3
Programação Genética

3.1.
Introdução

Este capítulo faz uma breve introdução à área de Computação Evolucionária, destacando os conceitos que fazem parte da técnica de Programação Genética (PG).

Na natureza, estruturas biológicas mais bem sucedidas na adaptação a certos ambientes, vivem mais e se reproduzem em taxas mais altas [13]. A medida de adaptação a um certo ambiente é conhecida por aptidão. A aptidão ao longo de um período de tempo é responsável pela manutenção de estruturas através da seleção natural e pela transformação de estruturas através de recombinação sexual (crossover) e mutação.

Análogo à evolução dos seres vivos, a PG visa evoluir programas de computador para solucionar problemas do mundo real. Estes programas são submetidos a um processo evolucionário que envolve avaliação, seleção e operações genéticas como crossover e mutação. Após vários ciclos de evolução a população deverá conter indivíduos mais aptos. A cada indivíduo atribui-se um valor de adaptação conhecido como aptidão, que indica quanto a solução representada por este indivíduo é boa em relação às outras soluções da população.

Estruturas em PG são programas de computador que representam os indivíduos em uma população. Estes indivíduos são estruturas em forma de árvore compostas por nós e folhas. Os nós representam funções ou operações matemáticas que interligam uma ou mais folhas representando os terminais. Os terminais podem ser variáveis ou constantes.

Em geral, a PG gera programas de computador para resolver problemas executando os três passos a seguir:

1. Gera-se uma população inicial formada aleatoriamente pela composição de funções e terminais do problema (programas de computador).

2. Iterativamente processa-se os seguintes sub-passos até o critério de terminação ser satisfeito:
   a) Executa-se cada programa na população e associa uma aptidão de acordo com o quão bem foi resolvido o problema.
   b) Cria-se uma nova população de programas de computador aplicando-se os operadores genéticos. As operações são aplicadas aos programas de computador selecionados probabilisticamente pela sua aptidão.
3. O melhor programa de computador que apareceu em qualquer geração (o "melhor até agora") é designado como resultado da PG. Este resultado pode ser a solução (ou a solução aproximada) para o problema.

A seguir é feita uma breve descrição da representação dos indivíduos em PG. As estruturas da PG são essenciais ao conhecimento do processo evolucionário e da compreensão das operações genéticas sobre os indivíduos.

3.2. Representação

Na PG, o Indivíduo representa um programa de computador composto por funções e terminais. As funções são compostas de parâmetros para entrada de valores e retornam uma ou mais saídas. Os terminais são parâmetros que recebem um valor de entrada. A combinação de todas as soluções possíveis definidas por estas funções e terminais é denominada espaço de busca.

A estrutura de cada indivíduo na PG é representada em forma de árvore. Os terminais são folhas da árvore situados nas extremidades. As funções conectam um ou mais terminais situando-se nos vértices (nós) das árvores (como mostra a Figura 1).

Koza em [13] e [14] utiliza o termo *S-expression* (*Symbolic Expression*) para designar a representação de um indivíduo em PG representado sob a forma de uma expressão linear. Esta forma é a interpretação para uma estrutura representada através de uma árvore. Como exemplo, a representação de um indivíduo identificado pela expressão x^2+y é representado pela sua forma linear (+ *y* (* *x x*)) e por sua estrutura em árvore na Figura 1:

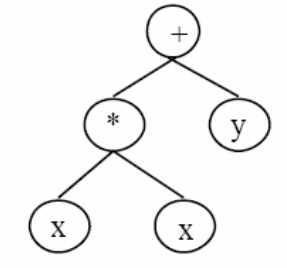

Figura 1 - Representação em árvore para indivíduo de Programação Genética [15].

Pode-se de acordo com [13][14][15], separar os elementos da estrutura arbórea em:

Funções: Aparecem nos vértices da árvore:

- operação aritmética (+, -, *, /, %, etc).
- função matemática (log, exp, sen, cos, etc).
- operação “booleana” (and, not, or).
- operadores condionais (se-então-senão).
- operadores iterativos (enquanto (condição) faça).
- funções recursivas.
- funções específicas do domínio do problema.

Terminais: Aparecem nas folhas das árvores:

- Variável.
- Constante.

O indivíduo representado na Figura 1 tem suas funções (no caso adição e multiplicação) e seus terminais (x e y) representado por pontos numericamente ordenados da direita para a esquerda. A ordenação destes pontos é mostrado na Figura 2. A interpretação do indivíduo se dá de acordo com esta ordenação, caminhando da raiz ou topo para o elemento mais a esquerda. A partir deste elemento caminha-se ponto a ponto até o elemento mais a direita.

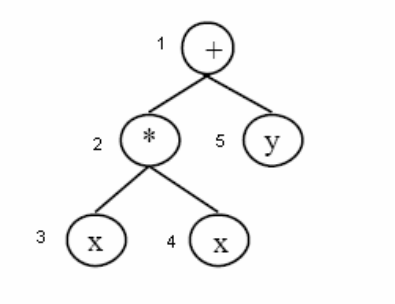

Figura 2 - Numeração dos pontos de um indivíduo.

Com os conjuntos de Terminais e de Funções definidos, a evolução por PG já possui a matéria prima necessária para criar um indivíduo.

Para que os indivíduos sejam válidos, devem atender a propriedade de clausura que estabelece que “funções devem aceitar como argumento qualquer valor ou tipo de dado que seja retornado pelo conjunto de funções e qualquer valor e tipo de dado que possa ser assumido por qualquer terminal” [13][14][15][16].

Após a breve descrição da representação de um indivíduo em PG e o significado dos elementos que fazem parte de sua estrutura, será descrito a seguir as pré condições para inicializar a PG, que depende desses elementos.

3.3. Preparação para Programação Genética

John R. Koza considera cinco passos preponderantes em [14] para montar um sistema de PG:

1. O conjunto de Terminais,
2. O conjunto de Funções Primitivas,
3. A medida de aptidão,
4. Parâmetros para controlar a execução do experimento,
5. Método para determinar um resultado e o critério de parada da execução.

O primeiro passo corresponde a determinação do conjunto de terminais. Os terminais correspondem às entradas do programa de computador.

O segundo passo é a identificação do conjunto de funções. As funções podem ser operações aritméticas padrão, funções de programação, funções matemáticas, funções lógicas, ou funções relativas ao domínio do problema conforme citado no item 3.2. As funções retornam um ou mais valores.

A medida de aptidão direciona o processo evolucionário em PG. Através dela, é determinado o desempenho de cada programa na população em relação ao ambiente do problema.

Os parâmetros de controle são sub-divididos em primários (tamanho da população e número de gerações) e os secundários (variáveis quantitativas e qualitativas) que devem ser especificados de forma a controlar a execução da PG.

Cada execução de PG requer um critério de parada, que pode ser qualitativo ou determinado pelo número de gerações, e um método para determinação dos resultados. É comum utilizar o melhor indivíduo obtido em toda a evolução como o melhor resultado. Este método também é conhecido como *best-so-far*.

O processo evolutivo na execução de um sistema de PG é ilustrado pela Figura 3. Na Figura 3 é mostrado uma população que pode ser inicial ou corrente. Os indivíduos desta população são selecionados conforme suas medidas de aptidão. Os mais aptos são selecionados para gerarem descendentes ou novos programas. Os novos programas são testados por uma função de avaliação que retorna suas aptidões e se bem avaliados farão parte da nova geração.

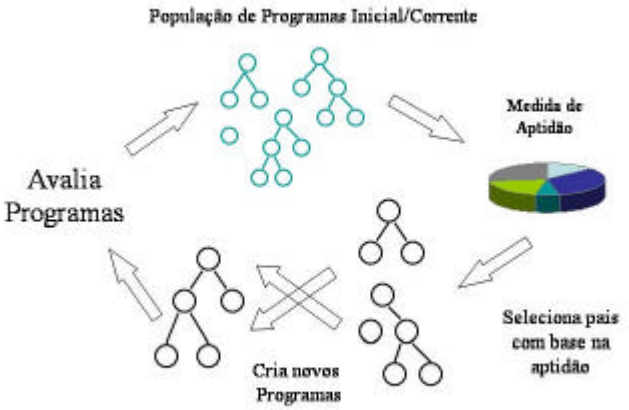

Figura 3 - Ilustração do processo evolutivo de um sistema de Programação Genética [15].

Pela Figura 3, pode-se observar um diagrama da evolução de uma PG numa geração qualquer. Para a criação dos indivíduos na geração inicial, geram-se estruturas em árvore aleatoriamente. A partir daí a avaliação de cada indivíduo será preponderante para criar os descendentes para as próximas gerações. A forma

de avaliar os indivíduos dentro da população ao longo das gerações é descrita a seguir.

3.4. Avaliação

A avaliação é o conceito absoluto atribuído ao indivíduo dentro de uma população. Avaliar um indivíduo é conhecer o desempenho deste em relação a um problema.

Em PG a avaliação de um indivíduo é medida pela sua aptidão. A aptidão é atribuída à um indivíduo na população através de um valor escalar numérico.

Para o cálculo da aptidão em PG é necessário possuir um conjunto de casos. Cada caso representa um estado do problema definido pelas variáveis do problema e o resultado produzido por seus diversos valores. Um caso pode ser entendido como um experimento onde observa-se a mudança do resultado produzido pela modificação dos valores de entrada das variáveis envolvidas. Cada indivíduo da população produz um resultado para cada caso. Os resultados produzidos para os valores de cada caso são comparados aos resultados dos respectivos casos. Essa comparação é utilizada para medir a aptidão do indivíduo.

Entre as aptidões utilizadas em PG, há as aptidões Bruta, Padronizada, Ajustada e Normalizada. A avaliação utilizada para medir o desempenho de um indivíduo em uma população é a aptidão normalizada. Para se calcular a aptidão normalizada é necessário primeiro calcular a aptidão bruta para depois calcular a aptidão padronizada e assim sucessivamente até o cálculo da aptidão normalizada. Essas aptidões estão descritas a seguir.

3.4.1. Aptidão Bruta

Aptidão Bruta é uma medida absoluta relacionada ao objetivo do problema.

Se o problema tem como objetivo encontrar o indivíduo que possua o menor erro em relação a resultados de experimentos conhecidos, pode-se definir a Aptidão Bruta como a soma dos desvios entre os resultados de um programa e os resultados obtidos por estes experimentos utilizados como amostras.

Sendo a Aptidão Bruta, uma medida de distância para uma solução amostral tida como referência, esta pode ser definida por:

$$r(i,t) = \sum_{j=1}^{N_e} \left| S(i,\ j) - C(j) \right| \quad \textbf{(26)}$$

Onde *r* é a aptidão bruta do indivíduo *i* na geração *t*. Esta aptidão é descrita como a soma das diferenças entre o valor retornado *S* pelo programa do indivíduo i para o caso j (experimento j) e o valor real do caso j (resultado j do experimento j) no conjunto de casos C. Isto é feito até atingir os N_e casos.

3.4.2. Aptidão Padronizada

A aptidão padronizada é uma medida que privilegia os indivíduos que a obtém com menor valor possível. Em problemas de minimização de erro, quanto mais próximo de zero estiver a aptidão padronizada, melhor é o indivíduo.

A Aptidão Padronizada é calculada em função da Aptidão Bruta podendo ser para certos problemas igual à aptidão bruta, ou uma subtração da aptidão bruta por um fator limitante ou qualquer medida em que o indivíduo com menor erro seja o melhor.

3.4.3. Aptidão Ajustada

A aptidão ajustada, é definida por John R. Koza em [13] por:

$$a(i,t) = \frac{1}{1 + s(i,t)} \quad \textbf{(27)}$$

Onde *s*(*i*,*t*) é a aptidão padronizada para um indivíduo *i* na geração *t*.

A aptidão ajustada possui valores entre 0 e 1 e as maiores aptidões são obtidas pelos melhores indivíduos.

O objetivo da aptidão ajustada é realçar as diferenças entre indivíduos que possuam aptidões padronizadas bastante próximas, o que é comum acontecer em gerações próximas ao final do ciclo de evolução [13].

3.4.4. Aptidão Normalizada

O cálculo da aptidão normalizada $n(i,t)$ é feito através do valor retornado pela aptidão ajustada $a(i,t)$ conforme o seguinte:

$$n(i,t) = \frac{a(i,t)}{\sum_{k=1}^{M} a(k,t)} \quad \textbf{(28)}$$

Onde i, é um indivíduo na geração t e k um indivíduo genérico. A aptidão normalizada é descrita como o resultado retornado da divisão da avaliação ajustada de um indivíduo i na geração t pela soma das avaliações ajustadas de todos os M indivíduos.

Há três características desejáveis para a aptidão normalizada:

- Varia de 0 a 1.
- É maior para o melhor indivíduo da população.
- A soma das aptidões normalizadas é 1.

Normalizar as aptidões é utilizado para comparar indivíduos de uma mesma geração. Como a normalização é feita entre indivíduos de uma mesma geração, é mais fácil saber quais indivíduos desta geração devem ser preservados, utilizados para criar novos indivíduos e quais devem ser destruídos.

3.5. Operadores

Os operadores em PG tem a função de criar descendentes dos indivíduos de uma geração.

Entre os operadores genéticos de PG, existem os operadores estruturais primários e secundários definidos assim por John R. Koza em [13].

Os primários são:

- Reprodução (cópia).
- Crossover.

Os operadores secundários são:

- Mutação.
- Permutação.
- Edição.
- Encapsulamento.
- Decimação.

Os operadores genéticos citados são detalhados a seguir.

3.5.1. Reprodução (cópia)

A operação de Reprodução permite ao indivíduo selecionado ser copiado para a geração seguinte sem sofrer qualquer modificação.

A seleção do indivíduo que será reproduzido é feita através da sua aptidão normalizada. A probabilidade de que um indivíduo seja copiado para a próxima geração é:

$$\frac{f(s_i(t))}{\sum_{j=1}^{M} f(s_j(t))} \quad \textbf{(29)}$$

Onde $f(s_i(t))$ pode ser considerada a aptidão normalizada $n(s_i(t))$ em (28).

A seleção do indivíduo para ser submetido à Reprodução pode ser feita por *Rank* ou por Torneio. Na seleção por *Rank*, há uma ordenação dos indivíduos por suas aptidões facilitando a escolha em favor dos melhores indivíduos. A ordenação acelera a busca dos melhores indivíduos e a inserção dos escolhidos nas gerações futuras. No Torneio, um grupo de indivíduos é escolhido aleatoriamente retirando-se uma fração deste grupo como indivíduos vencedores do Torneio. O número de indivíduos para reprodução pode ser determinado por

um parâmetro de controle assim como o número de indivíduos que fazem parte do Torneio.

3.5.2. Operador “Crossover”

O Crossover baseia-se na troca de material genético entre dois indivíduos. Para acontecer o Crossover, deve haver a seleção probabilística de dois indivíduos baseada nas suas aptidões normalizadas. A troca de segmentos será feita através de cortes nas estruturas individuais aleatoriamente, onde o pedaço do 1º indivíduo escolhido será substituído pelo pedaço do 2º indivíduo e vice-versa (conforme mostra a Figura 4).

A operação de crossover também acontece através das técnicas descritas no item 3.5.1, isto é, os indivíduos selecionados para gerar descendentes são sempre os indivíduos mais adaptados.

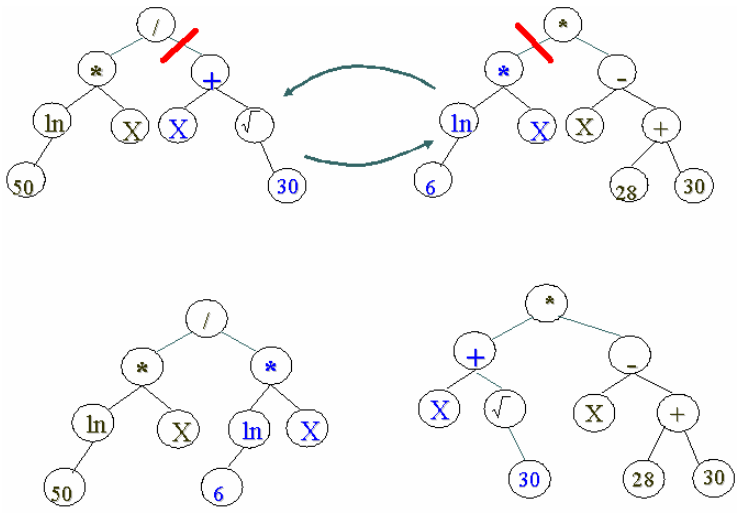

Figura 4 - Crossover entre dois indivíduos acima gerando os filhos abaixo com pedaços selecionados aleatoriamente nos pais trocados.

A Figura 4 mostra que dois indivíduos aptos selecionado para o crossover têm sorteado seus pontos de quebra. O indivíduo à esquerda no primeiro instante tem sorteado o ramo “$x+\sqrt{30}$” enquanto que o indivíduo à direita tem sorteado o ramo “x*ln(6)”. A troca destes ramos entre os dois indivíduos geram dois novos descendentes.

3.5.3.
Mutação

A Mutação opera sobre um indivíduo que é selecionado através de sua aptidão normalizada. O ponto em que a estrutura será alterada no indivíduo selecionado é gerado aleatoriamente. A partir da escolha aleatória do ponto de mutação, tudo que está abaixo deste ponto é substituído por uma nova sub-árvore aleatória (Figura 5). Esta nova sub-árvore terá um tamanho máximo controlado por um parâmetro que tem tipicamente um valor igual ao tamanho inicial da árvore para a população inicial aleatória.

A Figura 5 mostra que o nó representado por "$\sqrt{}$" é escolhido para mutação. A mutação nesse ponto provoca a troca do ramo composto por este nó e os demais pontos abaixo dele.

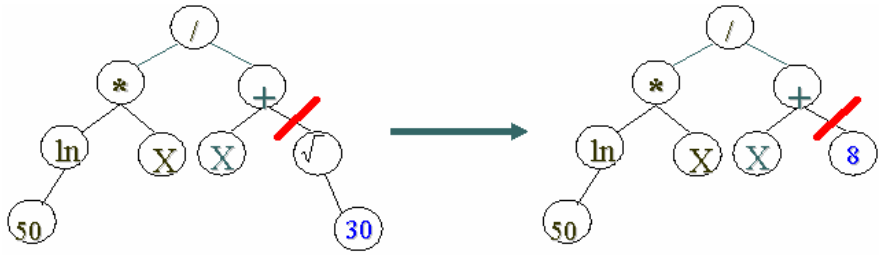

Figura 5 - Mutação. O segmento $\sqrt{30}$ foi substituído por 8.

3.5.4.
Permutação

A Permutação troca folhas ou terminais de um mesmo indivíduo respeitando a propriedade de clausura conforme Figura 6.

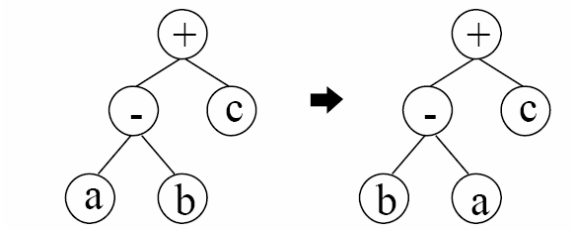

Figura 6 - Operação de Permutação [15].

Na Figura 6, um indivíduo é selecionado por sua aptidão normalizada e em seguida tem um nó sorteado para a permutação. No caso ilustrado, a função “a-b+c” se transforma em “b-a+c”.

3.5.5. Edição

A Edição contrai ou encolhe um indivíduo simplificando a estrutura existente como pode ser visto na Figura 7.

Na Figura 7 é mostrado a transformação da árvore que representa a função “*x*+*x*–*x*” na folha (terminal) que representa “*x*”.

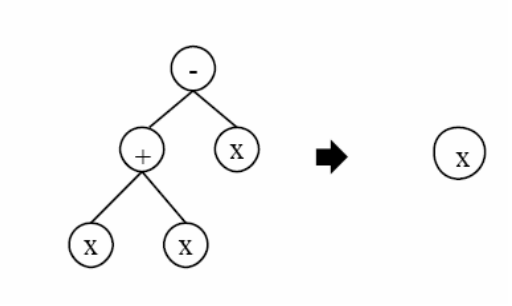

Figura 7 - Operação de Edição [15].

O Objetivo da Edição é preservar um programa que expresse um bom resultado, dos efeitos destrutivos das operações genéticas de crossover e mutação.

3.5.6. Encapsulamento

O Encapsulamento consiste em transformar uma sub-árvore em uma folha. Esta operação é útil para aproveitar uma sub-árvore ou descartá-la. Ao se ter uma sub-árvore encapsulada, um indivíduo fica livre de operações destrutivas desta sub-árvore como *crossover* e mutação. A operação de encapsulamento é ilustrada na Figura 8.

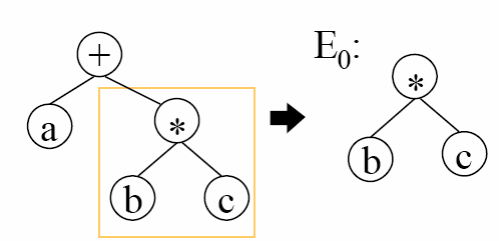

Figura 8 - Operação de Encapsulamento. A sub-árvore “b*c” se transforma em E_0 podendo ser re-aproveitado [15].

3.5.7. Destruição (*Decimation*)

A Destruição, é definida por dois parâmetros. Uma percentagem e uma condição especificando quando esta operação será utilizada.

A seleção dos indivíduos é baseada na aptidão de cada um. Se a destruição deve ser aplicada à geração inicial e a percentagem é 10%, a população da geração inicial deve ser criada 10 vezes maior do que o desejado, para que sejam removidos todos os indivíduos exceto os 10% melhores [13][15]. Com isto, cria-se uma quantidade 10 vezes maior do que o tamanho da população para re-inserir os 10% melhores como população inicial.

O objetivo da destruição é diminuir o esforço computacional dedicado para a evolução evitando diminuir a diversidade da população, pois com essa diminuição, gera-se convergência prematura.

3.6. Técnicas Evolucionárias

Entre as técnicas evolucionárias estão as técnicas de seleção dos operadores e técnicas de reprodução. Essas técnicas tem por objetivo acelerar a convergência da população encontrando um indivíduo satisfatório como solução ótima mais rapidamente.

Entre as técnicas de reprodução pode-se citar o Elitismo e o *Steady-State* que é uma extensão do Elitismo [17].

O Elitismo copia o indivíduo com melhor avaliação da geração corrente para a geração seguinte substituindo o pior indivíduo da geração seguinte. A substituição garante que o indivíduo com melhor avaliação da geração seguinte terá no mínimo, uma avaliação igual ao melhor indivíduo da geração anterior.

O *Steady-State* age como o Elitismo, porém substituindo *n* indivíduos piores da geração futura pelos *n* melhores da geração corrente, aonde *n* é um parâmetro da evolução também conhecido como *Gap*.

Como técnica para a seleção dos operadores pode ser utilizada a Roleta. Neste caso, cada operador genético tem como parâmetro um peso associado que é sorteado para aplicação sobre os indivíduos.

Além das técnicas evolucionárias, é necessário ter-se controle sobre o processo evolucionário. Para isso, a PG possui parâmetros onde pode-se variar tamanho da população, número de gerações, etc. para que a solução final retornada seja satisfatória.

3.7. Parâmetros de Controle

Na PG, são utilizados Parâmetros de Controle para explorar e tirar proveito ao máximo da evolução. Entre os mais usados tem-se:

- Tamanho da população: Número de indivíduos para cada geração
- Número máximo de gerações: Máximo de gerações que o programa irá evoluir até encerrar a execução
- Probabilidade de Mutação
- Probabilidade de Permutação
- Freqüência de Edição
- Freqüência de Encapsulamento
- Probabilidade de Reprodução
- Probabilidade de crossover
- Probabilidade de selecionar pontos para crossover. O valor sugerido por Koza em [13] é 0.90. O valor de 0.90 indica a probabilidade de ocorrer quebra em pontos mais internos da árvore, enquanto que 0.10 é a probabilidade de ocorrer quebra nos pontos mais externos (terminais ou folhas)
- Tamanho máximo permitido de um indivíduo criado por crossover. O tamanho máximo sugerido por Koza [13] é 17

- Tamanho máximo permitido para um indivíduo ao ser criado na população inicial ou geração zero
- Condição para utilizar a operação de Destruição (geralmente configurada para NIL)
- Percentagem para Destruição
- Método de geração da população inicial
- Método de geração dos pais para crossover (primeiro e segundo)
- Uso de aptidão ajustada

Os parâmetros descritos nesta seção são úteis para evoluir expressões matemáticas como é descrito a seguir.

3.8. Regressão Simbólica

Nesta seção é descrita a Regressão Simbólica (RS). A RS é uma aplicação de PG utilizada para inferência em conjuntos.

A RS é direcionada pela minimização do erro entre os resultados de um conjunto finito de amostras e o resultado retornado pelos indivíduos. A RS é tipicamente utilizada para problemas que necessitem de uma fórmula matemática como solução.

O objetivo da RS é encontrar uma função matemática que sirva para inferir sobre um conjunto de casos que possuem entradas de valor numérico para as variáveis. Para a obtenção de resultados, a RS utiliza um conjunto de Funções e Terminais necessários apenas para a obtenção de expressões matemáticas.

Dentre as Funções, podemos encontrar: +, -, /, x, Log, e, Ln, ^ (potenciação), $\sqrt{\ }$, N (Normal cumulativa), etc.

Para a avaliação dos indivíduos na RS utiliza-se a aptidão normalizada descrita em (28).

3.9. Técnicas para Aperfeiçoamento da PG

A técnica de PG, no princípio, apresentava algumas deficiências que prejudicavam seu desempenho ou os resultados encontrados. Nesta seção são

descritas algumas das técnicas utilizadas para aperfeiçoar a PG. Entre elas estão a STGP e a ADF.

3.9.1. Programação Genética Fortemente Tipada

A Programação Genética Fortemente Tipada (*Strongly Typed Genetic Programming* – STGP), foi sugerida pela primeira por Montana em [18].

Em PG, uma solução avaliada como ótima por suas aptidões numéricas, pode conter inconsistência de tipos, o que impede esta solução de ser viável para problemas reais. Para resolver este inconveniente, Montana propôs uma técnica [18], criando funções primitivas que aceitam apenas terminais de mesmo tipo. Esta técnica é conhecida por STGP (*Strongly Typed Genetic Programming*) e tem como contribuição à PG evitar a inconsistência na propriedade de clausura que determina que variáveis, constantes, argumentos para funções e valores retornados por estas funções devem ser do mesmo tipo [18].

O objetivo da STGP é encontrar soluções representadas por funções que não misturem tipos inconsistentes em operações matemáticas. Para tornar isso possível, é necessário criar novas operações matemáticas e funções que restrinjam os tipos a serem utilizados. Problemas que envolvem tempo e dinheiro podem apresentar uma função numericamente satisfatória como solução, porém, há o risco de se encontrar uma solução aparentemente satisfatória e bem avaliada (mesmo para um conjunto grande de amostras) que possua operações entre os dois. Para evitar esta inconsistência, criam-se tipos numéricos separados para operações próprias.

3.9.2. Função Automaticamente Definida (ADF)

Para permitir que a Programação Genética permeie por um espaço de busca mais definido, utiliza-se uma ou mais Funções Automaticamente Definidas (ADF-*Automatic Defined Functions*). A estrutura de uma ADF envolve alguns conceitos ilustrados na Figura 9.

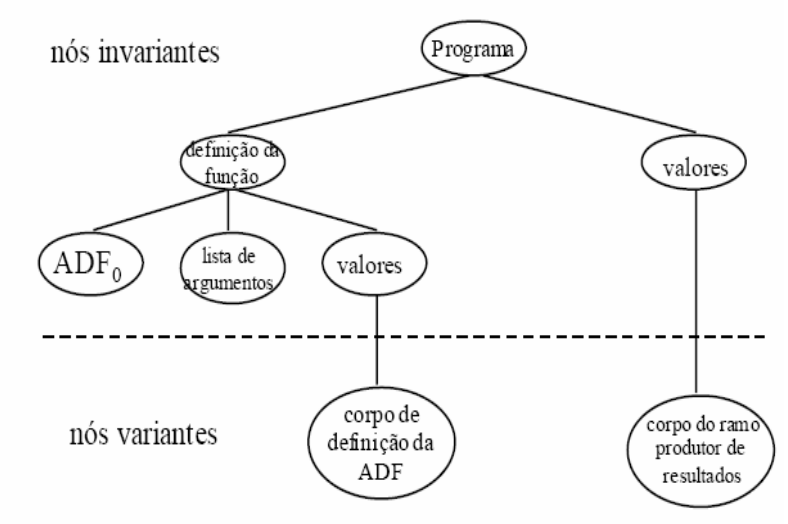

Figura 9 - Diagrama de uma ADF [15].

A Figura 9 mostra a estrutura de um indivíduo que possui um ramo para definição de função à esquerda e um produtor de resultados à direita.

Uma ADF comum tem oito tipos diferentes de pontos. Os seis primeiros tipos são parte da estrutura invariante. Os dois últimos tipos são parte da estrutura variante e constituem o corpo dos dois ramos (abaixo da linha pontilhada na Figura 9). Os oito tipos são, de acordo com Koza em [14]:

- A raiz da árvore que conecta a estrutura com o indivíduo definida por *programa*.
- O topo do ramo de *definição da função*.
- O nome da ADF, ADF_0.
- A *lista de argumentos*.
- A função de *valores* do ramo de definição da função, identificando o(s) valor(es) a ser(em) retornado(s) pela ADF.
- A função de *valores* do ramo produtor de resultados identificando os valores a serem retornados pelo ramo produtor de resultados.
- O corpo da ADF ADF_0.
- O corpo do ramo produtor de resultados.

Na geração da população inicial, todos os indivíduos são criados com a mesma estrutura invariante [13][14][15].

A evolução possui algumas peculiaridades na operação de crossover aplicada em ADFs. Este crossover tem como características:

- Preservar a mesma estrutura em todos indivíduos com um ramo de definição da função e um ramo de produção de resultados para programas com uma ADF.
- Para preservar a mesma estrutura em todos os indivíduos o crossover é aplicado apenas na estrutura não invariante do indivíduo.

Para ilustrar o crossover em um indivíduo que contenha uma ADF, utiliza-se a Figura 10.

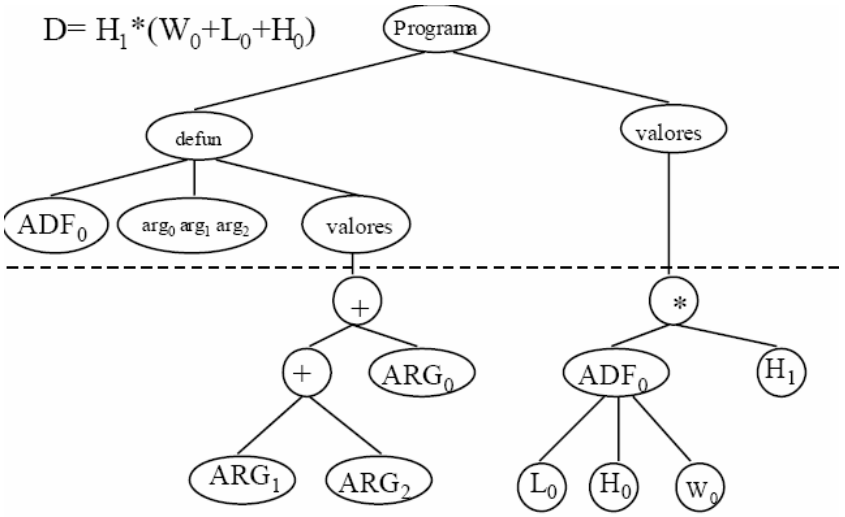

Figura 10 - Exemplo de uma ADF [15].

O crossover aplicado é ilustrado na Figura 11. O crossover só atua na parte da ADF que evolui, ou seja, na estrutura não invariante.

O caso estudado neste item diz respeito ao problema da diferença dos volumes de dois cubos citado por Koza em [14].

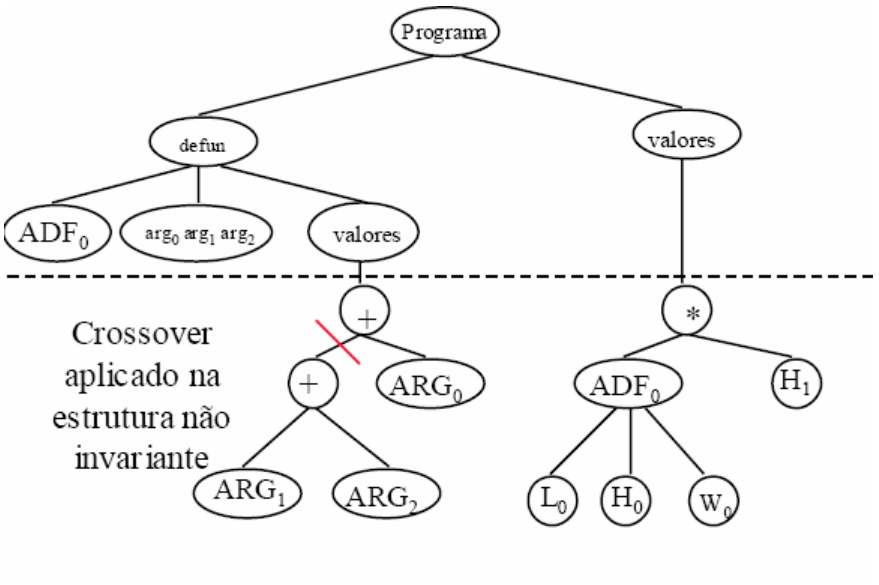

Figura 11 - Crossover aplicado a ADF [15].

No capítulo 4, é sugerido um modelo de inferência que utiliza a PG para encontrar uma função para curvas de exercício ótimo. Este modelo é uma aplicação PG de Regressão Simbólica, podendo ter futuras extensões com STGP e ADFs.