3 5
- Começa-se na primeira cidade de p1
 - f1=1 x x x x x x x x
- A respectiva em p2 é a 4, colocando-se essa cidade na posição de p1
 - f1=1 x x 4 x x x x x
- A respectiva em p2 é a 8, colocando-se essa cidade na posição de p1
 - f1=1 x x 4 x x x 8 x
- Itera-se até introduzir um ciclo (cidade já colocada)
 - f1=1 2 3 4 x x x 8 x
- Preenchem-se as restantes a partir da sequência no outro progenitor:
 - f1=1 2 3 4 7 6 9 8 5



Operadores de mutação

- Inversão (inverte subsequência)
 - 1 2 | 3 4 5 6 | 7 8 9
 - 1 2 | 6 5 4 3 | 7 8 9
- Inserção (move arbitrariamente 1 cidade)
 - 1 2 3 4 5 6 7 8 9
 - 1 2 7 3 4 5 6 8 9
- Deslocamento (move subsequência)
 - 1 2 3 4 5 | 6 7 8 | 9
 - 1 | 6 7 8 | 2 3 4 5 9
- Troca (troca duas cidades)
 - 1 2 3 4 5 6 7 8 9
 - 7 2 3 4 5 6 1 8 9



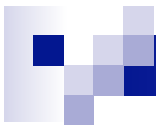
Algoritmos meméticos

- Após aplicação de qualquer um dos operadores genéticos e mesmo da inicialização da população, efectua-se um procura local
- Estes algoritmos são designados por algoritmos meméticos
- Por exemplo, pode-se aplicar o algoritmo trepa-colinas para melhorar o indivíduo
 - Considerando todas as trocas de duas ou três cidades e escolhendo a melhor alternativa até atingir um mínimo local (2-opt e 3-opt)
- É preciso ter cuidado na escolha do operador de mutação para não efectuar as mesmas operações que a procura local.
- Uma possibilidade é não efectuar procura local após mutação.



Outros algoritmos de inspiração evolucionária

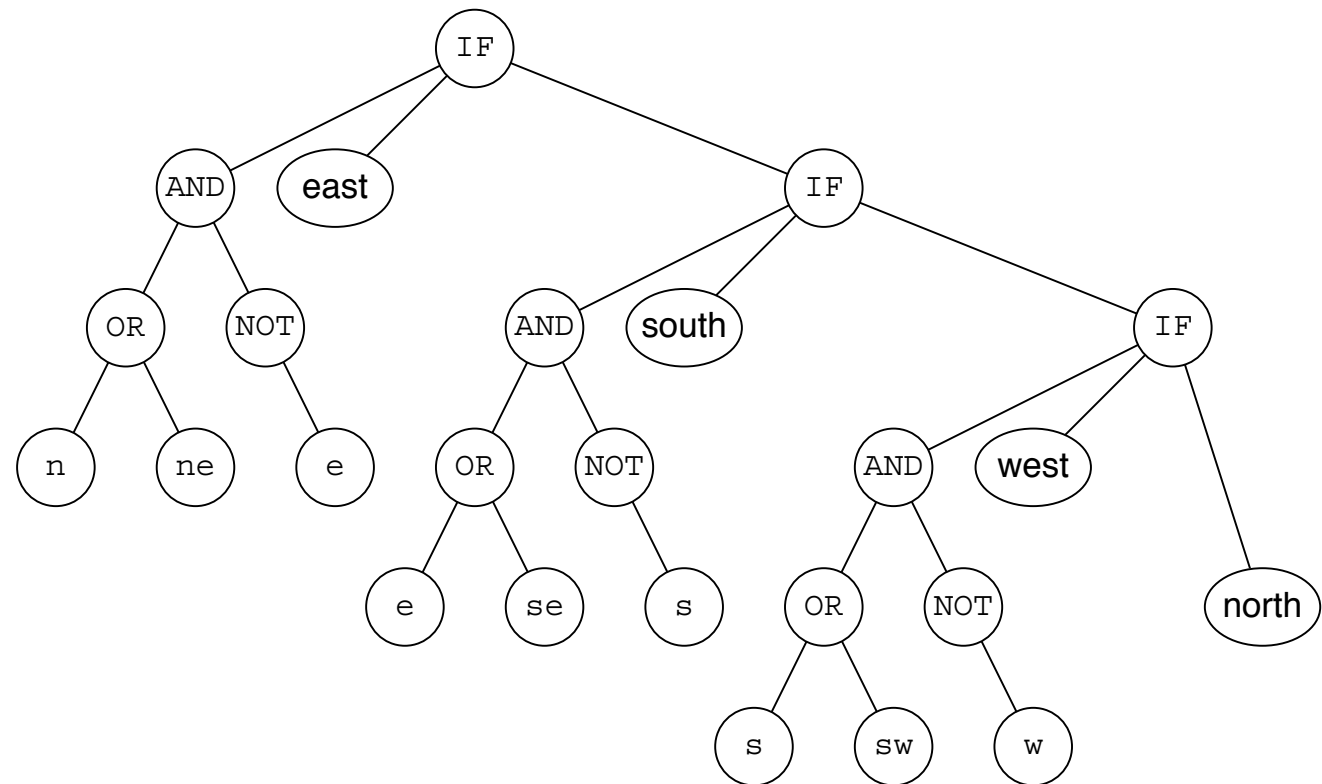
- Programação Genética
 - Evolução de programas
- Estratégias Evolutivas
 - Evolução de vectores reais com distribuições normais para resolução de problemas de Engenharia
 - O operador dominante é a mutação
- Programação Evolutiva
 - Os indivíduos são máquinas de estados



Programação Genética

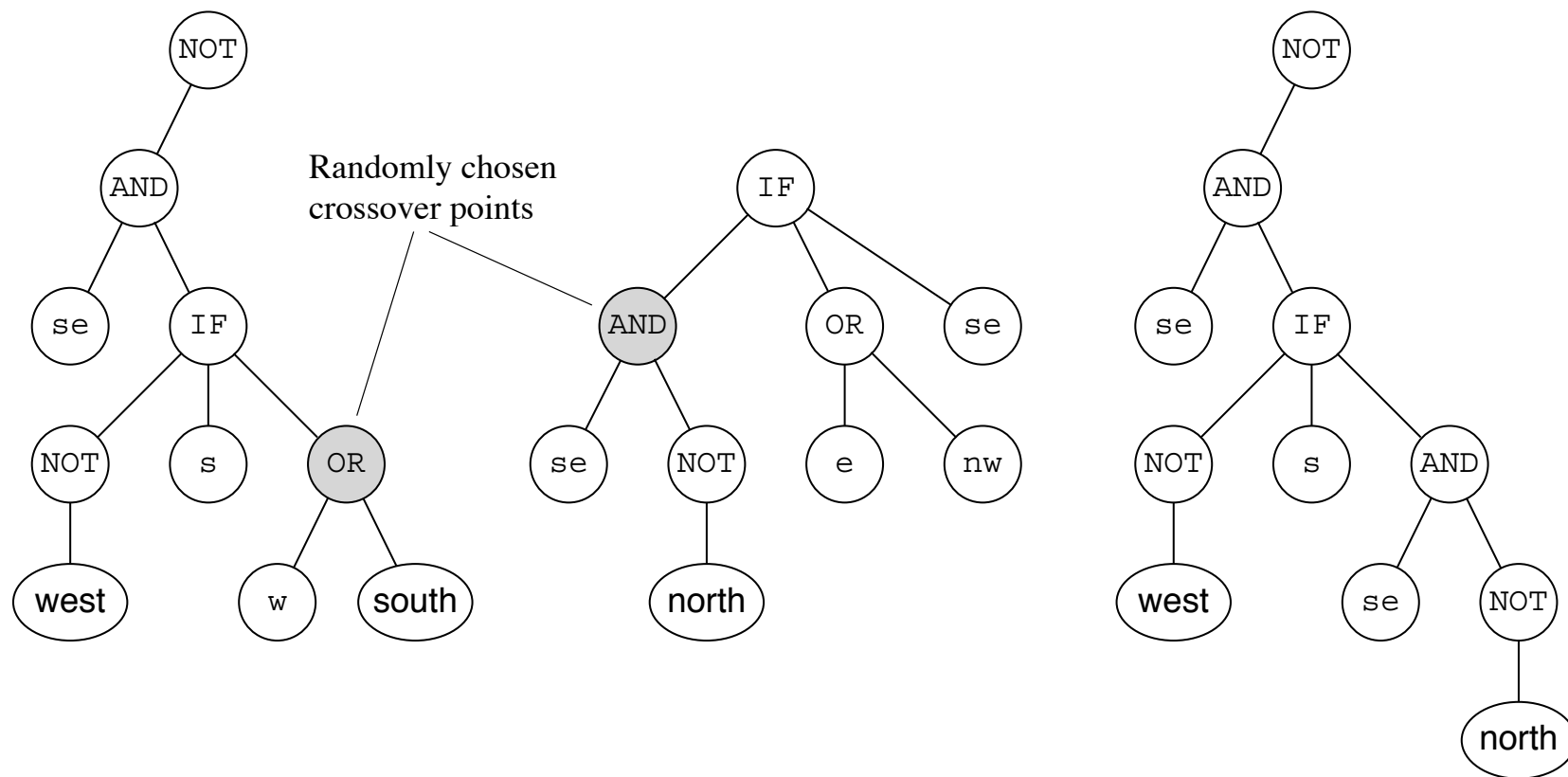
- Cada indivíduo é um programa. A qualidade de um indivíduo é obtida por execução do programa relativamente a um conjunto de testes.
- O material genético encontra-se estaturado em árvore
- O material genético é de tamanho variável durante o processo evolutivo
- O operador de recombinação troca subárvores entre os dois progenitores, preservando a sintaxe
- O operador de mutação substitui uma subárvore por outra construída aleatoriamente
- A população inicial é obtida pela geração aleatória de funções e terminais que constituem os programas.

Evolução de seguidores de paredes [Nilsson]



```
(IF (AND (OR (n) (ne)) (NOT (e)))
    (east)
    (IF (AND (OR (e) (se)) (NOT (s)))
        (south)
        (IF (AND (OR (s) (sw)) (NOT (w)))
            (west)
            (north))))))
```

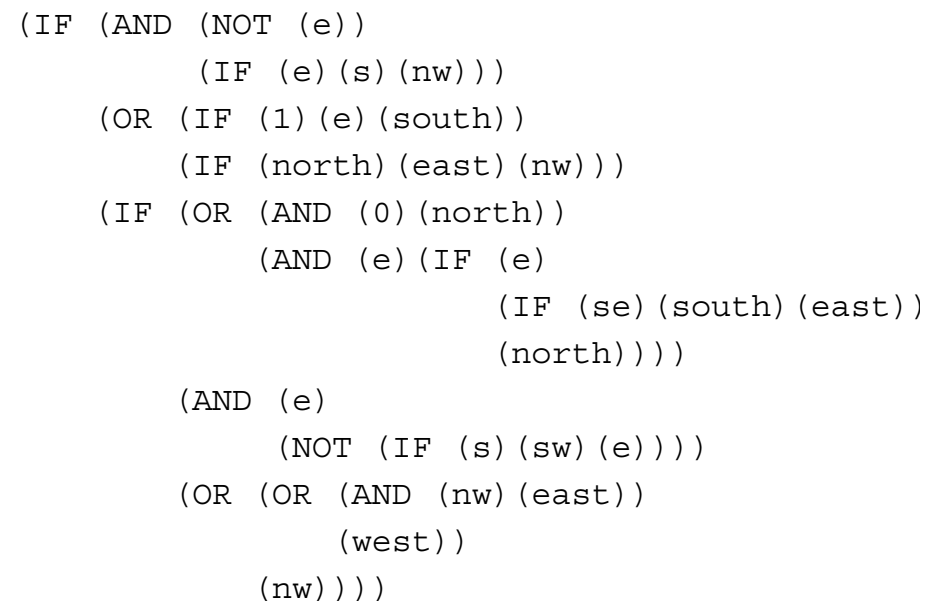
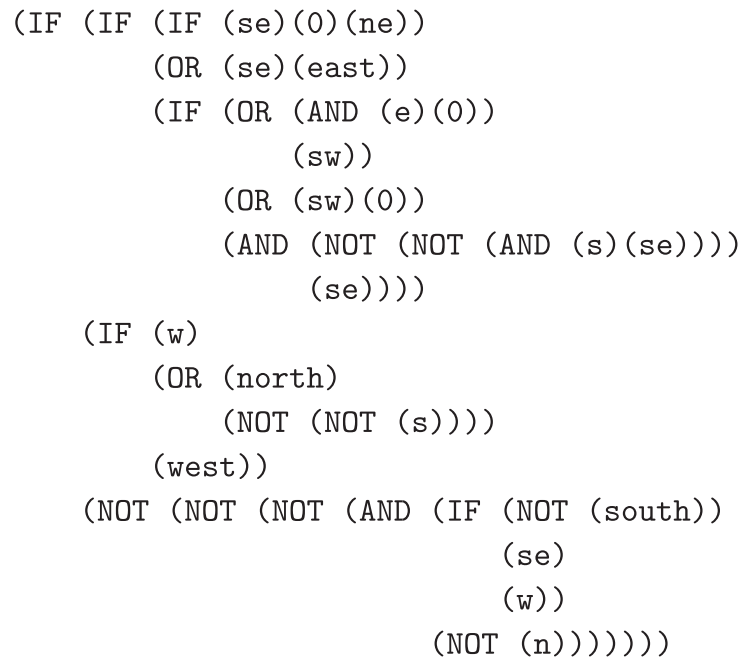
Recombinação de programas

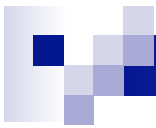


Mother program

Father program

Child program





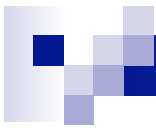
Conclusões

- Grande impacto em problemas de otimização, nomeadamente problemas de escalonamento, desenho de circuitos integrados, classificação, simulação e controlo.
- Algoritmos que suscitam muito interesse, provavelmente por causa das suas origens na teoria da evolução.
- Falta estudar condições que identifiquem claramente as situações em que os algoritmos genéticos têm bom comportamento.



Resultados recientes ([Christos Papadimitriou](#))

- A [mixability theory of sex in evolution](#), with Adi Livnat, Jonathan Dushoff, and Marcus Feldman, PNAS, December 2008. There had been no satisfactory explanation of the advantages of sex in evolution, and yet sex is almost ubiquitous among species despite its huge costs. Here we propose a novel explanation: Using standard models, we establish that, rather astonishingly, evolution of sexual species does not result in maximization of fitness, but in improvement of another important measure which we call mixability: The ability of a genetic variant to function adequately in the presence of a wide variety of genetic partners.
- In a [follow-up article](#) we also point out that mixability lies at the root of modularity and the genetic hierarchy (without it there are no genes, families of genes, etc.)



Bibliografia

- Costa & Simões, capítulo de agentes adaptativos.
- Secção 4.1.4 (3ª edição) ou 4.3 (2ª edição) do AIMA.
- Poderão obter uma boa panorâmica em <http://www.obitko.com/tutorials/genetic-algorithms/>