**Trabalho Inteligência Artificial 2014/1**

### Professor: Jacques Nelson Corleta Schreiber

**Alunos: Rafael Rodrigues Viana e Téu Solano**

**1. Introdução**

Este trabalho tem como objetivo criar um algoritmo, o qual tem a tarefa de encontrar uma expressão matemática cujo valor é o mais próximo a um valor escolhido aleatoriamente usando apenas as 4 operações básicas (+,-,/ e \*) e um conjunto predefinido de números. O jogo tem outras restrições: todos os resultados intermediários e o resultado final deve ser inteiros positivos. Restrições adicionais são também colocadas sobre os números selecionados e objetivo. O jogador recebe quatro números aleatórios entre 1 e 9, um número entre os três seguintes {10, 15, 20} e um entre os números {25, 50, 75, 100}. O número objetivo é escolhido aleatoriamente a partir do intervalo [100,999]. O algoritmo pode usar os números selecionados apenas uma vez. Para encontrar as melhores expressões matemáticas este trabalho utiliza um algoritmo genético que será descrito nos capítulos seguintes.

**2. Definição de Cromossomo**

Um algoritmo genético, da mesma forma que acontece com a espécie humana, é formado por uma população que evolui com o decorrer de novas gerações. O cromossomo é a forma de representar um individuo de uma população, neste trabalho ele é criado em forma de uma árvore de expressão binária, na qual os nodos folha são os operandos (números) e os nodos pais são os operadores. Um exemplo de expressão em forma de árvore binária de expressão é mostrada abaixo.

**3. Criação de indivíduos da população inicial**

A população inicial é composta por indivíduos gerados a partir do algoritmo descrito abaixo. Se este algoritmo gerar um filho inválido, este filho não é utilizado. Se a população é composta de 10 indivíduos, a população inicial será composta com os 10 primeiros cromossomos válidos gerados.

**Definição do algoritmo:**

**1º Passo:** A raiz será sempre um operador. Por isso, deve-se gerar um operador aleatório para a raiz. Todos os nodos quando são criados são inseridos em uma lista aberta. Assim, que um nodo tem o seu filho do lado esquerdo e do lado direito ocupados ele é retirado desta lista aberta.

**2º Passo:** Pegar um nodo aleatório da lista aberta.

Com o nodo selecionado:

Se não possui nenhum filho, sorteia se insere no lado direito ou esquerdo.

Se já possui um filho do lado esquerdo, deve-se inserir o novo nodo à direita.

Se já possui um filho do lado direito, deve-se inserir o novo nodo à esquerda.

Esta inserção de novo nodo será escolhido aleatoriamente se será um operador ou um número. Após a inserção de um novo nodo, se ele ainda não estiver inserido na lista aberta, ele deve ser incluído na lista aberta.

Existem algumas exceções que a inserção deverá ser obrigatoriamente um operador ou obrigatoriamente um número, conforme exemplos abaixo.

**Exceção 1:** Não pode incluir operador se ainda não atingiu a quantidade de números desejada e o nodo é o último a estar na lista aberta.

Neste caso, se preciso incluir um nodo à direita do + é obrigatório incluir um operador para chegar aos seis números. Se incluísse um número a árvore teria somente três números.

**Exceção 2:**

Neste caso, é obrigatório o incluir um novo nodo como número porque se incluísse um operador , teria, no mínimo, sete números na expressão.

**4. Função Aptidão (Fitness)**

A função aptidão é utilizada para determinar os indivíduos mais aptos da população. Quanto mais apto o individuo está, mais próximo do resultado ele está. Por isso, se utiliza como função aptidão a função abaixo:

1000 / | resultado da árvore - número alvo | = diferença para o número alvo

Poderia usar 1 para fazer a inversão, mas foi usado o número 1000 para se ter números maiores, para facilitar na criação da roleta.

**Resultado da árvore** = A árvore será percorrida de maneira in-fixada para encontrar o resultado da expressão.

**Número alvo** = Número que está procurando.

Usa-se o módulo para obter o número absoluto, ou seja, se estou procurando o número 100, tanto 99 quanto 101 terão a mesma aptidão. Exemplo: Quando se deseja encontrar o número 100 e tem uma expressão com o resultado 120. Ele terá o fitness = 1000 / | 120 – 100 | = 50. Outra expressão com o resultado de 140, terá o fitness = 25. Ele será menos apto do que o 120 que está mais próximo.

**5. Roleta**

A escolha dos indivíduos que serão usados no cruzamento é realizada através do algoritmo de roleta. Neste algoritmo, é criada uma tabela com todas as funções aptidões e o seu percentual relativo ao valor total de aptidão.

Valor Esperado = 100

|  |  |  |  |
| --- | --- | --- | --- |
| **Indivíduo** | **Valor do Indivíduo** | **Função de Aptidão** | **Fitness** |
| 1 | 99 | 1/[100-99] | 1/1=**1** |
| 2 | 101 | 1/[100-101] | 1/1=**1** |
| 3 | 110 | 1/[100-110] | 1/10=**0,1** |
| 4 | 200 | 1/[100-200] | 1/100=**0,01** |

Tabela 1. Exemplo de população e fitness.

Fitness Total= **2,11**

|  |  |  |  |
| --- | --- | --- | --- |
| Indivíduo | Função | Porcentagem | Soma |
| 1 | 1/2,11=**0**,**4739** | **0**,**4739 x 100 =47,39** | 0 + 47,39 = **47,39** |
| 2 | 1/2,11=**0,4739** | **0,4739 x 100=47,39** | 47,39 + 47,39 = **94,78** |
| 3 | 0,1/2,11=**0,0473** | **0,0473 x 100=4,73** | 94,78 + 4,73 = **99,51** |
| 4 | 0,01/2,11=**0,0048** | **0,0048 x 100=0,48** | 99,51 + 0,48 = **99,99** |
|  | **100%** |  |  |

Tabela 2. Exemplo de roleta.

**6. Crossover**

O crossover (cruzamento) de indivíduos é a técnica utilizada em algoritmos genéticos para a geração de novos indivíduos. Nesta técnica, se escolhe dois indivíduos da população atual através do método da roleta descrito no capítulo 5.

Para realizar o cruzamento de dois indivíduos é necessário apresentar o conceito utilizado neste trabalho de pontos de corte em um árvore. Cada árvore possuirá uma quantidade de pontos possíveis de corte igual ao (número de números folhas – 2). Em um arvore com quatro nodos folhas, terão dois pontos possíveis de corte, conforme a imagem abaixo. Os pontos de corte acontecerão na aresta de dois nodos com operadores, não há ponto de corte entre um nodo com operador e um nodo com número.

ponto de corte 1 ponto de corte 2

Inicialmente, para realizar o cruzamento, optou-se por escolher um ponto de corte aleatoriamente no pai e um ponto de corte aleatoriamente na mãe. Porém, estes cruzamentos geravam uma alta taxa de expressões inválidas. Percebeu-se que em um árvore com seis números existirão quatro pontos de corte, logo, se o pai e a mãe tiveram seis números, será possível gerar 16 combinações possíveis de cruzamentos com a combinação de todos os pontos de cortes.

Portanto, optou-se para cada crossover fazer o cruzamento de todos os pontos possíveis de corte e gerar todas as combinações possíveis com as duas árvores selecionadas, as árvores inválidas são desconsideradas. As árvores válidas geradas, juntamente com toda a população atual são incluídas em uma lista, na qual são escolhidos os 10 melhores indivíduos.

**Exemplo de crossover:**

Pai

Mãe

Ponto de corte no pai = 2.

Ponto de corte na mãe = 2.

Observação: Nos pontos de cortes se inicia a numeração em 1 e são contabilizados da esquerda para a direita de cima para baixo. Por exemplo, no pai mostrado acima, o ponto de corte 1 é o traço que liga o nodo com operador + e o nodo com o operador -. E o ponto de corte 2 é o traço que liga o nodo com o operador – e o nodo com o operador +.

Filho gerado com o cruzamento do ponto de corte 2 do pai com o ponto de corte 2 da mãe. Os nodos abaixo do ponto de corte 2 do pai são substituídos com os nodos abaixo do ponto de corte 2 da mãe.

Filho gerado com o cruzamento do ponto de corte 2 da mãe com o ponto de corte 2 do pai. Os nodos abaixo do ponto de corte 2 da mãe são substituídos com os nodos abaixo do ponto de corte 2 do pai.

**Exemplo de crossover apresentado pelo software desenvolvido neste trabalho:**

Filho 1 => (1+((((2+4)-3)+25)\*10))

Filho 2 => (3+(25\*(10+4)))

Ponto de Corte Filho 1 => 0

Ponto de Corte Filho 2 => 0

Filho 1 (apos crossover) => (1+(25\*(10+4))) = 351 (válido)

Filho 2 (apos crossover) => (3+((((2+4)-3)+25)\*10)) (inválido)

Filho 1 (após mutação) => (1+(25\*(10+4))) = 351 (válido)

Filho 2 (após mutação) => (3+((((2+4)-3)+25)\*10)) (inválido)

Todos os indivíduos válidos são incluídos em uma lista juntamente com os indivíduos da população atual. Eles são ordenados em ordem decrescente de fitness, de modo que os melhores indivíduos estejam nas primeiras posições da lista. São escolhidos os 10 melhores indivíduos para se manter na população da próxima geração.

Este processo de cruzamento e avaliação de indivíduos de uma população para a geração de uma nova população é repetido até encontrar uma expressão que encontre o resultado procurado, ou, então até um número limite de gerações (pois nem sempre se encontra o resultado procurado) ou até atingir um tempo determinado.

**7. Elitismo**

A técnica citada acima de manter sempre os melhores indivíduos na população é chamada de **elitismo** ela tem como vantagem nunca perder o melhor elemento encontrado, porém apresenta problemas graves para a convergência de melhores resultados, já que como se mantém sempre os melhores indivíduos, quando se encontra os melhores resultados, dificilmente se encontrará um resultado melhor, logo, os mesmos indivíduos se manterão na população por mais gerações. Com isso, se perderá a diversidade da população e será cada vez mais difícil de aprimorar as expressões.

**8. Mutação**

A mutação é uma técnica para tentar gerar uma diversidade maior nos indivíduos da população. Ao aplicar a mutação a uma árvore cada nodo do individuo é avaliado individualmente, e, será aleatoriamente escolhido alguns destes para serem modificados por outro valor.

Por exemplo, se a taxa de mutação é 2% cada nodo do individuo será avaliado e será sorteado um número entre 0 e 100, se este número for menor que 2, este nodo será alterado para outro valor aleatório Se este nodo é um número será trocado por um número aleatório (conforme números que estão sendo utilizados), se este nodo é um operador será por um operador aleatório.

**9. Conclusão**

Este trabalho foi muito importante para fixar os conhecimentos aprendidos em aula sobre algoritmos genéticos. Como a maioria dos algoritmos genéticos encontrados na literatura têm seus cromossomos representados através de bits, encontrou-se uma dificuldade para adaptar os conceitos de função aptidão, crossover e mutação para o cromossomo em forma de árvores de expressão binária. Porém, esta dificuldade foi fundamental para estudar com mais profundidade a teoria de algoritmos genéticos.