FACOM·UFMS

Inteligência Artificial - T01

Prof. Bruno M. Nogueira
Faculdade de Computação - UFMS

Trabalho I - Algoritmos de Busca

Neste primeiro trabalho, vocês devem, em grupos de no mínimo 3 e no máximo 4 pessoas, implementar algoritmos de busca para resolver os problemas destacados.

1 Problema do Jogo Resta Um

Resta um é jogo de tabuleiro cujo objetivo é, por meio de movimentos "pula-peça", deixar apenas uma peça no tabuleiro. No início do jogo, há 32 peças no tabuleiro, deixando vazia a posição central, tal como exibido na Figura 1.

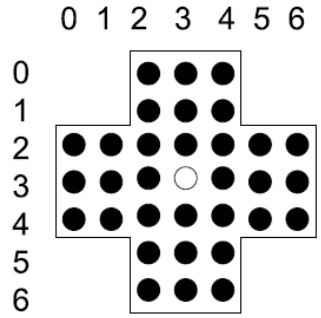

Figura 1: Configuração inicial do tabuleiro do jogo Resta Um.

Peças podem saltar sobre outras imediatamente na horizontal ou imadiatamente na vertical, desde que o salto termine em um espaço vazio. A peça que foi "saltada" é retirada do tabuleiro. O jogo termina quando não é mais possível fazer nenhum outro movimento. O jogador ganha se restar apenas uma peça no tabuleiro.

Implemente uma solução em Python para o problema do Resta Um, utilizando o algoritmo A*. Os estados serão representados por uma matriz numérica (lista de listas) de tamanho 6x6. Posições ocupadas serão representadas pelo valor 1, enquanto posições vazias serão ocupadas pelo valor 0. As posições fora do intervalo válido (as duas primeiras e duas últimas colunas para as duas primeiras e duas últimas linhas) deverão, sempre, ter valor 0 - não fazem parte do tabuleiro e, portanto, não podem ser ocupadas. Assim, o estado inicial do problema será, sempre, da forma:

```
[[0,0,1,1,1,0,0],[0,0,1,1,1,0,0], [1,1,1,1,1,1,1], [1,1,1,0,1,1,1], [1,1,1,1,1,1,1], [0,0,1,1,1,0,0], [0,0,1,1,1,0,0]]
```

Ao encontrar um estado final, deve-se listar, em ordem, todos os movimentos realizados, de maneira a identificar a transição de estado escolhida. Cada movimento deve ser identificado com $(X, Y) - (X', Y')$, em que o par (X, Y) determina a linha e a coluna em que se encontra a peça a ser movida, enquanto (X', Y') indicam a linha e a coluna de destino da peça movida. Linhas

e colunas são numeradas de 0 a 6, como mostrado na Figura 1. Por exemplo, um dos possíveis movimentos iniciais seria $(3,1) - (3,3)$, indicando a movimentação da peça da linha 3, coluna 1, para o espaço em branco na linha 3, coluna 3. Neste movimento, a peça localizada em $(3,2)$ foi pulada e, portanto, deve ser removida.

Você deverá reportar o tempo de execução e o número de estados expandidos (enfileirados) até encontrar a solução. A heurística a ser utilizada deverá ser desenvolvida pelo grupo. Seu programa deverá gerar um arquivo denominado "*saida-resta-um.txt*", que deverá conter todas as soluções encontradas pelo seu programa, seguindo EXATAMENTE o seguinte formato:

- Cada solução deve ser precedida da diretiva *==SOLUCAO*;
- Cada linha deve conter um movimento, com par ordenado da posição de origem - par ordenado da posição de destino da peça movimentada. Observem a existência de um espaço em branco antes e após o hífen;
- O movimento que leva ao estado final deve ser iniciado com a diretiva FINAL, seguido de um espaço em branco e a posterior sequência de pares identificando o movimento;
- As soluções devem ser separadas por exatamente uma linha em branco.

Abaixo, um exemplo do conteúdo esperado do arquivo de saída.

```
==SOLUCAO
(X1, Y1) - (X2, Y2)
...
FINAL (XN, YN) - (XK, YK)

==SOLUCAO
(X1, Y1) - (X2, Y2)
...
FINAL (XT, YT) - (XS, YS)
```

2 Problema de alocação de artigos

Conferências científicas baseiam a seleção de artigos a serem publicados em processos de revisão por pares. Nestes, cientistas proeminentes da área da conferência são convidados a contribuir, revisando artigos submetidos pelos autores, indicando a aprovação ou rejeição dos mesmos.

Um dos grandes problemas enfrentados pelos organizadores dos eventos, no entanto, consiste na atribuição dos artigos aos revisores. Em geral, um processo de votação é feito antes da atribuição, no qual os revisores são apresentados ao título e ao resumo dos artigos a serem avaliados e atribuem uma nota inteira de afinidade entre 0 (não desejo revisar, não conheço o tema do artigo) e 5 (desejo muito revisar, sou especialista na área do artigo). Além disso, os revisores indicam quantos artigos, no máximo, gostariam de revisar para este evento. O grande desafio consiste em atribuir os artigos recebidos aos melhores revisores, buscando sempre respeitar os limites de cada colaborador.

Elabore uma solução de atribuição de artigos baseada em algoritmos genéticos, implementada em Python, para auxiliar este processo. Sua solução deverá atribuir artigos de maneira a maximizar a afinidade revisor / artigo da distribuição. Nenhum artigo pode ficar sem revisão e deve-se, sempre, respeitar o máximo de artigos que um revisor pode receber.

Seu algoritmo deve receber como entrada uma matriz $NxM + 1$, lida a partir de um arquivo textual. Nesta matriz, N é o número de revisores cadastrados e M é o número de artigos a serem atribuídos. Ao final de cada linha, estará expresso o máximo de artigos que o revisor aceita receber (sempre maior ou igual a 1). Abaixo, um exemplo de entrada:

```
0,0,3,4,4,1
3,3,0,0,1,2
4,0,0,1,0,1
2,2,2,3,2,2
```

Neste exemplo, 5 artigos foram recebidos e 4 revisores estão disponíveis. O revisor 1 tem afinidade 0 com os artigos 1 e 2; afinidade 3 com o artigo 3; afinidade 4 com os artigos 4 e 5; e aceita receber, no máximo, 1 artigo para revisar.

Na sua solução, deve-se ler uma matriz a partir de um arquivo de entrada, no formato $NxM+1$, como a de exemplo apresentada anteriormente. A codificação genética dos estados deve ser definida por você. Você deve inicializar os indivíduos aleatoriamente. Por isso, 10 repetições devem ser feitas para um mesmo problema. Você deverá utilizar os operadores de seleção, mutação e *crossover*. Devem ser parâmetros do seu algoritmo a taxa de *crossover* (com o nome *crossoverrate*), a taxa de mutação (com o nome *mutationrate*), o máximo de gerações a serem desenvolvidas (com o nome *maxgen* e valor padrão de 100) e o caminho para o arquivo de entrada da matriz (com o nome *inputpath*). A função de *fitness* a ser utilizada também deve ser definida por você, devendo ser aplicada para o processo de seleção por roleta.

Seu programa deverá gerar um gráfico da evolução da função de fitness ao longo das gerações (gráfico de linha, com o nome "*fitness.png*"). Duas linhas devem estar no gráfico, uma para a melhor solução e outra para a média das 10 repetições.

Também, deve ser gerado um arquivo "*saida-genetico.txt*", reportando o resultado obtido ao final da execução que, dentre as 10 repetições, teve a melhor função de *fitness*. Nesta saída, uma única linha de tamanho M deve ser gerada, indicando o índice do revisor atribuído a um determinado artigo. Abaixo, um exemplo de saída para o problema de entrada reportado anteriormente.

```
3,2,1,4,4
```

Neste exemplo, o artigo 1 foi atribuído ao revisor 3; o artigo 2 ao revisor 2; o artigo 3 ao revisor 1; o artigo 4 ao revisor 4; e o artigo 5 também ao revisor 4.

3 Entregas

Para estes problemas, deve-se entregar soluções codificadas em Python. Para o primeiro problema, deve-se denominar o script principal de *restaUm.py*. Para o segundo problema, o script principal deve ser denominado *alocacaoArtigos.py*. Todos os scripts devem ter no cabeçalho o nome e o RGA de cada membro do grupo, bem como comentários ao longo do código que possam facilitar o entendimento do mesmo.

Junto ao código, deve ser entregue um relatório no formato PDF, de tamanho entre duas a quatro páginas. Este relatório deve estar identificado com o nome de todos os integrantes do grupo e deve, obrigatoriamente, apresentar:

1. Uma explicação das heurísticas utilizadas (funções de custo, avaliação e fitness). As funções devem ser aplicáveis e condizentes com os problemas. Deve-se mostrar matematicamente como são expressas as funções escolhidas, bem como explicar a intuição que motivou a escolha da mesma.

2. Para o primeiro problema, deve-se reportar tempo de execução e número de estados expandidos. É desejável, mas não obrigatório, que mais de uma heurística seja adotada, para fins de comparação. Gráficos e tabelas, com as respectivas explicações, devem ser apresentados no relatório.

3. Para o segundo problema, deve-se testar, pelo menos, três combinações de parâmetros diferentes, reportando os resultados obtidos utilizando os gráficos de evolução do fitness ao longo

das iterações. Também deve ser mensurado o tempo de execução e o número de iterações levado até a convergência (se ocorrer). Também, é obrigatório o uso de gráficos e tabelas, com as respectivas explicações, para reportar os valores dos resultados.

O prazo para entrega do trabalho será as **23:55** do dia **05/05/2019**. Deverão ser entregues todos os scripts com os códigos das soluções, bem como o relatório em formato PDF. Todos os arquivos devem ser compactados em um único arquivo (.rar ou .zip). Uma única entrega por grupo deve ser feita, via Moodle (EAD). Entregas fora do prazo, ou feitas por outros meios, serão desconsideradas.

O professor consultará os alunos para que os integrantes apresentem os códigos. Inicialmente, tal apresentação está marcada para o dia 07/05/2019, podendo ser ajustada a critério do professor, em acordo com os alunos. Todos os membros deverão estar presentes no momento da apresentação, quando um dos integrantes será sorteado como apresentador e cabendo a ele, e somente ele, as explicações solicitadas pelo professor. A ausência de algum membro implicará em nota zero ao mesmo.

ATENÇÃO: Os códigos serão submetidos a análise de plágio em sistemas próprios para isso. Não será tolerado plágio de quaisquer fontes, mesmo que parcial. Quando detectado plágio, o trabalho terá nota zero.

Havendo qualquer dúvida, consulte o professor.

Bom trabalho!