Trabalho 1 - MC906A

AKARI ISHIKAWA (163282) *

*Ciência da Computação - Graduação
E-mail: ueda.aka@gmail.com

***Resumo* – Neste projeto, modelamos um agente solucionador de problemas com a tarefa de se locomover de um ponto a outro num mapa com 3600 pontos. A modelagem foi feita transformando os estados em que o agente se encontra em nós de um grafo cujos nós vizinhos eram as ações que o agente poderia tomar. Realizamos análises com 4 e 8 ações. Aplicamos então os algoritmos de Breadth First Search, Depth First Search e A* e os comparamos sob o ponto de vista de tempo de execução, nós visitados e qualidade da solução encontrada.**

***Palavras-chave* – Agente solucionador de problemas, Buscas, Modelagem de agentes**

I. INTRODUÇÃO

Dentro da Inteligência Artificial (IA), Russel e Norvig [1] definiram um agente inteligente como uma entidade que toma ações buscando um objetivo e que possui sensores para perceber um ambiente e atuadores para interagir com esse ambiente. Este conceito abstrato de agente pode ser utilizado para modelar inúmeras aplicações dentro da IA.

Quando temos um agente baseado em realizar um objetivo, umas das formas clássicas de modelar o agente é buscando uma solução dentro da árvore de ações possíveis que ele pode tomar. Métodos baseados em análise da árvore de ações são chamados de métodos de busca e são divididos entre Busca-Não-Informada e Busca-Informada. Na Busca-Não-Informada, o agente apenas tem a informação de qual objetivo quer atingir, enquanto que na Busca-Informada ele tem outras informações que permitem ele realizar escolhas possivelmente melhores.

A proposta desse projeto é implementar um agente que possui o único objetivo de se deslocar de um ponto inicial (xi, yi) a um nó-meta (xf, yf) e realizar experimentos com duas formas de busca não informada e uma busca informada com duas heurísticas distintas.

Neste relatório explanaremos detalhes sobre a especificação, a modelagem proposta, os experimentos e discussão sobre os resultados obtidos.

II. PROCEDIMENTOS EXPERIMENTAIS

A. Modelagem do Estado

No problema proposto, o agente se encontra no mapa representado pela figura 1 na posição inicial indicada pelo ponto vermelho e seu objetivo é chegar na posição indicada pelo ponto verde.

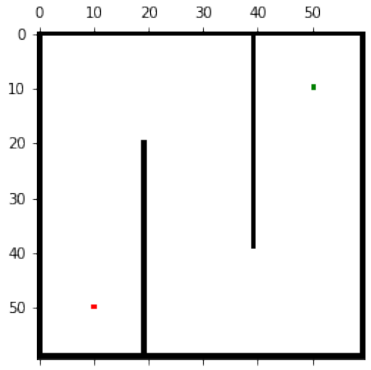


Figura 1. Mapa do ambiente proposto. Ponto vermelho indicando a posição inicial em (50,10) e ponto verde indicando o nó meta em (10,50).

Ao discretizarmos o espaço, o mapa foi transformado em uma matriz de 60x60 (3600 pontos), onde os pontos pretos são bordas (obstáculos) e os pontos brancos são livres para o agente se locomover. Determinamos a coordenada (50,10) como a posição inicial e (10,50) como o nó-meta.

Apesar de ser possível discretizar o espaço com um número maior de pontos, acreditamos que 3600 são suficientes para analisar o problema sem ter limitações causadas por custo computacional excessivo.

Por fim, para a execução dos algoritmos de busca, o estado do ambiente foi modelado como um grafo. Cada nó v representa um estado que o agente pode se encontrar. As arestas (v, u) representam as ações que o agente toma estando no estado v para chegar ao estado u.

B. Ações

Neste projeto, testamos dois tipos de conjuntos de ações para o agente:

- No estado (x, y), o agente pode se locomover para as quatro direções a 90º.
- No estado (x, y), o agente pode se locomover para as oito direções a 90º e 45º.

Em ambos os casos, o agente só pode se locomover para outra posição caso ela seja livre, ou seja, se ela não for um ponto de borda.

É importante enfatizar que a ordem de geração dos vizinhos para cada nó foi dada na seguinte ordem: *norte, sul, oeste, leste, noroeste, nordeste, sudoeste e sudeste*, sendo os quatro últimos apenas para o agente com oito ações. Essa ordem é importante pois influencia na solução encontrada em alguns algoritmos de busca, principalmente o *Depth First Search*.

C. Posição Inicial e Teste Objetivo

Em todos os experimentos, os algoritmos de busca foram inicializados a partir do nó correspondente à posição inicial (50,10) e, a cada nó visitado, foi feito um teste de objetivo verificando se a tarefa foi cumprida. Em nosso caso, o teste verifica se já estamos em (10,50). Se sim, o algoritmo retorna aquele caminho como uma solução.

D. Algoritmos de Busca

Neste projeto, analisamos a solução do problema utilizando os seguintes algoritmos:

- Breadth First Search: Possui complexidade de tempo estimada em $O(b^d)$, sendo b o número médio de vizinhos por nó (i.e., 4 ou 8 em nosso caso) e d a profundidade da solução.
- Depth First Search: Possui complexidade de tempo estimada em $O(b^m)$, sendo b o número médio de vizinhos por nó (i.e., 4 ou 8 em nosso caso) e m a profundidade máxima de uma sequência de expansões.
- A* com heurística $f(n) = g(n) + h(n)$, onde $g(n)$ é o custo de se chegar a n a partir do nó inicial e $h(n)$ é a distância euclidiana entre o n e o nó-meta.
- A* com heurística $f(n) = g(n) + h(n)$. Para este algoritmo mantivemos $g(n)$ como o custo de se chegar a n a partir do nó inicial. Para determinar $h(n)$, fizemos o seguinte procedimento: simulamos a execução de um agente saindo do nó-meta buscando chegar ao ponto inicial. A cada nó n que o agente original está, ele terá o agente-simulado no nó u. Para cada vizinho m de n, calculamos a distância euclidiana (m, v) para cada vizinho v de u. Definimos então $h(n)$ como a menor distância (m, v) encontrada. Ambos os algoritmos de A* possuem complexidade média de $O(b^d)$ como a Breadth First Search. Porém, devemos embutir uma constante α do cálculo da heurística a cada expansão, levando a uma complexidade de $O(\alpha b^d)$. Sendo o α da segunda heurística significativamente maior do que a primeira.

Todas as implementações foram **copiadas e adaptadas** das implementações em Python dos exercícios do livro *Artificial Intelligence: A Modern Approach* [1] disponibilizado em https://github.com/aimacode/aima-python.

O código fonte deste projeto pode ser acessado no github da autora (https://github.com/AkariUeda/mc906-lab1/blob/master/mc906-lab1.ipynb).

III. Resultados

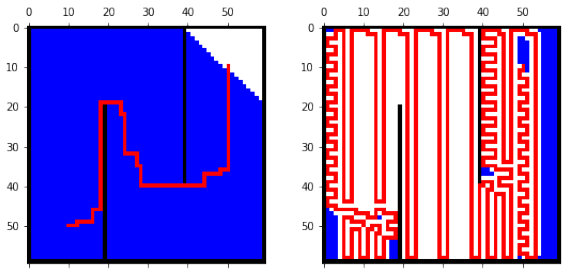
Para cada execução de um algoritmo de busca, coletamos os nós visitados, a primeira solução encontrada e o tempo de execução.

Separamos a seção de resultados em duas seções: os resultados dos experimentos com os nós com 4 vizinhos, ou seja, onde o agente podia ir apenas para a direita, esquerda, para cima e para baixo e os experimentos em que o agente podia ir para todas as 8 posições vizinhas.

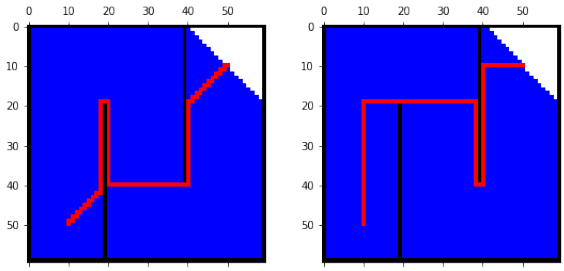
A. Quatro ações

Na figura 2, representamos a execução de cada um dos algoritmos. Representamos em azul todos os nós que foram buscados mas que não levavam a uma solução e denotamos em vermelho a **primeira** solução encontrada pelo algoritmo.

Os algoritmos de Breadth First Search e A* com as duas heurísticas encontraram o caminho mínimo, com 123 nós. Já a Depth First Search encontrou uma solução com 1111 nós.



(a) Breadth First Search (b) Depth First Search



(c) A* heurística 1 (d) A* heurística 2

Figura 2. Representação da execução dos quatro tipos de buscas realizadas para o robô com quatro ações. A região em azul indica os nós visitados e a solução encontrada foi indicada em vermelho.

Na figura 3 podemos ver os tempos de execução para cada uma das buscas. Vemos que a Breadth First Search apresenta um tempo de execução significativamente menor em relação às outras buscas. O resultado é condizente com o que estavamos esperando dada as suas complexidades de execução detalhadas na seção II-D. É importante notar também a diferença nos tempos de execução entre os dois algoritmos de A*. A heurística adotada no segundo A* é mais custosa computacionalmente, levando a um tempo de execução significativamente maior.

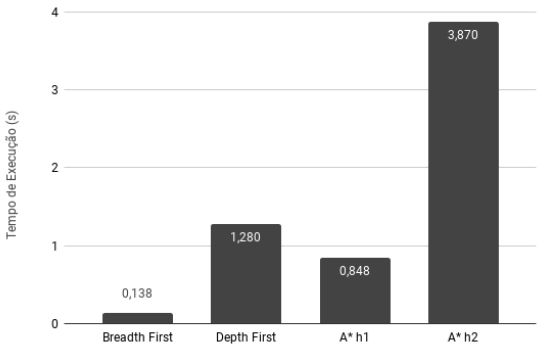


Figura 3. Comparação dos tempos de execução nos algoritmos executados com 4 ações para cada estado

Um fato interessante que vale a pena ser mencionado é a quantidade de nós visitados por cada um dos algoritmos (figura 4). A Breadth First Search e os A* apresentaram quantidades semelhantes de visitações enquanto que a Depth First Search visitou menos da metade dos nós visitados pelos outros algoritmos. Isso se deve ao fato da Depth First Search sempre procurar uma solução dentro de um caminho até eventualmente encontrar ou falhar, diminuindo a quantidade de nós que seriam visitados sem que façam parte da solução.

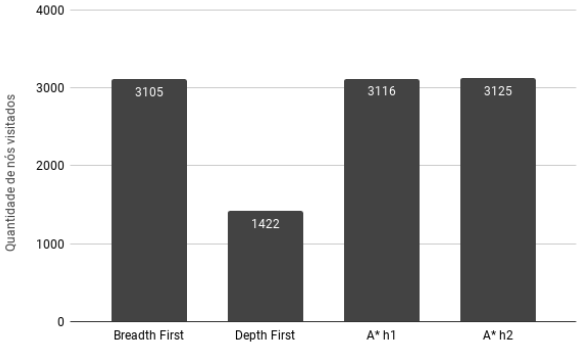
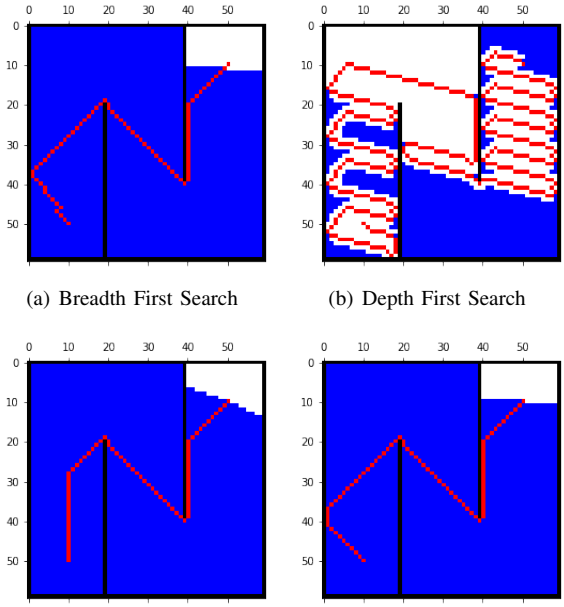


Figura 4. Comparação da quantidade de nós visitados nos algoritmos executados com 4 ações para cada estado

B. Oito ações

Apesar de já ter obtido uma primeira visão geral sobre os algoritmos, queríamos com esses novos experimentos analisar eventuais mudanças em suas execuções ao dobrarmos a quantidade de ações por estado.

Ao compararmos as soluções encontradas por cada algoritmo (figura 6), notamos que a Breadth First Search a os A* ainda obtêm a solução ótima. No entanto, nesta configuração, a solução mínima possui 83 nós. A Depth First Search obteve uma solução com 394 nós.



(a) Breadth First Search (b) Depth First Search

(c) A* heurística 1 (d) A* heurística 2

Figura 5. Representação da execução dos quatro tipos de buscas realizadas para o robô com oito ações. A região em azul indica os nós visitados e a solução encontrada foi indicada em vermelho.

A análise dos tempos de execução foi a que apresentou mais diferenças. A Breadth First Search, a Depth First Search e o A* com a primeira heurística tiveram seus tempos de execução duplicados, resultado condizente com o fato de suas ações terem sido duplicadas. No entanto, o A* com a segunda heurística apresentou agora um tempo de execução de 25,6 segundos, um grande aumento comparado com os 3,8 segundos do experimento anterior.

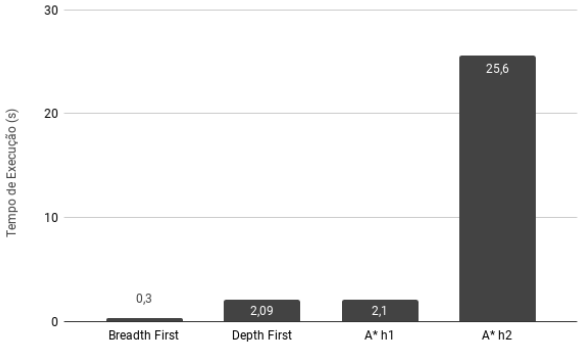


Figura 6. Comparação dos tempos de execução nos algoritmos executados com 8 ações para cada estado

A quantidade de nós visitados se manteve similar ao do experimento anterior. Este fato também é esperado, dado que da forma que o mapa foi discretizado, visitaremos a mesma quantidade de nós andando uma distância x não importando o sentido do movimento. Isto é, andar uma distância x na diagonal visita a mesma quantidade de nós do que andar uma distância x na vertical ou horizontal.

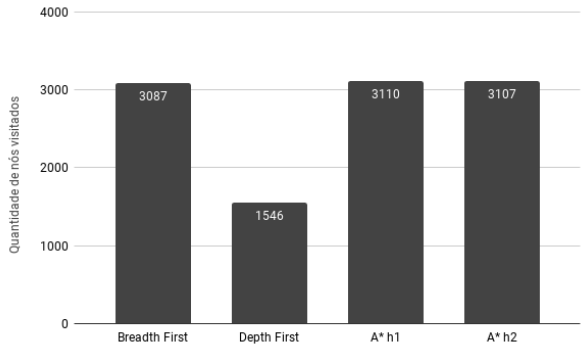


Figura 7. Comparação da quantidade de nós visitados nos algoritmos executados com 8 ações para cada estado

IV. Conclusão

Neste projeto, estressamos a aplicação dos algoritmos de Breadth First Search, Depth First Search e A* para modelar um agente que deveria ir de um ponto ao outro em um mapa discretizado em 3600 pontos com alguns obstáculos.

Apesar de todos os três algoritmos apresentarem completude, isto é, todos são capazes de encontrar uma solução caso ela exista, pudemos perceber que os algoritmos Breadth First Search e A* também possuem optimicidade, propriedade de algoritmos de garantirem que a solução encontrada será ótima.

Por fim, dentre todos os métodos analisados, a Breadth First Search foi a que demonstrou não só completude e optimicidade mas também o menor tempo de execução (0,138 segundos para o agente com quatro ações).

Acreditamos que apesar do A* ser um algoritmo mais robusto para problemas de busca, a tarefa deste projeto era simples o suficiente para ser solucionado com um algoritmo menos complexo.

Para trabalhos futuros, podemos também propor novas heurísticas menos custosas e mais adequadas para o problema proposto para obter melhores performances do A*.

Referências

[1] S. Russel and P. NORVIG, “Inteligência artificial. 3a edição,” *Editora Campus*, 2013. 1, 2