Inteligência Artificial

Primeiro Trabalho - Algoritmos de Busca

Profa. Heloisa

2016/2

Este trabalho consta da execução dos algoritmos de busca desinformada (busca em largura, busca em profundidade, busca uniforme) e informada (busca gulosa, algoritmo A) estudados na disciplina, com o uso das implementações em R disponibilizadas no repositório basicAI_Search e descritas neste documento.

Os arquivos disponíveis contém implementações gerais de todos os algoritmos, que retornam o caminho (solução) encontrado durante o processo de busca. Os códigos devem ser complementados com definições específicas do problema a ser tratado, como: representação dos estados, operadores de mudança de estado, função heurística.

Um exemplo de implementação desses elementos, específicos para o problema dos Missionários e Canibais, foi incluído no repositório para auxiliar a compreensão e definição exigida para a correta execução dos algoritmos.

Os algoritmos devem ser aplicados ao problema do Mundo do aspirador de pó, definido a seguir. A função heurística utilizada nos algoritmos de busca informada pode ser da escolha de cada grupo.

Mundo do Aspirador de Pó - Um cenário é representado por uma grade de 2X2, sendo que cada quadrado pode ter ou não sujeira. Um aspirador de pó pode aspirar sujeira ou se mover nesse cenário, com os seguintes operadores: 1) aspirar sujeira do quadrado em que se encontra; 2) mover para o quadrado da direita; 3) mover para o quadrado de baixo; 4) mover para o quadrado da esquerda; 5) mover para o quadrado de cima. Esses operadores têm custos diferentes que são: aspirar a sujeira - custo 2; andar para a esquerda ou para a direita - custo 1; andar para cima ou para baixo - custo 3. Encontre a sequência de movimentos para chegar a um estado onde todos os quadrados estão limpos, a partir de uma situação inicial com pelo menos 2 quadrados sujos.

Relatório

O trabalho deve ser acompanhado de um relatório que documente claramente como o problema foi formulado e como os itens dessa formulação foram definidos e representados na implementação. Os principais itens que devem ser descritos são: a representação do estado, os operadores aplicáveis aos estados (ações), a função heurística, o estado inicial e o estado objetivo.

Observações

- O trabalho pode ser feito em duplas;
- Entregar (no moodle – tarefa de arquivo único):
  - Relatório, com o conteúdo detalhado anteriormente;
  - Código específico definido para o problema;
  - Saídas de todos os algoritmos (gerado pela implementações do repositório);
- **DATA DE ENTREGA**: 13/01/2017.

Implementações Disponíveis

O repositório basicAI_Search contém os códigos em R que serão necessários para a execução do trabalho de Inteligência Artificial sobre buscas.

São necessários os arquivos `Estado.R`, `buscaDesinformada.R` (algoritmos busca em largura, profundidade e custo uniforme) e `buscaInformada.R` (algoritmos greedy e A*).

Seu trabalho corresponde à implementação dos métodos genéricos definidos em `Estado.R` e Sobrecarga do operador `==`, que define se dois estados são iguais, para o problema específico do **Mundo do Aspirados de Pó**.

Missionários e Canibais

O arquivo `Canibais.R` traz uma implementação para o problema de 3 Missionários e 3 Canibais, definindo a geração de novos estados considerando um conjunto de operadores e a avaliação do valor de heurística para esse problema.

Um script com a instanciação do problema dos Missionários e Canibais e execução dos algoritmos de busca também está disponível no repositório (arquivo `exemploCanibais.R`).

Inicialização

Definição dos nós inicial e objetivo:

```
inicial <- Canibais(desc = c(M = 3, C = 3, B = 1))

objetivo <- Canibais()
objetivo$desc <- c(M = 0, C = 0, B = 0)
```

Busca Desinformada

Para executar a busca em largura, chamamos a função `buscaEmLargura(inicial, objetivo)`, que retorna o caminho obtido do nó inicial ao nó objetivo, como um objeto do tipo lista:

```
buscaEmLargura(inicial, objetivo)
```

```
## [[1]]
## (M C B): ( 3 3 1 )
## G(n):  0
## H(n):  Inf
## F(n):  Inf
##
## [[2]]
## (M C B): ( 3 1 0 )
## G(n):  1
## H(n):  4
## F(n):  Inf
##
## [[3]]
## (M C B): ( 3 2 1 )
## G(n):  2
## H(n):  6
## F(n):  Inf
##
```

```
## [[4]]
## (M C B): ( 3 0 0 )
## G(n):  3
## H(n):  3
## F(n):  Inf
##
## [[5]]
## (M C B): ( 3 1 1 )
## G(n):  4
## H(n):  5
## F(n):  Inf
##
## [[6]]
## (M C B): ( 1 1 0 )
## G(n):  5
## H(n):  2
## F(n):  Inf
##
## [[7]]
## (M C B): ( 2 1 1 )
## G(n):  6
## H(n):  4
## F(n):  Inf
##
## [[8]]
## (M C B): ( 1 0 0 )
## G(n):  7
## H(n):  1
## F(n):  Inf
##
## [[9]]
## (M C B): ( 2 0 1 )
## G(n):  8
## H(n):  3
## F(n):  Inf
##
## [[10]]
## (M C B): ( 0 0 0 )
## G(n):  9
## H(n):  0
## F(n):  Inf
```

Para outras implementações de busca desinformada, a chamada de função é semelhante.

Busca Informada

Para a busca informada, o algoritmo utilizado é o *best first*. A execução do *best first* exige um a passagem de um valor para o parâmetro `abordagem`, além dos nós inicial e objetivo. O valor do parâmetro deve ser uma string indicando qual o cálculo da função de avaliação deve ser utilizado: como definido para o algoritmo "Greedy" ou como definido para o algoritmo "AEstrela".

A string "AEstrela" foi definida como padrão, então se não for passado valor para o parâmetro `abordagem`, o *best first* utilizará a função de avaliação definida para o A*.

```
buscaBestFirst(inicial, objetivo, "AEstrela")
```

```
## [[1]]
## (M C B): ( 3 3 1 )
## G(n):  0
## H(n):  Inf
## F(n):  Inf
##
## [[2]]
## (M C B): ( 3 1 0 )
## G(n):  1
## H(n):  4
## F(n):  5
##
## [[3]]
## (M C B): ( 3 2 1 )
## G(n):  2
## H(n):  6
## F(n):  8
##
## [[4]]
## (M C B): ( 2 1 0 )
## G(n):  3
## H(n):  3
## F(n):  6
##
## [[5]]
## (M C B): ( 2 2 1 )
## G(n):  4
## H(n):  5
## F(n):  9
##
## [[6]]
## (M C B): ( 2 0 0 )
## G(n):  5
## H(n):  2
## F(n):  7
##
## [[7]]
## (M C B): ( 3 0 1 )
## G(n):  6
## H(n):  4
## F(n):  10
##
## [[8]]
## (M C B): ( 1 0 0 )
## G(n):  7
## H(n):  1
## F(n):  8
##
## [[9]]
## (M C B): ( 2 0 1 )
## G(n):  8
## H(n):  3
```

```
## F(n):  11
##
## [[10]]
## (M C B): ( 0 0 0 )
## G(n):  9
## H(n):  0
## F(n):  9
```

Nota: Vale lembrar que o problema com 3 Missionários e 3 Canibais possui uma árvore de busca pequena e simples, então o caminho retornado para todos os algoritmos de busca é o mesmo.