Programação Genética

Computação Natural

Gisele L. Pappa

Programação Genética

- Criada com o objetivo de evoluir programas

- Lista de 36 (re-)invenções que são competitivas com as soluções propostas por humanos
  - 2 geraram patentes

www.genetic-programming.org

http://www.genetic-programming.org/hc2005/main.html

Programação Genética

- Principais características
  - Um indivíduo é uma solução candidata contendo funções e terminais, e não apenas “dados” (variáveis / constantes)
  - Normalmente indivíduos tem tamanhos e formas variadas
  - Na teoria, um indivíduo é uma “receita” para resolver um dado problema, ao invés de uma solução para uma dada instância de um problema

GA vs. GP

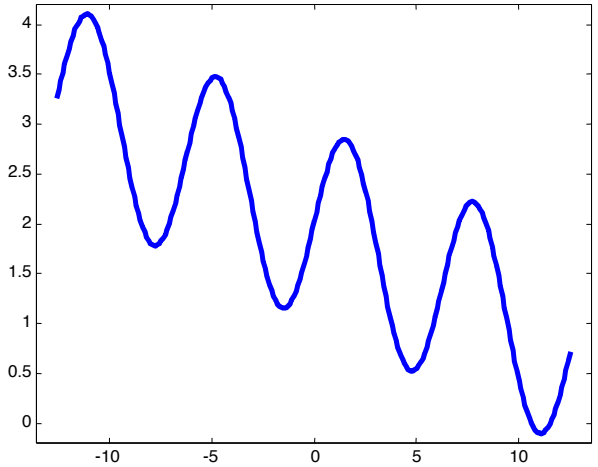
- Ex: otimização de funções vs. aproximação de funções
  - Dada uma função complexa, por exemplo ***sin*(*x*) - 0.1 *x* + 2,** podemos usar um GA para encontrar o valor ótimo da função
  - Dado um conjunto de dados, contento pares <x, f(x)>, podemos utilizar GP para encontrar uma função g(x) que se aproxime da função desconhecida f(x) (regressão simbólica)
- Considere o problema do caxeiro viajante
  - Em um GA um indivíduo é uma sequencia de cidades
  - Em GP um algoritmo deveria ser um "algoritmo" para resolver qualquer instância do problema, dado qualquer conjunto de cidades

Otimização versus Aproximação de funções

Exemplo de aplicação de um GA:

Encontre os máximos ou mínimos da função

f(x) = $sin(x) - 0.1\,x + 2$

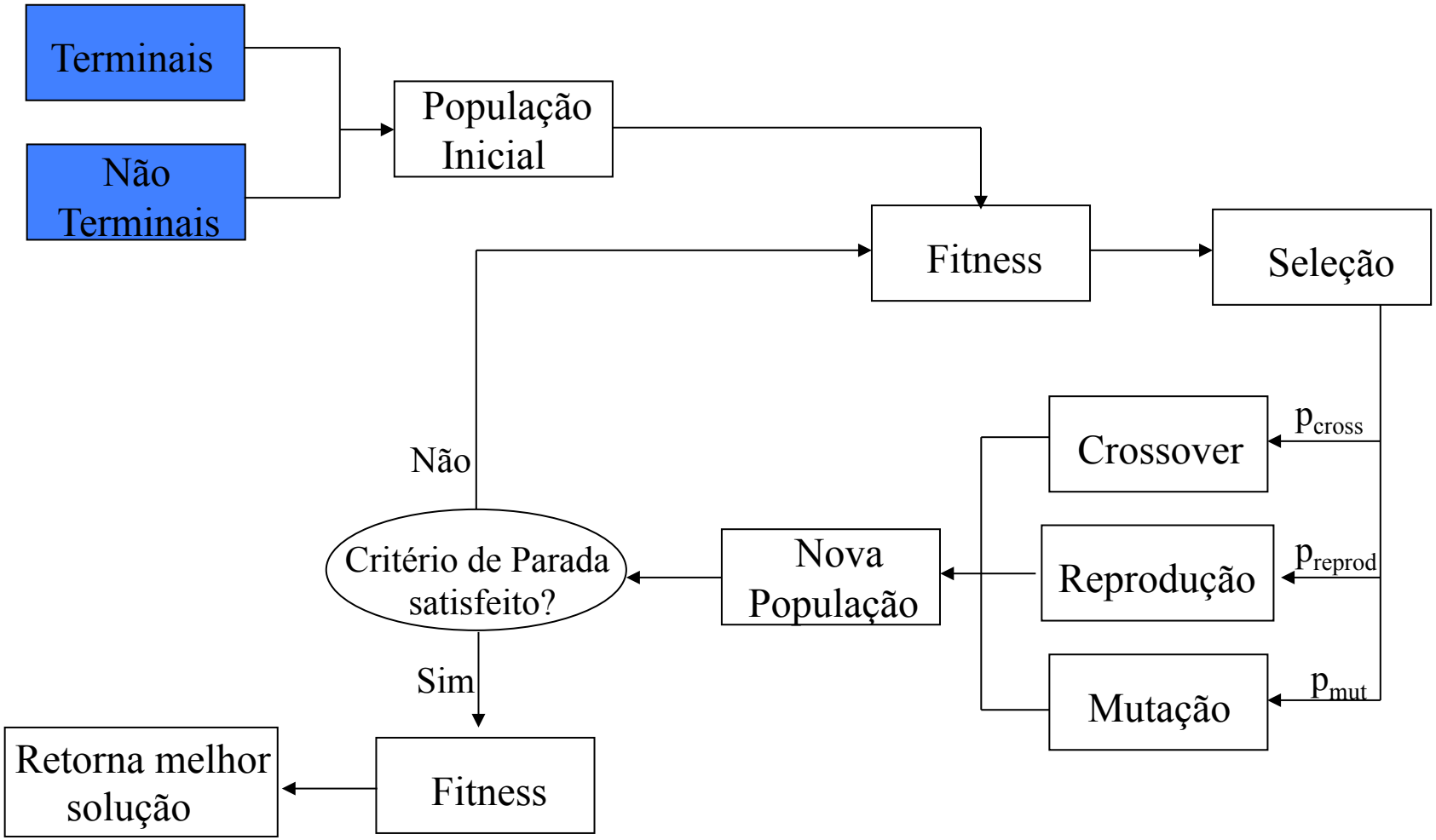


Exemplo de aplicação de um GP:

Encontre a função (programa) que produz os pontos dados na tabela a seguir:

x	saída
-10	3.6
-8	2.1
-6	2.7
-4	2.6
.	.
.	.

Programação Genética



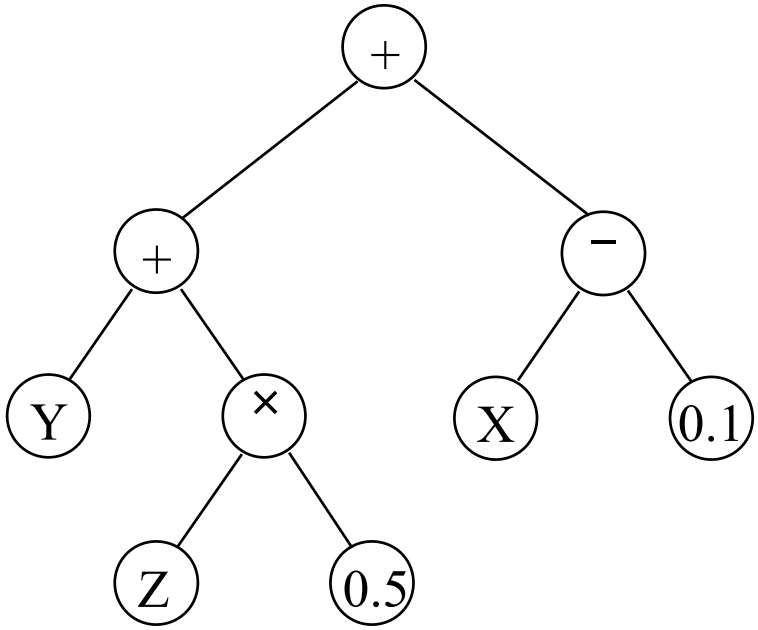
Programação Genética (GP)

- 2 componentes básicos:
  - Conjunto de terminais – variáveis e constantes
  - Conjunto de funções – funções apropriadas para resolver o problema em questão

- Tipos de representação:
  - Representação linear
  - Representação por árvores (mais comum)
  - Representação por grafos

Componentes básicos de um GP

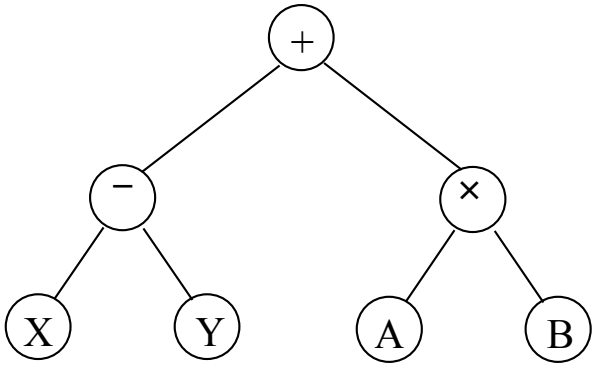
- Focaremos na representação por árvore
  - Nós internos: funções ou operadores
  - Nós folhas: variáveis ou constantes



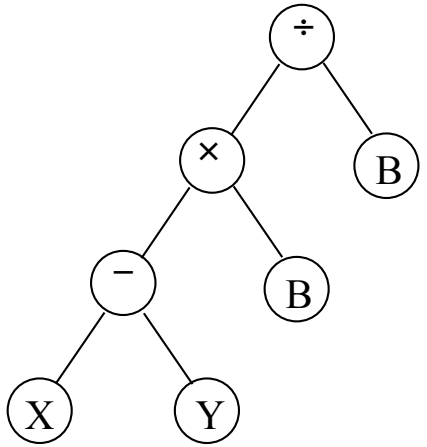
Exemplos de Indivíduos

– Conjunto de funções: $F = \{+, -, \times, \div\}$

– Conjunto de terminais: $T = \{A, B, X, Y\}$



Indivíduo 1: (+ (– X Y) (× A B))



Indivíduo 2: (÷ (× (– X Y) B) B)

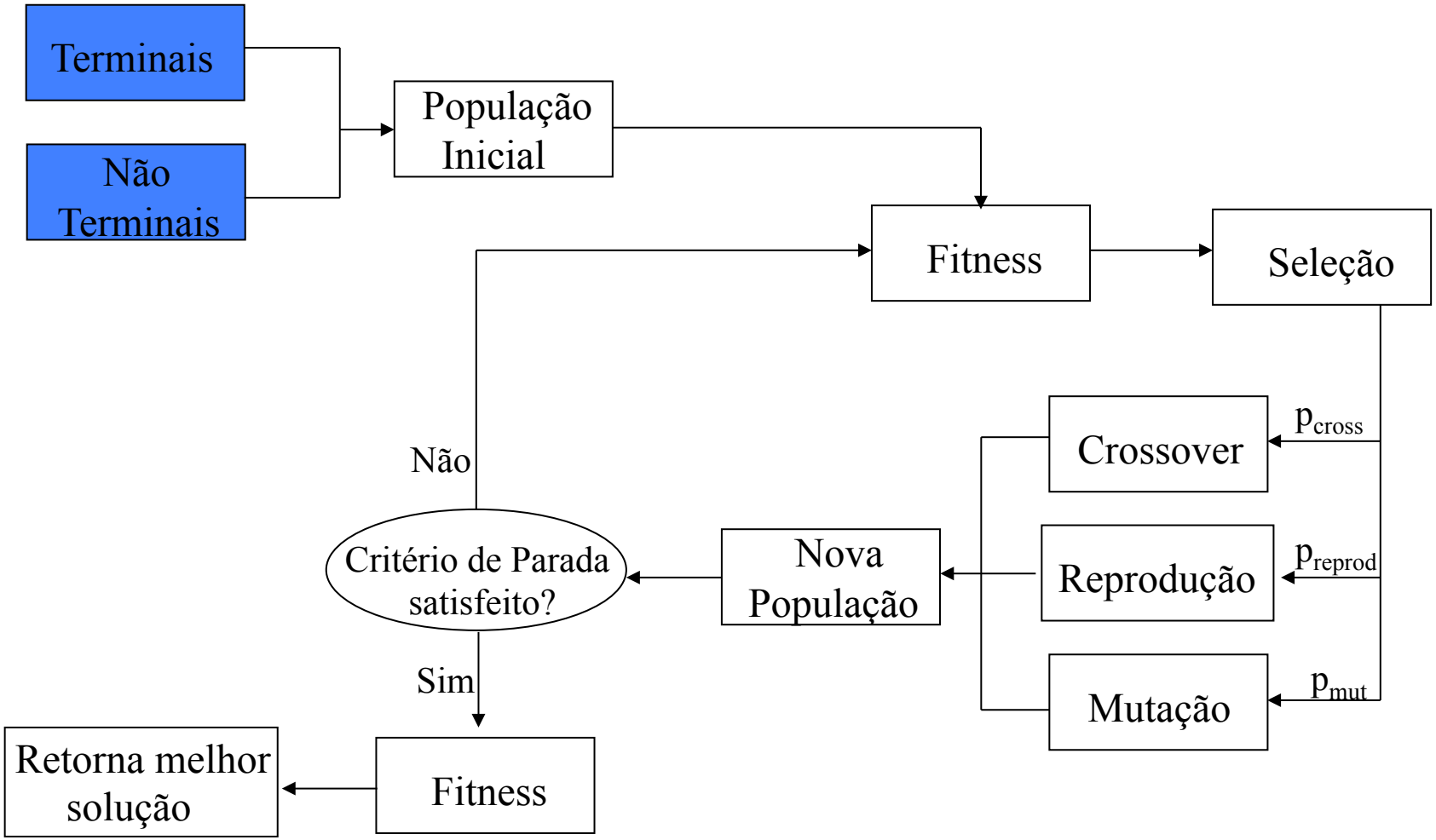
Inicialização da População

- 3 métodos principais
  - *Grow*
    - O nó de uma árvore é escolhido considerando elementos em ambos os conjuntos de terminais e funções, considerando uma altura máxima
    - Produz árvores com formas irregulares
  - *Full*
    - O nó de uma árvore é escolhido considerando elementos apenas do conjuntos de funções, até que a profundidade máxima seja alcançada.
    - Nesse momento, nós passam a ser escolhidos do conjunto de terminais
    - Produz árvores balanceadas

Inicialização da População

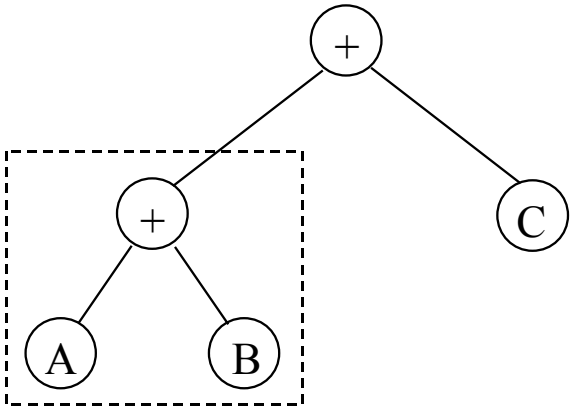
– *Ramped half-and-half*
  - Combina os métodos *full* e *grow* para aumentar diversidade
  - Divide a população em subconjuntos com o mesmo número de indivíduos, e inicializa metade dos indivíduos de cada conjunto com o método *grow* e metade com o método *full*
  - Se a profundidade máxima da árvore é 6, e o tamanho da população 50, serão criados um mesmo número de indivíduos com profundidades 2, 3, 4, 5 e 6 (nesse caso, 10 indivíduos). 5 deles serão inicializados utilizando *grow* e 5 *full*.

Programação Genética

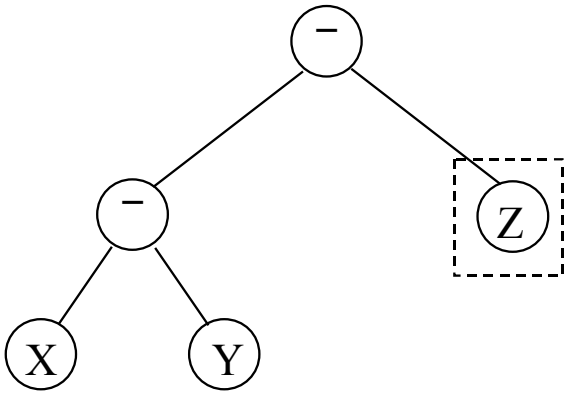


Operadores Genéticos

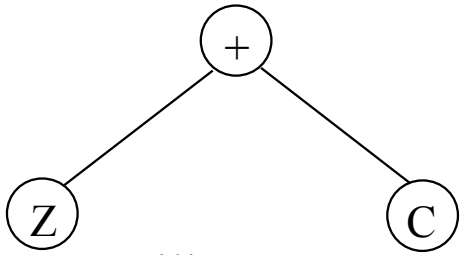
- Crossover: troca sub-árvores



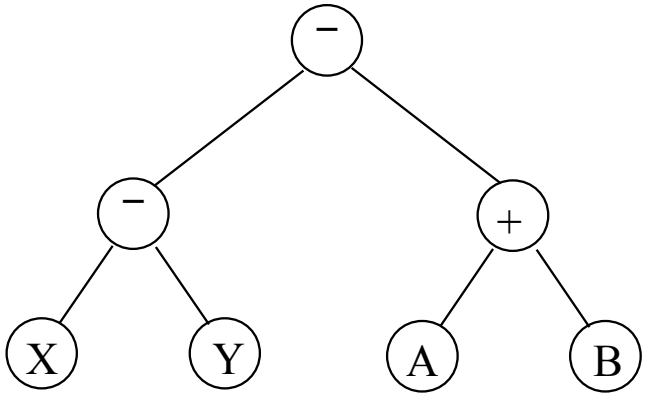
Pai 1



Pai 2



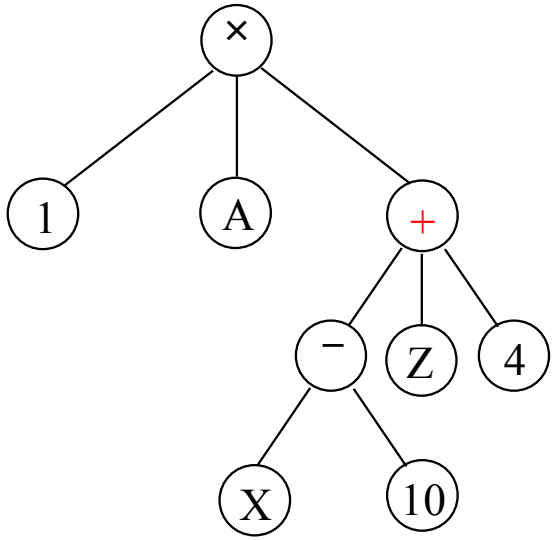
Filho 1



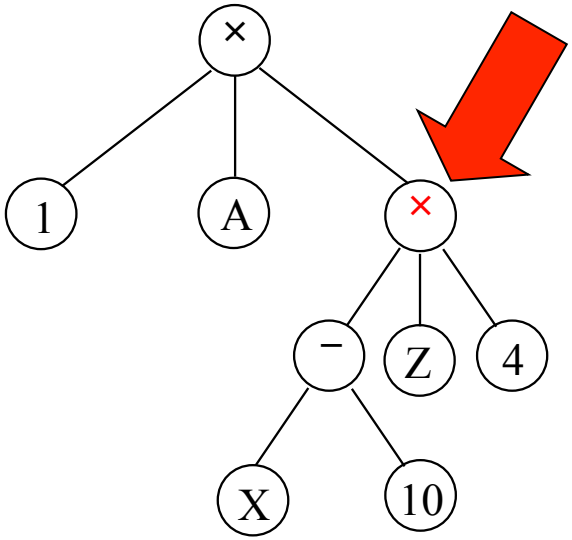
Filho 2

Operadores Genéticos

- Mutação de um ponto



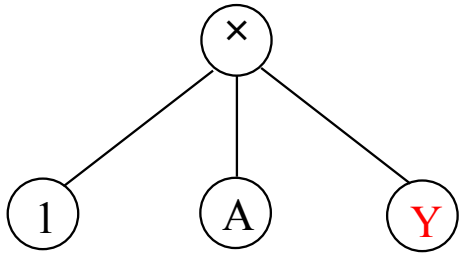
Pai



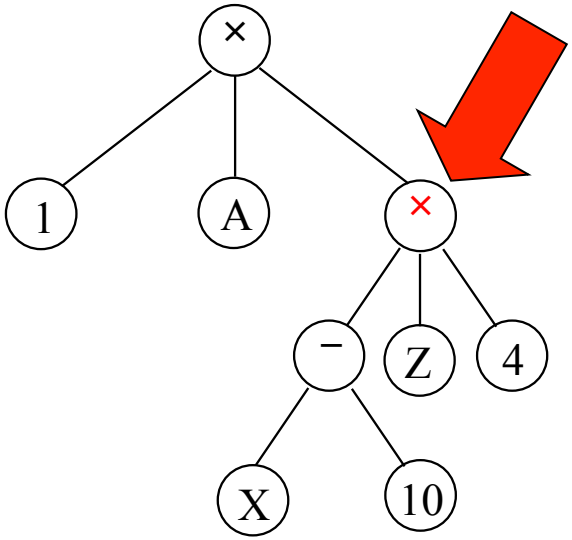
Filho

Operadores Genéticos

- Mutação de expansão



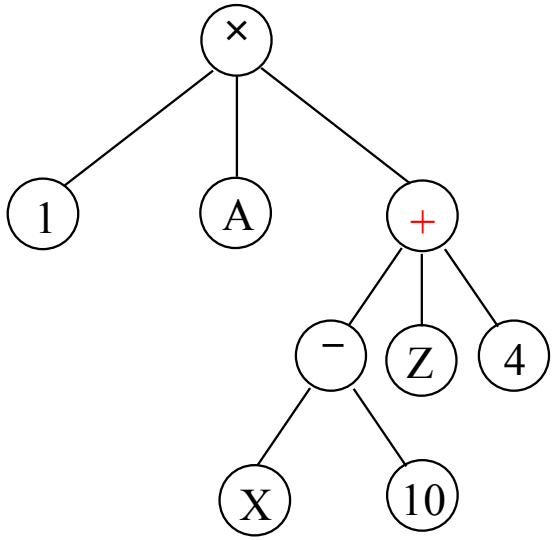
Pai



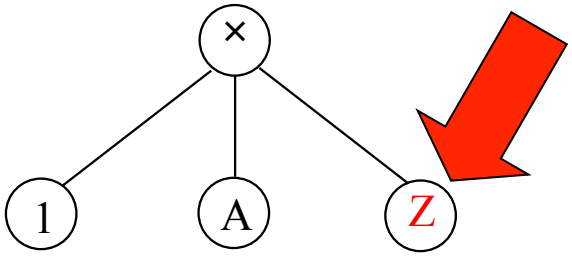
Filho

Operadores Genéticos

- Mutação de redução

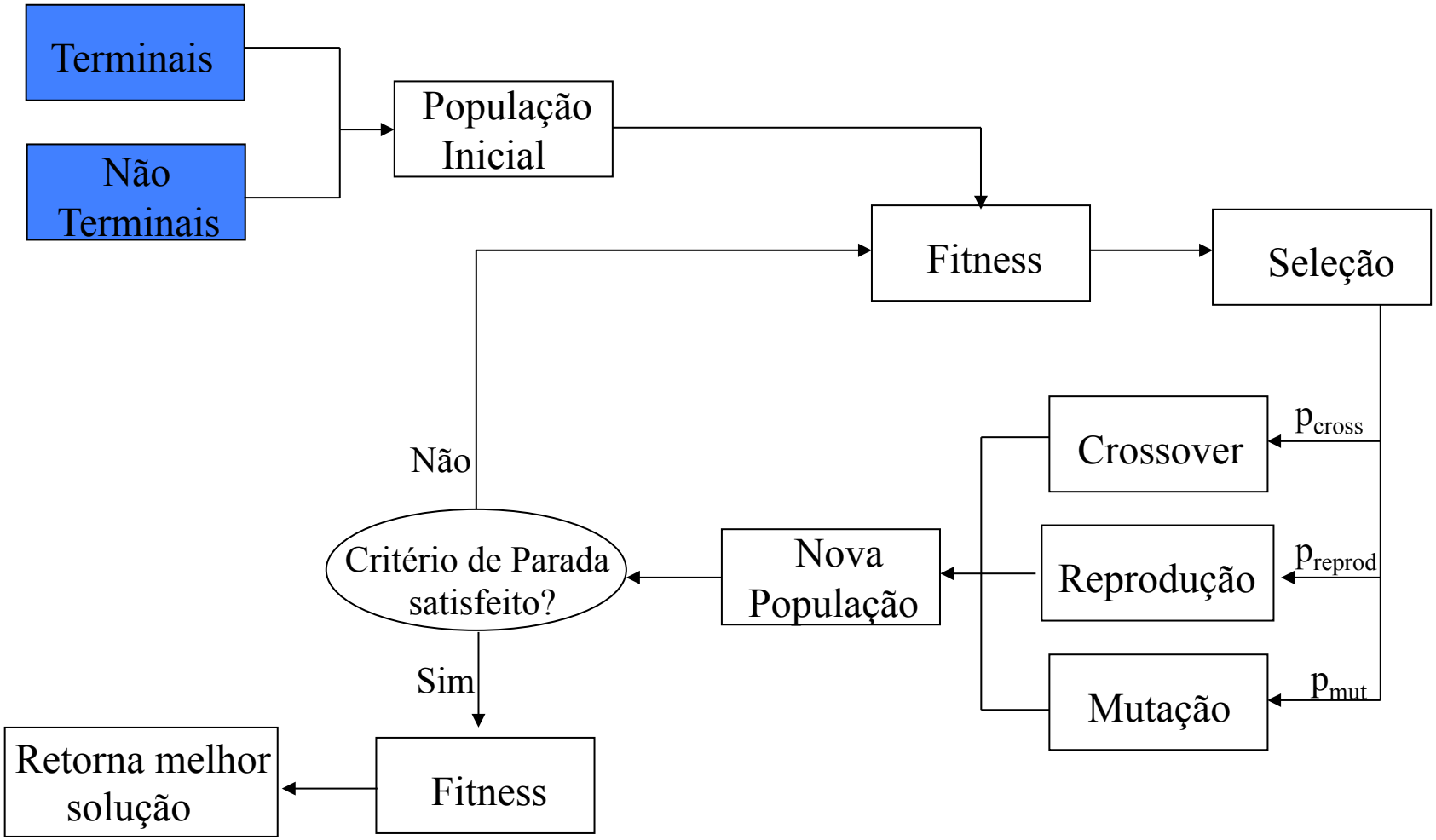


Pai



Filho

Programação Genética



Conjunto de Funções

- Propriedades desejadas
  - *Suficiência*: o poder de expressão é suficiente para representar uma solução candidata para o problema em questão
  - *Fechamento (Closure)*: uma função deve aceitar como entrada qualquer saída produzida por qualquer outro elemento do conjunto de funções ou do conjunto de terminais
  - *Parcimônia*: idealmente, conter apenas funções necessárias para resolver o problema em questão (propriedade não necessária, mas desejada)

Conjunto de Funções

- Como encontrar um equilíbrio entre poder de expressão e parcimônia?
  - Sugestão [Banzhaf et al. 1998, p. 111]:

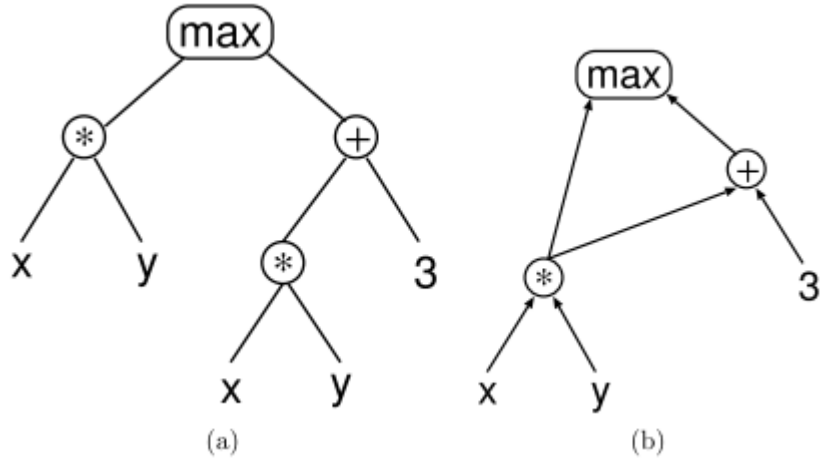
    FS = {+, -, x, /, OR, AND, XOR}

Conjunto de Funções

- Problemas com a propriedade de fechamento
  - Exige a modificação de certas funções/operadores
    - Ex. divisão por 0 é indefinida
    - Solução: divisão protegida – se o denominador for zero, retornar um valor padrão
  - Variáveis com diferentes tipos de dados
    - Ex., *Sexo* (binário: masculino, feminino) e *Idade* (inteiro)
    - Solução: GP restrito a sintaxe e/ou GP baseado em gramática

GP baseado em grafos

- Exemplo: *Parallel Distributed GP*
  - Evolução de programas altamente paralelos



Arestas representam
tanto estruturas de controle
quando fluxo de dados

Ilustração do mesmo indivíduo representado
por uma árvore (a) e um grafo(b)

GP baseado em grafos

- Na forma mais simples, as arestas do grafos são direcionadas e não rotuladas
- GP pode representar estruturas mais complexas, como redes neurais ou autômatos finitos, associando rótulos as arestas
  - Nesse caso, além dos terminais e funções, um conjunto de *links* (que dão nome as arestas) deve ser definido
- Cruzamento e mutação baseados em um grid

Introns

- Em biologia, introns são partes "inúteis" do DNA
  - Partes dos genes que não são utilizadas durante a produção de proteínas
- Em GP, introns são partes inúteis de um indivíduo, ou seja, partes do código que não tem nenhum efeito na saída do "programa" (solução candidata)
- Exemplos:
  - X = X + 0
  - X = X + X – X
- Introns são comuns em indivíduos de GP

Bloat

- Como resultado do aparecimento de introns, execução de um GP normalmente "incha" (*bloat*)
  - faz com que um indivíduo cresça incontrolavelmente até que o tamanho máximo seja alcançado
- Desvantagens de *bloating*
  - População utilizada muito mais memória
  - Execução do GP torna-se mais lenta
- Possíveis "vantagens" de introns
  - Efeito protetor contra o efeito destrutivo do crossover
  - Quanto maior a porcentagem de introns, maior a probabilidade do crossover trocar introns ao invés de trocar partes úteis do código

O que fazer em relação a introns e *bloating*

- Estimular a evolução de indivíduos mais simples, utilizando uma pressão de parsimônia (penelizando soluções complexas) na fitness
- Eliminar crossovers destrutivos, i.e., crossover é realizado apenas se os filhos gerados tiveram fitness pelo menos tão boas quanto as fitness dos pais (medida um pouco drástica)
- Implementar um operador que explicitamente remove introns depois que um indivíduo é gerado

Diferenças “Padrão” entre GA e GP

- Algoritmos Genéticos (GA)
  - Representação do indivíduo é, originalmente, um vetor binário. Atualmente vários tipos de representação são permitidas
  - Principal operador: crossover (altas probabilidades)
  - Operador secundário: mutação (baixa probabilidade)
- Programação Genética (GP)
  - Representação: utilizada não apenas dados, mas também funções
  - Objetivo original é evoluir programas, ou seja, “receitas” para um tipo de problema, ao invés de soluções para uma instância particular do problema
  - Acredita-se que crossover pode ter efeito destrutivo

GA versus GP

- Diferença está na interpretação da representação [Woodward 2003]
  - GA o mapeamento entre a descrição e o objeto sendo descrito é sempre um para um
  - GP esse mapeamento é de muitos para um
    - Em regressão simbólica, a mesma função pode ser descrita por diversos indivíduos

Leitura Recomendada

- A Field Guide to Genetic Programming, Livro online, http://www.gp-field-guide.org.uk/

Curiosidades

- http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/

- http://alphard.ethz.ch/gerber/approx/default.html

Agradecimentos

- A maioria dos slides dessa aula foram retiradas das aulas de Computação Natural de Alex A. Freitas