Universidade Federal do ABC

Comparação entre um Algoritmo de Enxame de Populações Discreto e um Algoritmo Genético:

Projeto Final de Inteligência Artificial

Felipe Anchieta Santos Costa
Pedro Henrique Gomes
Rodrigo Martins de Oliveira

Santo André – SP
2015

Sumário

1 Introdução

Algoritmos genéticos vêm sendo utilizados para resolver diversos problemas de otimização que não possuem solução determinística, em problemas como N-Puzzle e N-Queens. Um célebre uso recente dos algoritmos genéticos foi na implementação da *evolved antenna* da espaçonave ST5 da NASA, em 2006.

Estes algoritmos possuem uma base bastante simples e similar à evolução darwiniana, onde uma população evolui através da seleção natural, mecanismo onde indivíduos mais propícios à sobrevivência tem mais chances de reprodução, levando a um aumento na população desses indivíduos mais propícios. Para isso, a reprodução dos seres vivos depende da combinação dos genes maternos e paternos e mutações no código genético, conferindo um caráter aleatório a este gene, que pode ser benéfico ou maléfico da sua probabilidade de sobrevivência e posterior reprodução.

A otimização por enxame de partículas, do inglês *Particle Swarm Optimization*, é um algoritmo de inteligência artificial onde, dada uma população inicial P e suas partículas $p_i \in P$, cada qual com posições x_i e velocidades v_i, o algoritmo otimiza esta população para uma dada meta de acordo com a melhor posição vizinha de cada uma dessas partículas.

2 Objetivos

Este projeto tem como fim executar testes de um algoritmo genético e um algoritmo de otimização por enxame de partículas em três problemas sem solução determinística em tempo polinomial de computação, que são os problemas N-Puzzle, N-Queens e Labirinto na linguagem de programação Python, que vem sido trabalhada no curso de Inteligência Artificial por ser uma linguagem corriqueira nesta área da Computação, tendo como características a orientação a objetos e ser uma linguagem interpretada.

3 Metodologia

Para a implementação dos algoritmos deste projeto, algoritmo genético e PSO, foi utilizada uma abordagem modular para o desenvolvimento do software. Existem dois algoritmos em python no projeto, o mGenetic e o pso, e módulos de benchmark para os testes dos algoritmos. Um módulo general_settings.py foi utilizado para permitir a fácil comunicação dos algoritmos com os módulos.

3.1 Implementação do Algoritmo Genético

A implementação do algoritmo genético seguiu a definição mais tradicional presente na literatura. Resumidamente, uma população de genes é tomada e são realizados processos reprodutivos com os indivíduos desta para gerar uma prole. No processo reprodutivo ocorrem crossovers entre genes e mutações com uma determinada probabilidade. Os piores indivíduos da população resultante são, então, eliminados e o processo se repete até que o critério de parada seja atingido.

3.2 Implementação do Algoritmo de Otimização por Nuvem de Partículas Discretas

A implementação do PSO seguiu o modelo de algoritmo proposto por Spears (2010), que utiliza o parâmetro ω para as partículas do enxame. Implementado a versão de Spears, acrescentamos as modificações propostas por Kennedy e Eberhart (1997) para correta discretização das partículas da nuvem.

3.3 Benchmarks

Os benchmarks foram separados em módulos que geram uma população inicial apropriada para o problema, fornecem uma função check(gene) para retornar o fitness de um gene da população e uma função stopCriteria() que devolve verdadeiro caso o critério de parada para aquele problema tenha sido atingido.

4 Resultados

O algoritmo genético, para o benchmark das N-Rainhas, leva em média ˜200 gerações com desvio padrão de ˜130 gerações. Já o algoritmo PSO leva em média ˜900 gerações com desvio padrão de ˜400 gerações.

5 Conclusão

O algoritmo genético apresentou um melhor desempenho no problema das N-Rainhas em relação ao algoritmo de otimização por enxame de partículas discretas. No entanto este resultado não é conclusivo para a comparação dos dois algoritmos, pois a implementação do algoritmo PSO não está adequada para trabalhar com os vetores de população fornecidos pelos módulos, onde cada componente da solução é representado por N bits contíguos e não por um único bit, como o PSO interpreta o vetor.

Portanto, para o problema das N-Rainhas, por exemplo, num tabuleiro 8x8, cada componente é composto por 3 bits, mas o algoritmo PSO interpreta cada componente representado por 1 bit. Logo o processo de movimentação de partículas no PSO é amortizado pela divisão em 3 sub-dimensões do hipercubo que cada solução representa e, consequentemente, precisa de muitas mais gerações de partículas até que se atinja o critério de parada do que caso a interpretação dos vetores de soluções estivesse correta.

Corrigir este problema não é uma tarefa que carrega grande complexidade, porém demanda um tempo de implementação maior do que dispomos.