Resolução de Labirinto por Meio de Busca

1st Matheus dos Santos Wogt
Ciências da Computação
Instituto Federal Catarinense
Videira, Brasil
matheusnotas07@gmail.com

***Abstract*—Este trabalho apresenta a implementação e análise de quatro algoritmos de busca — Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e A* — aplicados à navegação em um labirinto composto por 21 salas conectadas por portas. O objetivo foi comparar o desempenho dos algoritmos em termos de custo do caminho e tempo de execução. Os experimentos mostraram que BFS, Busca Gulosa e A* retornaram caminhos equivalentes em custo, sendo a Busca Gulosa a mais eficiente em tempo de execução. O DFS apresentou o caminho de maior custo, apesar de menor tempo de computação, evidenciando que seu desempenho depende da estrutura do labirinto. Os resultados destacam a eficiência de algoritmos heurísticos, como Busca Gulosa e A*, na obtenção de caminhos equilibrados em custo e tempo, enquanto métodos não heurísticos podem ser mais previsíveis ou rápidos dependendo da topologia do ambiente.**

***Index Terms*—Busca em Largura, BFS, Busca em Profundidade, DFS, Busca Gulosa, Busca A*, Resolução de Labirinto**

I. Introdução

A navegação em ambientes desconhecidos ou complexos é um dos grandes desafios enfrentados por sistemas autônomos, como robôs móveis. A capacidade de encontrar um caminho eficiente de um ponto a outro, respeitando as restrições impostas pelo ambiente, é fundamental para o desempenho desses sistemas. Nesse contexto, o problema de busca em labirintos torna-se uma excelente representação prática de situações enfrentadas por agentes inteligentes.

O presente trabalho aborda a implementação de algoritmos de busca para resolver o problema de navegação em um ambiente representado por 21 salas, identificadas pelas letras de A a U, conectadas por portas e separadas por paredes, conforme ilustrado na Fig. 1.

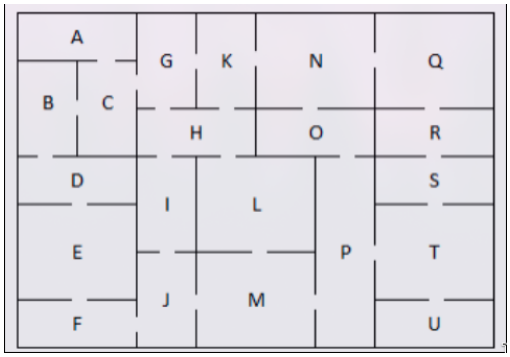

Fig. 1. Representação visual do labirinto

Identify applicable funding agency here. If none, delete this.

O objetivo deste trabalho é a criação de um programa cujos input's serão a sala inicial e a final, assim calculando a rota usando quatro métodos sendo eles Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e Busca A*, para os que utilizam heurística será utilizado a Distância de Manhattan. Ao final da execução será apresentado o caminho encontrado por cada algoritmo, não sendo analisado qual foi o melhor, deixando isso ao encargue do usuário.

II. Fundamentação Teórica

A resolução de problemas por meio de algoritmos de busca constitui um dos pilares da Inteligência Artificial, sendo aplicável a cenários como a navegação em labirintos. Entre os métodos mais estudados, destaca-se a Busca em Largura (BFS), que expande primeiramente o nó raiz, depois todos os seus sucessores imediatos, e assim sucessivamente em camadas. Essa estratégia utiliza uma fila do tipo FIFO, garantindo que os nós mais rasos sejam sempre explorados antes dos mais profundos, o que assegura completude e a obtenção de soluções mais curtas em termos de número de passos, quando os custos são uniformes [1, p. 101–102].

Já a Busca em Profundidade (DFS) segue pelo caminho mais profundo da árvore de busca, retrocedendo apenas quando não é mais possível prosseguir. Embora seja eficiente em termos de uso de memória, pode se perder em caminhos infinitos ou muito longos sem atingir o objetivo. Como alternativa, pode-se adotar versões modificadas, como a profundidade limitada ou o aprofundamento iterativo, que combinam aspectos positivos da BFS e da DFS [1, p. 108–109].

A Busca Gulosa de Melhor Escolha é um método que se baseia exclusivamente em informações heurísticas. Ela seleciona para expansão o nó que aparenta estar mais próximo do objetivo, avaliando os estados apenas pela função f(n)=h(n), onde h(n) representa a estimativa da distância até a meta. Apesar de, em muitos casos, encontrar soluções rapidamente, essa abordagem não garante a optimalidade, uma vez que pode ser induzida ao erro por heurísticas locais [1, p. 111–112].

Por sua vez, o algoritmo A* combina as vantagens da busca de custo uniforme e da busca gulosa, utilizando a função de avaliação f(n)=g(n)+h(n), em que g(n) corresponde ao custo acumulado do caminho até o nó atual e h(n) representa a estimativa do custo restante até o objetivo. Quando a heurística é admissível, ou seja, não superestima o custo real, A* é completo e ótimo. Em problemas de navegação em labirintos, uma

heurística comumente adotada é a Distância de Manhattan, definida como a soma das diferenças horizontais e verticais entre o estado atual e o estado objetivo [1, p. 94, 99, 113–115].

III. Metodologia

O ambiente considerado consiste em um labirinto com 21 salas identificadas pelas letras de A a U, conectadas por portas e separadas por paredes, conforme ilustrado na Fig. 1. Cada porta entre duas salas possui um valor associado que representa o custo de transição de uma sala para outra.

Para resolver o problema foi utilizado a linguagem C++, e para representar tal ambiente foi escolhido utilizar grafos. A representação de um quarto foi feita utilizando classe nomeada de **room**, na tabela 1.

TABLE I
ESTRUTURA DA CLASSE ROOM

Atributo	Tipo
name	char
neighborns	std::map<room*, int>
visited	bool

O labirinto foi representado em uma arte ASCII que foi utilizada para definir os pesos da Heurística de Manhattan e dos pesos de cada porta, na Fig. 2. podemos ver que cada elemento (porta, sala ou parede) corresponde a um caractere em uma posição específica.

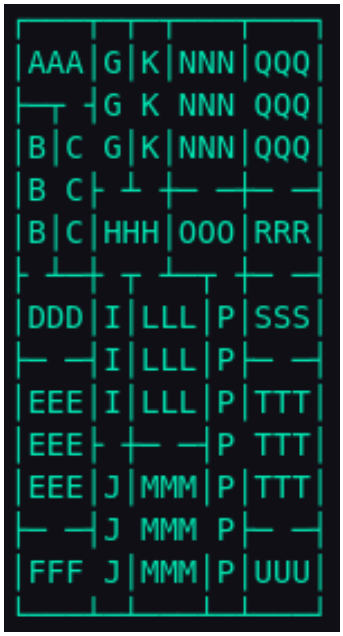

Fig. 2. Representação em ASCII do labirinto

O Cálculo do peso e uma porta foi feito contando quantos caracteres é necessário para ir do centro da sala até o centro da sala através essa porta, como por exemplo ir da sala A até a sala C custa 4 pois é necessário, partindo do centro da sala A, andar para a direita uma vez e para baixo 3 vezes contabilizando 4. Para calcular a Heurística de Manhattan foi utilizado posições x e y, como em um plano cartesiano, e em cada sala sendo atribuído apenas ao seu centro, por exemplo na sala A a posição ela é o caractere do meio tendo a posição x de 2 e y de 1, e para calcular a heurística foi utilizado a seguinte equação:

$$h(n) = |x_{\text{atual}} - x_{\text{objetivo}}| + |y_{\text{atual}} - y_{\text{objetivo}}| \quad (1)$$

Onde x_{atual} e y_{atual} é o endereço da sala inicial enquanto $x_{objetivo}$ e $y_{objetivo}$ da sala final, esta equação resulta na distância da Heurística de Manhattan.

A. Algoritmos

Os algoritmos BFS e DFS foram feitos conforme suas representações básicas como podemos ver respectivamente na Fig. [**?**]. e Fig. [**?**].

A busca gulosa foi implementada usando a Heurística de Manhattan como dita a sua implementação, e a busca A* (Aestrela) utilizando tanto a Heurística de Manhattan como a distância real.

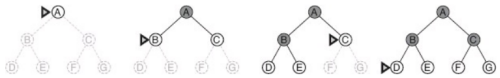

Fig. 3. Representação visual da busca em largura

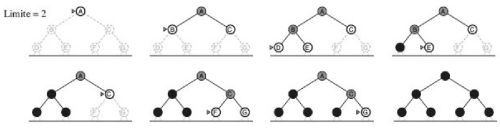

Fig. 4. Representação visual da busca em profundidade

IV. Resultados e Discussão

Para a obtenção dos seguintes resultados foi realizado uma busca do caminho partindo da sala A até a sala U, os 4 algoritmos retornaram um caminho e entre eles apenas o DFS retornou um caminho diferente e mais longo.

Os algoritmos BFS, busca gulosa e busca A* retornaram o caminho:

$$A->C->G->H->L->M->P->T->U$$

Sendo o custo dele de 33, porém o que os diferenciou foi o tempo que cada um demorou, como podemos ver na Tabela 2. endo a busca gulosa o que mais se destacou pelo seu menor tempo de execução.

TABLE II
ESTRUTURA DA CLASSE ROOM

Algoritmo	Tempo (ms)
Busca Gulosa	0.021177
BFS	0.024539
A*	0.061378

Enquanto o DFS em 0.020232 ms resultou no caminho de peso 53:

$$A \to C \to B \to D \to E \to F \to J \to I \to H \to G \to K \to N \to O \to P \to T \to U$$

Sendo o que apresentou o resultado mais rapidamente porém o pior, acredito que isto ocorreu por causa do jeito que o labirinto é formado, com muitas conexões e sem becos sem saída pois isso faz com que o DFS passa-se por muitas salas e sempre acabasse encontrando uma próxima, em um outro ambiente onde houvessem menos conexões ele teria um desempenho melhor.

V. Conclusão

Este trabalho apresentou a implementação e análise de quatro algoritmos de busca aplicados à navegação em um labirinto composto por 21 salas conectadas por portas. Os algoritmos estudados foram: Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e A*.

Foi realizado um experimento de busca partindo da sala A até U e a partir dos resultados obtidos, observou-se que os algoritmos BFS, Busca Gulosa e A* retornaram caminhos equivalentes em termos de custo, sendo a Busca Gulosa a mais eficiente em tempo de execução. Por outro lado, o DFS apresentou o caminho de maior custo, embora tenha sido o mais rápido em termos de tempo de computação, evidenciando que seu desempenho depende da estrutura do ambiente.

A análise demonstra que algoritmos baseados em heurísticas, como Busca Gulosa e A*, oferecem um bom equilíbrio entre eficiência e qualidade do caminho encontrado, enquanto algoritmos não heurísticos, como BFS e DFS, podem ser mais previsíveis ou rápidos, dependendo da topologia do labirinto e da forma que foram implementados em código como ”quais conexões eles explorarão primeiro”.

Para futuros trabalhos, sugere-se a aplicação em um ambiente diferente com menos conexões e mais salas, incluindo os pesos das portas de forma fixa para que o pesquisador não tenha que supor um peso, e para algum mais avançado a criação de um labirinto por meio do usuário e adaptação do código atual sendo assim mais cenários poderiam ser testados sem a necessidade de rescrever código ou a necessidade de algum programador para esses novos trabalhos, categorizado os ambientes em que cada algoritmo prevalece dado algumas características (muitas conexões labirinto aberto, poucas conexões labirinto fechado).

References

[1] S. Russell, P. Norvig. Inteligência artificial: referência completa para cursos de computação, adotado em mais de 750 universidades em 85 países. Rio de Janeiro: Campus, Elsevier, 2013.