Pontifícia Universidade Católica – Rio Grande do Sul

Relatório – Trabalho I - Inteligência Artificial

Nomes: Lorenzo Fiore Bonder, Thiago Roly Gomes
Turma: 127
Professora: Silvia Moraes

Introdução

Este documento consiste no relatório sobre o primeiro trabalho proposto na cadeira de Inteligência Artificial, turma 127, semestre 2019/2. O trabalho consiste em ler um labirinto, em formato de arquivo .txt e implementar um algoritmo genético, que tentará descobrir a saída desse labirinto e posteriormente implementar um algoritmo em estrela, que esse sim descobrirá sempre a saída para o labirinto informado.

Nesse documento explicaremos como foi feito a implementação do código fonte, o tamanho da população inicial para o algoritmo genético, as funções heurísticas para os dois algoritmos, operadores escolhidos, as taxas de mutações e cruzamento dos cromossomos do algoritmo genético e por fim, possíveis contravenções e considerações sobre o desenvolvimento do trabalho.

Desenvolvimento

O desenvolvimento consiste em quatro fases:

1. Ler um arquivo .txt
2. Montar a matriz contida dentro do arquivo .txt
3. Executar o algoritmo genético para tentar encontrar a saída do labirinto
4. Executar o algoritmo estrela para achar a saída ideal do labirinto

Para as fases 1 e 2, criamos as classes Arquivo e Labirinto, que serão responsáveis por ler os arquivos de entrada e posteriormente montar em caráter de matrizes os labirintos contidos nos arquivos para que possamos manipula-las posteriormente. Para conseguirmos ler o .txt, usamos a biblioteca *BufferedReader* do Java para ler o arquivo e nos retornar uma matriz de *string*. Quando adquirida a matriz, passamos essa matriz para a classe Labirinto para que possamos manipulá-la durante a execução dos algoritmos de busca e refinamento.

A fase 3, desenvolvemos o algoritmo genético, com o intuito de descobrirmos a saída do labirinto imposto. Aqui usaremos as classes Labirinto, Coordenada e o AlgoritmoGeneticoCustom. A primeira classe precisamos para podermos saber qual a matriz que precisamos usar, o número máximo de genes que usaremos para cada cromossomo, o número máximo de cromossomos, esse para limitarmos a população ou o total de soluções, o máximo de gerações que o algoritmo executará e por fim um objeto da Coordenada que nos servirá como o ponto de entrada para o labirinto.

A definição da população inicial nós informamos anterior a execução do algoritmo genético, como um parâmetro da classe construtora do algoritmo genético, em vez de definição como um atributo global fixo do algoritmo. Como função heurística, calculamos durante a função de aptidão uma pontuação total sobre o caminho que vai ser percorrido pelo agente.

Definimos que o ponto que receberá o maior valor de pontuação será, naturalmente, a saída, caso o algoritmo consiga encontrar sua coordenada. Definimos um valor fixo para ele, que se sobressaí a todas outras posições no labirinto. Enquanto isso, avaliamos cada posição a ser visitada do labirinto. O agente aumentará sua pontuação somente se visitar um espaço válido (com valor no labirinto = “0” ou “S”, conforme mencionado e não foi visitado

anteriormente). Caso contrário, tentar sair dos limites do labirinto, revisitar um espaço já visitado ou bater numa parede (valor no labirinto = “1”), a pontuação será decrementada.

As taxas de mutação estabelecemos em 50%, com o cromossomo de elitismo “sobrevivendo” a mutação. Os cromossomos restante são selecionados randomicamente para cruzamento via torneio, onde usamos o valores maiores de pontuação do cromossomo como fator de corte.

A última fase, que consiste em desenvolver um algoritmo em estrela, usamos uma estrutura que consiste na matriz do labirinto e o id do nodo inicial e nodo final do grafo de busca do algoritmo, aspecto de busca bem diferente do que o algoritmo genético. E também usamos as classes de suporte Nodo e Celulas, para que possamos fazer o controle das células do grafo e visitação e manipulação de cada nodo visitado durante a busca pela saída. Como função heurística, usamos a distância Manhattan, pois era a heurística que conhecíamos melhor e tem uma boa performance para encontrarmos o próximo nodo nó ideal da nossa busca.

Em ambos os algoritmos, fazemos a mostra do caminho encontrado pelas populações dos cromossomos do algoritmo genético, indicando o total da função de aptidão de cada população e caso encontrado a saída, informamos o cromossomo e o caminho feito pelo cromossomo até a saída. No caso do algoritmo estrela, desenhamos o caminho buscado, percorrendo nodo a nodo até chegar na saída.

Encontramos alguns empecilhos durante o desenvolvimento deste trabalho. Primeiro, não fizemos uma definição dinâmica da pontuação da coordenada de saída no algoritmo genético devido à falta de tempo para elaborarmos uma pontuação ideal, apesar de algumas soluções terem sido pensadas durante a implementação do trabalho.