Universidade Federal de São João del-Rei
Algoritmo Bioinspirados
Implementação de Algoritmo Genético com Representação Binária

Felipe Francisco Rios de Melo

1. INTRODUÇÃO

Um algoritmo genético (AG) é uma técnica de busca utilizada para achar soluções aproximadas em problemas de otimização e busca. (usam técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação)

Estes métodos estão sendo utilizados, cada vez mais, pela comunidade de inteligência artificial para obter modelos de inteligência computacional (Barreto 1997).

Por se tratar de uma técnica bastante difundida e geralmente oferecer bons resultados, os algoritmos genéticos são usados para solucionar diversos tipos de problemas, entre eles: Sistemas de classificação, escalonamento e grade horária.

2. IMPLEMENTAÇÃO

A seguir, será mostrado um resumo das decisões tomadas para a implementação do Algoritmo Genético:

2.1. Estrutura de dados

Foi implementado uma classe nomeada AG, que é instanciada uma única vez, contendo os atributos e métodos necessários para inicialização e manipulação do algoritmo.

- Atributos: tamanho da população, número de gerações, taxa de mutação, probabilidade de vitória na seleção por torneio, número de bits de cada indivíduo, o limite inferior (xmin) e superior (xmax);
- Métodos: gerar população, avaliar população, seleção por torneio, crossover, mutação e etc.

Além da classe AG, existe também a classe Indivíduo, usada para alocar os parâmetros (x1 e x2) de cada indivíduo, bem como também, seu fitness na função objetivo.

2.2 População

A população foi gerada como uma matriz com n x m, onde n é o tamanho da população e m o número de bits. O valor de cada elemento, 0 ou 1, é obtido aleatoriamente.

2.3. Indivíduos

Obtida a população inicial, para cada indivíduo da população, instanciamos um objeto da classe *Individuo*, e setamos o seu x1 e x2, onde x1 corresponde aos bits de [0, 5], e x2 os bits do intervalo [6, 11].

2.4. Função Objetivo

Para todos os indivíduos da população, ajusta-se os parâmetros para o intervalo de busca sugerido através da seguinte função:

$$x_n = xmin + (xmax - xmin) / (2^{nbits/2} - 1) * (\text{conversão de } x_n \text{ para float})$$

Por fim, aplica-se a função objetivo, guardando o resultado no atributo avaliação na instância da classe *Individuo,* referente ao indivíduo em questão.

2.5. Elitismo

Busca em todos os indivíduos, àquele que obteve a melhor avaliação (mais próximo de zero), e guarda-o para ser usado mais adiante.

2.6. Seleção

A seleção dos indivíduos para o cruzamento é feito por torneio. No torneio, é sorteado dois indivíduos, e estes indivíduos têm seu fitness comparado. O indivíduo com melhor fitness têm uma maior chance de se consagrar vencedor do torneio. A chance de vitória é dado por um atributo *prob_vitoria* (estipulado na instanciação da classe AG). Na implementação foi usado como 90% como probabilidade de vitória ao indivíduo que contém o maior fitness.

Por questões de implementação, a partir do momento que um indivíduo vence um torneio, ele é removido da lista de indivíduos, para que nas próximas chamadas da função torneio() não corra o risco dele ser selecionado novamente.

No fluxo do código, é chamado duas vezes a função torneio(). O vencedor do primeiro torneio chamamos de *pai1*, e do segundo, *pai2*.

2.7. Crossover

Tendo os pais, conseguimos realizar cruzamentos. No crossover, é necessário um ponto de corte, para que haja herança genética (herança de bits) por parte dos dois pais. Este ponto de corte é obtido pelo sorteio de um valor compreendido entre 0 e 11 (número de bits de cada indivíduo).

Tendo o ponto de corte, podemos gerar filhos. Para reposição de toda a população inicial, é gerado dois filhos por casal. O primeiro filho recebe, do *pai1*, os bits de 0 até o ponto de corte e o restante do *pai2*. Já o segundo filho recebe, do *pai2*, os bits de 0 até o ponto de corte e o restante do *pai1*.

Em seguida, adiciona-se os dois filhos à nova população, criando instâncias para cada um deles de classe *Individuo*.

2.8. Nova População

Este processo de torneios (seção 2.6) e crossovers (seção 2.7), é repetido até que a nova população tenha obtido o tamanho da população anterior.

Logo, a nova população está no tamanho fixado, porém, devido a decisão de implementação do Elitismo - o melhor indivíduo da geração, sobrevive para a geração seguinte - é necessário abrir um espaço na nova população para o melhor indivíduo. Então, optou-se por aleatoriamente, remover um indivíduo da nova população para inserir o indivíduo mais bem avaliado.

2.9. Mutação

Com a nova população em mãos, para cada indivíduo dela, sorteamos um inteiro entre 0 e 100, se este inteiro for menor ou igual a taxa de mutação (estipulada na instanciação da classe AG), o indivíduo sofrerá uma mutação.

A mutação provoca a inversão de um bit aleatório (entre 0 e 11) do código genético do indivíduo.

2.10. Gerações

O passo a passo da seção 2.4 até a 2.9 é repetido para todas as gerações. A quantidade de gerações é estipulado no momento da instanciação da classe AG.

Em cada geração, o melhor indivíduo é imprimido na tela, junto a sua avaliação na função objetivo.

3. ANÁLISE DOS RESULTADOS

Para as entradas de sugestão, no documento de proposta do trabalho (tamanho população = 50; número de gerações = 100; taxa de mutação = 10; número de bits = 12; xmin = -2 e xmax = 2, as saída, resumidamente, foi o seguinte:

TABELA 1. Execução do AG ocultando iterações intermediárias.

Melhor individuo da geração 0 -> Fitness: 3.123800345277538 x1: -0.0952380952380953 4 x2: 0.4761904761904763
Melhor individuo da geração 1 -> Fitness: 2.842204781054637 x1: -0.9841269841269842 x2: 0.0952380952380949
Melhor individuo da geração 2 -> Fitness: 0.5327205100166741 x1: 0.03174603174603163 x2: 0.0952380952380949
Melhor individuo da geração 3 -> Fitness: 0.5327205100166741 x1: 0.03174603174603163 x2: 0.0952380952380949
Melhor individuo da geração 4 -> Fitness: 0.5327205100166776 x1: 0.03174603174603163 x2: -0.09523809523809534
Melhor individuo da geração 5 -> Fitness: 0.5327205100166776 x1: 0.03174603174603163 x2: -0.09523809523809534
Melhor individuo da geração 6 -> Fitness: 0.5327205100166776 x1: 0.03174603174603163 x2: -0.09523809523809534
Melhor individuo da geração 7 -> Fitness: 0.5327205100166776 x1: 0.03174603174603163 x2: -0.09523809523809534
Melhor individuo da geração 8 -> Fitness: 0.5327205100166776 x1: 0.03174603174603163 x2: -0.09523809523809534
Melhor individuo da geração 9 -> Fitness: 0.1799478151580023 x1: 0.03174603174603163 x2: -0.03174603174603185
.
.
.
Melhor individuo da geração 98 -> Fitness: 0.1799478151580023 x1: 0.03174603174603163 x2: 0.03174603174603163
Melhor individuo da geração 99 -> Fitness: 0.1799478151580023 x1: 0.03174603174603163 x2: 0.03174603174603163

É possível observar que o fitness converge rapidamente. No caso desta execução do algoritmo, ele convergiu na geração 9.

A solução ótima deste problema é 0, a função do algoritmo genético é chegar o mais próximo possível da solução ótima, com o espaço de busca limitado entre -2 e 2, chegou-se em 0.1799478151580023.

Fixando o espaço de busca entre 2 e 2, e variando os outros parâmetros como tamanho da população, taxa de mutação e número de gerações, em nenhuma execução obteve-se um fitness melhor que 0.1799478151580023.

Já variando o espaço de busca, quanto menor a amplitude melhor é o resultado final, em contrapartida, a medida que o escopo amostral cresce pior fica o fitness. O que é de se esperar, pois quanto maior o espaço de busca, maior as possibilidades de valores para os parâmetros x1 e x2, levando mais tempo para o solução convergir.

Outra análise importante, é a relevância do elitismo na evolução ao passar das gerações. Abaixo está um trecho de uma saída do programa, sem o uso de elitismo:

TABELA 2. Execução do AG sem elitismo.

Melhor individuo da geração 0	->	Fitness: 1.5627014812797673
Melhor individuo da geração 1	->	Fitness: 1.5627014812797673
Melhor individuo da geração 2	->	Fitness: 0.8109197657095
Melhor individuo da geração 3	->	Fitness: 2.740439182286281
Melhor individuo da geração 4	->	Fitness: 2.773323195518437
Melhor individuo da geração 5	->	Fitness: 2.045366001208482
Melhor individuo da geração 6	->	Fitness: 2.045366001208482
Melhor individuo da geração 7	->	Fitness: 2.045366001208482
Melhor individuo da geração 8	->	Fitness: 1.5627014812797673
Melhor individuo da geração 9	->	Fitness: 0.1799478151580023

Agora os mesmos parâmetros mas com elitismo:

TABELA 3. Execução do AG com elitismo

Melhor individuo da geração 0	->	Fitness: 3.1909145217461696
Melhor individuo da geração 1	->	Fitness: 1.2524899991331933
Melhor individuo da geração 2	->	Fitness: 0.5327205100166812
Melhor individuo da geração 3	->	Fitness: 0.8109197657095
Melhor individuo da geração 4	->	Fitness: 0.1799478151580023
Melhor individuo da geração 5	->	Fitness: 0.1799478151580023
Melhor individuo da geração 6	->	Fitness: 0.1799478151580023
Melhor individuo da geração 7	->	Fitness: 0.1799478151580023
Melhor individuo da geração 8	->	Fitness: 0.1799478151580023
Melhor individuo da geração 9	->	Fitness: 0.1799478151580023

Observamos que a AG com o método elitista converge muito mais rápido, isso devido ao fato de que o melhor indivíduo de determinada geração vai perseverar pelas gerações até que exista um melhor do que ele. Logo, se o algoritmo ser executado com 20 gerações, e um indivíduo da geração 4 consegue o melhor fitness possível, caso não houvesse o elitismo, o indivíduo geraria dois filhos compartilhando apenas uma parte de seus genes (a chance dos filhos não serem tão bons quanto o pai é alta), e por fim seria removido. Esta condição descrita, desperdiçaria uma solução que tinha o potencial de ser a melhor. Assim, vemos a importância do elitismo na implementação do Algoritmo Genético.

4. CONCLUSÃO

Com este exercício de implementação foi possível colocar em prática a codificação de um algoritmo genético com representação binária, e entender um pouco mais sobre o seu funcionamento.

Quanto às análises dos resultados, apesar de superficiais, foi capaz de mostrar a relação entre cada parâmetro da AG e a função objetivo.

5. REFERÊNCIAS BIBLIOGŔAFICAS

O que são algoritmos genéticos? - StackOverflow - Disponível em: <https://pt.stackoverflow.com/questions/185996/o-que-s%C3%A3o-algoritmos-gen%C3%A9ticos>