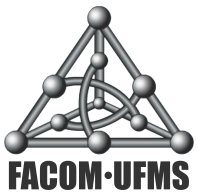

Inteligência Artificial

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 Problemas de busca tradicionais

- **Missionários e canibais**: inicialmente, tem-se 3 missionários e 3 canibais em uma margem do rio. O objetivo é levar todos os missionários e todos os canibais para a outra margem do rio utilizando um bote com capacidade para duas pessoas. Deve-se ter sempre, pelo menos, uma pessoa dentro do bote quando da transição de uma margem para a outra. Além disso, não se pode ter mais canibais do que missionários em qualquer uma das margens do rio.
  - **Representação dos estados**: uma lista no formato $[X, Y, Z]$, na qual: X é o número de missionários na margem esquerda, Y é o número de canibais na margem esquerda e Z é o número de botes na margem esquerda.
  - **Estado inicial**: $[3, 3, 1]$.
  - **Estado final**: $[0, 0, 0]$.
- **Potes de d'agua**: dispõe-se de dois recipientes de água, um com capacidade de 7 litros e outro com capacidade de 5 litro. Inicialmente, ambos estão vazios. É necessário encher um dos recipientes com 4 litros, sendo que as únicas operações possíveis são encher (existe uma fonte externa); esvaziar completamente um dos recipientes; ou utilizar o conteúdo de um recipiente para encher outro.
  - **Representação dos estados**: uma lista no formato [X; Y], na qual: X é o volume ocupado no primeiro pote e Y é o volume ocupado no segundo pote.
  - **Estado inicial**: [0; 0].
  - **Estado final**: qualquer estado em que se tenha o valor 4 em X ou Y.

Para todos estes problemas, deve-se implementar soluções utilizando os seguintes algoritmos: profundidade, largura e *A**. Para os algoritmos que utilizam funções de avaliação e / ou função de custo, o grupo deve propor funções que sejam aplicáveis e condizentes. Escolhas ruins serão penalizadas, mesmo que o algoritmo esteja correto.

Em todos estes algoritmos, deve-se imprimir mensagens mostrando, a cada passo: a fila de espera, o nó visitado e quais nós foram enfileirados. Quando uma solução for encontrada, o

algoritmo deve imprimir o conjunto de estados percorridos, desde o nó inicial até o nó final. Deve-se continuar a busca até que a fila de candidatos esteja vazia. Deve-se seguir o padrão de mensagens fornecidos no script "potes.py", fornecido pelo professor. Quando o algoritmo implementado usar uma função heurística, o valor da mesma deve ser impressa junto ao nó. Por exemplo: $[1, 2, 3], 2.5$, mostra que o nó $[1, 2, 3]$ tem função heurística calculada em 2.5. Além disso, as funções dos algoritmos de busca devem receber como parâmetro o estado inicial da busca (obrigatório) e o estado final (obrigatório apenas quando o estado final for único).

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 os problemas tradicionais, deve-se denominar os scripts principais de *missionarios.py* e *potes.py*, respectivamente. Para o problema da alocação de artigos, 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 os problemas tradicionais, 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 **03/05/2020**. 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 (AVA). 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 05/05/2019, 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. Caso ainda estejamos em período EaD, a apresentação será por videoconferência.

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!