E-fólio B | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Introdução à Inteligência Artificial

CÓDIGO: 21071

DOCENTE: José Coelho

A preencher pelo estudante

NOME: Gonçalo Caraça

N.º DE ESTUDANTE: 2000130

CURSO: Engenharia Informática

DATA DE ENTREGA: 22.05.2023

Índice

1. INTRODUÇÃO

Este trabalho, conhecido como "Efólio B", faz parte das atividades de avaliação da UC de Inteligência Artificial, integrada no currículo do curso de Engenharia Informática da Universidade Aberta, no ano letivo de 2022/2023. O principal objetivo deste efólio é avaliar o conhecimento e a capacidade dos alunos em relação aos algoritmos de procura informada.

Ao longo do relatório, serão apresentados vários algoritmos, suas vantagens e desvantagens e o porquê de serem, ou não, considerados ideias para o problema apresentado. Serão também abordados os detalhes da implementação dos algoritmos escolhidos, os resultados obtidos e uma análise detalhada aos mesmos.

Este trabalho é de extrema importância para a compreensão e aplicação dos princípios fundamentais da inteligência artificial, contribuindo para a formação dos estudantes na área de algoritmos de procura informada. Através desta experiência, espera-se que os alunos possam adquirir habilidades práticas na resolução de problemas complexos e desenvolver uma visão crítica sobre a eficiência dos algoritmos utilizados.

2. ANÁLISE DO PROBLEMA

O problema envolve uma cidade medieval com zonas habitacionais, áreas de passagem e muralhas. Há 4 portas, uma em cada ponto cardeal. O objetivo é garantir que todas as zonas habitacionais tenham acesso às 4 portas.

O objetivo é minimizar o número de movimentos necessários para que todas as zonas habitacionais tenham acesso às portas da cidade, as zonas habitacionais sem acesso às portas da muralha têm um custo adicional de 100 pontos cada uma. Portanto, é importante minimizar o número de zonas habitacionais sem acesso, para reduzir ao máximo o número de penalizações.

Para resolver o problema poderão ser usadas procuras informadas, tais como ***melhor primeiro***, ***escalada do monte, BnB*** ou ***AStar***. De seguida falaremos um pouco acerca delas.

2.1. Melhor Primeiro

É uma procura orientada por heurística, o que significa que sua eficácia depende da qualidade da função heurística utilizada. A função heurística deve ser admissível, ou seja, não pode superestimar o custo para atingir o objetivo. Se a função heurística for admissível, a procura Melhor Primeiro garantirá a solução ótima.

Tendo em conta as suas características, poderá ser menos adequada para o problema proposto pelas seguintes razões:

Comportamento Guloso: Pode ser considerada uma procura gulosa, pois seleciona o nó com a menor função heurística sem levar em consideração o custo real acumulado até o momento. Isso pode levar a escolhas que pareçam promissoras com base apenas na heurística local, mas que podem levar a caminhos subótimos globalmente.

Complexidade do Espaço de Procura: O problema em questão pode ter um espaço de procura muito grande, considerando a quantidade de configurações possíveis para as zonas habitacionais. A procura Melhor Primeiro não garante uma exploração completa do espaço de procura e pode ficar presa em locais ótimos ou não encontrar a solução ótima globalmente.

Heurística Adequada: A qualidade da função heurística é essencial para o desempenho da procura Melhor Primeiro. E tendo em conta a complexidade do problema proposto, encontrar uma função heurística admissível e eficaz que estime com precisão a qualidade das configurações das zonas habitacionais em relação ao objetivo de acesso às 4 portas pode ser bastante desafiador e difícil de atingir.

2.2. Escalada do Monte

É um algoritmo de procura local que procura melhorar iterativamente um estado atual através de movimentos que levem a melhorias locais. O algoritmo seleciona o próximo estado com base na avaliação heurística e faz a transição para esse estado se for considerado melhor que o estado atual.

No entanto, esta procura pode não ser a abordagem mais ideal para o problema em questão por algumas razões:

Locais ótimos: A procura Escalada do Monte pode ficar presa em locais ótimos, onde não é possível fazer melhorias adicionais sem retroceder. No contexto do problema, isso significa que pode encontrar uma configuração onde algumas zonas habitacionais têm acesso a três portas, mas não é possível alcançar as quatro portas sem passar por outras zonas habitacionais.

Exploração Limitada do Espaço de Procura: Tendo em conta que é uma procura local que foca apenas em movimentos que levam a melhorias locais imediatas, ela não explora o espaço de procura de forma abrangente, o que pode resultar em soluções subótimas, não encontrando a solução ótima globalmente.

Dependência da Configuração Inicial: A eficácia desta procura pode depender da configuração inicial das zonas habitacionais. Se a configuração inicial não permitir um caminho claro para alcançar as quatro portas, o algoritmo pode não ser capaz de encontrar uma solução satisfatória.

2.3. Branch and Bound (BnB)

É um algoritmo de procura exaustiva que procura encontrar a solução ótima em problemas de otimização. Ele utiliza a técnica de dividir e conquistar, explorando o espaço de procura através de uma árvore de estados.

No algoritmo BnB, a procura é realizada de forma sistemática, dividindo o espaço de procura em subespaços menores e avaliando os limites superiores e inferiores dos valores de solução em cada subespaço. A partir desses limites,

ramos da árvore de procura que não podem levar à solução ótima são podados, reduzindo o espaço de procura a ser explorado.

Embora a procura BnB seja eficaz para encontrar soluções ótimas em problemas de otimização, ela pode não ser a mais ideal para o problema proposto por algumas razões:

Complexidade do Espaço de Procura: O problema proposto pode ter um espaço de procura muito grande, considerando as várias configurações possíveis das zonas habitacionais. A procura BnB exploraria exaustivamente todas as configurações possíveis, o que pode ser computacionalmente inviável devido ao alto custo de tempo de execução.

Limitações da Estrutura de Árvore: A abordagem de árvore da procura BnB pode não ser a mais adequada para o problema, pois pode levar a uma grande árvore de procura com muitos ramos e nós. Isso pode dificultar a exploração eficiente do espaço de procura e levar a um alto consumo de memória.

Restrições e Complexidade das Restrições: O problema proposto envolve várias restrições e complexidades, como garantir que todas as zonas habitacionais tenham acesso às quatro portas. Considerar todas essas restrições e encontrar limites superiores e inferiores para cada configuração pode ser bastante desafiador e requerer um esforço significativo na abordagem do problema.

2.4. AStar

É um algoritmo de procura informada que pode combinar, a procura em largura ou a procura em profundidade, com uma função heurística para encontrar a solução ótima em problemas de otimização.

No algoritmo AStar, cada nó é avaliado com base na soma do custo acumulado do caminho percorrido até o nó atual e em uma função heurística que estima o custo restante para alcançar o objetivo a partir desse nó. Essa soma é

chamada de "função de avaliação" ou "valor f". O algoritmo prioriza a expansão dos nós com menor valor f, pois eles são considerados os mais promissores em direção à solução ótima.

A procura AStar é considerada ideal para o problema por várias razões:

Exploração Eficiente do Espaço de Procura: Tendo em conta que podemos combiná-lo com uma procura em largura ou profundidade, isso permite uma exploração mais eficiente do espaço de procura, direcionando a procura para as áreas mais promissoras em direção à solução ótima.

Consideração da Heurística e do Custo Real: A função heurística no AStar fornece informações adicionais sobre a qualidade dos estados e direciona a procura em direção ao objetivo. Ao mesmo tempo, o algoritmo leva em consideração o custo acumulado até o estado atual, garantindo que a solução encontrada seja ótima em termos de custo total.

Flexibilidade para Incorporar Restrições: O algoritmo AStar permite que sejam incorporadas restrições específicas do problema, como garantir que todas as zonas habitacionais tenham acesso às quatro portas. A função heurística pode ser projetada para considerar essas restrições, orientando a procura para encontrar soluções que atendam a essas condições.

Garantia de Otimização: Se a função heurística utilizada no AStar for admissível (não superestimando o custo para atingir o objetivo), o algoritmo garante a obtenção da solução ótima.

Ao utilizar o algoritmo AStar combinado com uma procura em profundidade, é possível explorar diferentes configurações das zonas habitacionais, levando em conta a relação entre custo e distância para movê-las de forma eficiente. É importante conseguir ajustar a função heurística de acordo com as características específicas do problema para obter melhores resultados.

Portanto, devido à sua capacidade de explorar eficientemente o espaço de procura, considerar heurísticas e garantir otimização, optei por usar o algoritmo AStar combinado com a procura em profundidade para resolver o problema proposto.

3. IDÊNTIFICAÇÃO DOS ALGORITMOS

O algoritmo implementado para resolver o problema proposto pelo Rei, combina os benefícios do algoritmo A* (A-estrela) e da procura em profundidade. Essa combinação permite uma procura direcionada e eficiente, explorando as possibilidades de forma otimizada e priorizando os nós mais promissores.

No programa, o algoritmo A* utiliza uma função heurística para estimar o custo de alcançar o objetivo a partir de um estado atual. Essa função combina o custo atual com a heurística para determinar a prioridade de exploração de cada estado, permitindo que o algoritmo escolha caminhos mais promissores e encontre soluções eficientes.

Por sua vez, a procura em profundidade é utilizada para implementar uma pilha de nós, explorando um ramo do espaço de soluções o mais longe possível antes de retroceder. Essa abordagem é útil em problemas em que a velocidade com que se chega á solução é mais importante do que o custo associado a ela.

Além dos algoritmos de procura, o programa utiliza estruturas de dados adequadas, como vetores e matrizes, para representar a cidade. Funções auxiliares são implementadas para verificar o acesso às portas, obter os vizinhos de uma posição na cidade e calcular a heurística.

Durante a execução do programa, várias instâncias do problema são resolvidas por meio do algoritmo A*. Para cada instância, são registados o tempo de execução, o número de nós gerados e o número de nós avaliados. Os resultados são apresentados de forma clara e organizada para análise e comparação.

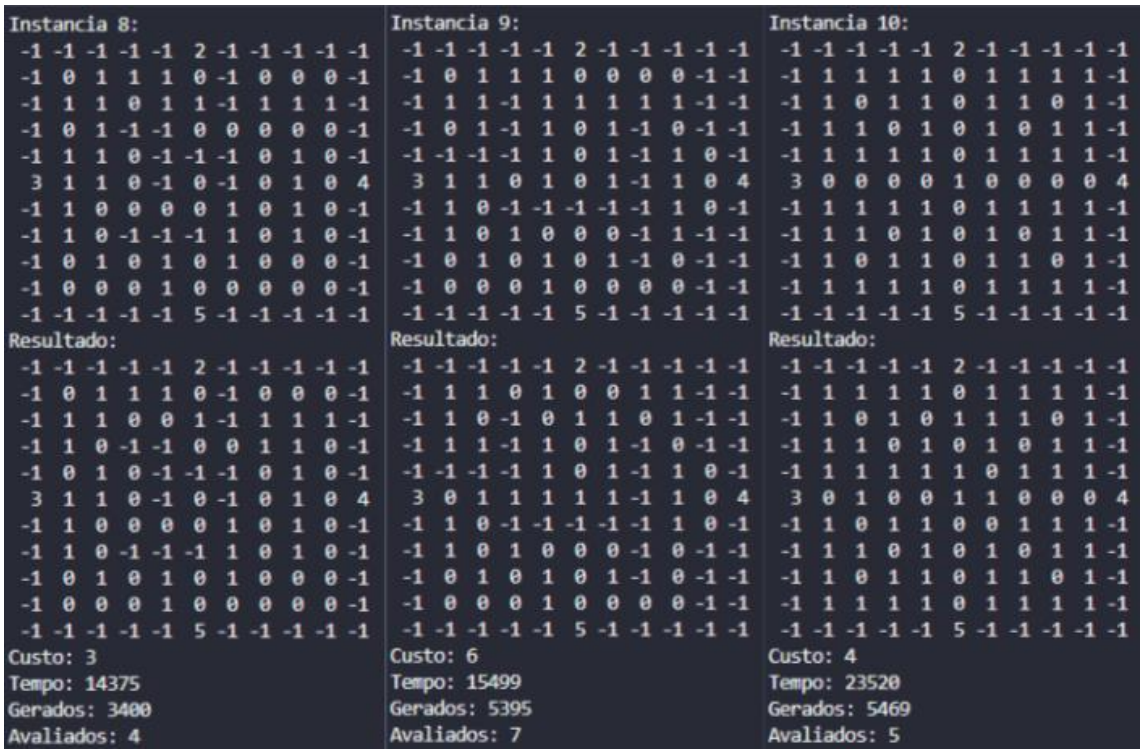
Dessa forma, o algoritmo implementado procura resolver o problema de garantir acesso de todas as zonas habitacionais às 4 portas, utilizando o algoritmo A* para guiar a procura, e o algoritmo procura em profundidade para explorar as possibilidades de movimentação. Essa abordagem permite encontrar uma solução ótima ou uma solução aproximada com o menor custo de movimentos dentro de um tempo razoável de execução.

4. RESULTADOS

Neste capítulo será apresentado o output do programa desenvolvido, uma tabela com todos os resultados pretendidos, bem como uma análise precisa e concisa aos mesmo.

4.1. Output do programa

```
Instancia 1:
-1 -1 -1  2 -1 -1 -1
-1  0  1  1  1  0 -1
-1  1  1  0  1  1 -1
 3  0  1 -1 -1  0  4
-1  1  1  0  1  0 -1
-1  1  1  0  0  0 -1
-1 -1 -1  5 -1 -1 -1
Resultado:
-1 -1 -1  2 -1 -1 -1
-1  1  0  1  1  0 -1
-1  1  1  0  1  1 -1
 3  1  0 -1 -1  0  4
-1  0  1  0  1  0 -1
-1  1  1  1  0  0 -1
-1 -1 -1  5 -1 -1 -1
Custo: 3
Tempo: 179
Gerados: 388
Avaliados: 4

Instancia 2:
-1 -1 -1  2 -1 -1 -1
-1  0  1  1  0  0 -1
-1  1  1 -1  1  1 -1
 3  0 -1 -1 -1  0  4
-1  1  1 -1  1  0 -1
-1  0  1  0  1  1 -1
-1 -1 -1  5 -1 -1 -1
Resultado:
-1 -1 -1  2 -1 -1 -1
-1  1  1  1  0  1 -1
-1  0  1 -1  0  1 -1
 3  0 -1 -1 -1  0  4
-1  1  0 -1  0  1 -1
-1  1  1  0  1  1 -1
-1 -1 -1  5 -1 -1 -1
Custo: 4
Tempo: 100
Gerados: 381
Avaliados: 5

Instancia 3:
-1 -1 -1  2 -1 -1 -1
-1  0  1  1  0  0 -1
-1  1 -1  0  1  1 -1
 3  1 -1 -1 -1  0  4
-1  0  1  1 -1  0 -1
-1  0  1  0  1  1 -1
-1 -1 -1  5 -1 -1 -1
Resultado:
-1 -1 -1  2 -1 -1 -1
-1  1  0  1  1  1 -1
-1  1 -1  0  0  1 -1
 3  1 -1 -1 -1  0  4
-1  0  0  1 -1  0 -1
-1  1  1  0  0  1 -1
-1 -1 -1  5 -1 -1 -1
Custo: 4
Tempo: 126
Gerados: 393
Avaliados: 5

Instancia 4:
-1 -1 -1  2 -1 -1 -1
-1  1  1  1  1  0 -1
-1  1  1  0  0  1 -1
 3  1  0  1  0  1  4
-1  1  0  0  1  1 -1
-1  0  1  1  1  1 -1
-1 -1 -1  5 -1 -1 -1
Resultado:
-1 -1 -1  2 -1 -1 -1
-1  1  0  1  1  1 -1
-1  0  1  0  0  1 -1
 3  1  1  1  0  1  4
-1  1  1  0  1  0 -1
-1  1  0  1  1  1 -1
-1 -1 -1  5 -1 -1 -1
Custo: 4
Tempo: 331
Gerados: 541
Avaliados: 5

Instancia 5:
-1 -1 -1 -1  2 -1 -1 -1 -1
-1  1  1  1  1  0  1  1 -1
-1  0  1  0  0  0  0  0 -1
-1  1  1  0  1  0 -1  1 -1
 3  0  1  0  0  1 -1  0  4
-1  1 -1 -1 -1  1 -1  1 -1
-1  1  0  0  0  1  0  1 -1
-1  0  0  1  0  0  0  0 -1
-1 -1 -1 -1  5 -1 -1 -1 -1
Resultado:
-1 -1 -1 -1  2 -1 -1 -1 -1
-1  1  1  1  1  0  1  1 -1
-1  0  1  1  0  1  0  1 -1
-1  1  0  0  1  0 -1  0 -1
 3  0  1  0  0  1 -1  0  4
-1  1 -1 -1 -1  0 -1  1 -1
-1  1  0  0  0  1  0  1 -1
-1  0  0  1  0  0  0  0 -1
-1 -1 -1 -1  5 -1 -1 -1 -1
Custo: 4
Tempo: 3614
Gerados: 1837
Avaliados: 5

Instancia 6:
-1 -1 -1 -1  2 -1 -1 -1 -1
-1  0  1  0  0  0  1  0 -1
-1  0  1  0  1  1  1  0 -1
-1  1  0 -1 -1 -1  0  1 -1
 3  0  1 -1 -1 -1  1  0  4
-1  1  1 -1 -1 -1  0  1 -1
-1  0  1  1  0  1  0  1 -1
-1  0  0  1  1  0  1  0 -1
-1 -1 -1 -1  5 -1 -1 -1 -1
Resultado:
-1 -1 -1 -1  2 -1 -1 -1 -1
-1  1  1  1  0  0  0  0 -1
-1  0  1  0  1  1  1  0 -1
-1  1  0 -1 -1 -1  0  1 -1
 3  0  1 -1 -1 -1  1  0  4
-1  0  1 -1 -1 -1  0  1 -1
-1  0  1  1  0  1  0  1 -1
-1  0  0  1  1  0  1  0 -1
-1 -1 -1 -1  5 -1 -1 -1 -1
Custo: 2
Tempo: 1229
Gerados: 799
Avaliados: 3

Instancia 7:
-1 -1 -1 -1  2 -1 -1 -1 -1
-1  1  1  1  1  1  1  1 -1
-1  1  1  0  0  0  1  1 -1
-1  1  0  1  0  1  0  1 -1
-1  1  1  0  0  0  1  1 -1
-1  1  1  1  1  1  1  1 -1
-1 -1 -1 -1  5 -1 -1 -1 -1
Resultado:
-1 -1 -1 -1  2 -1 -1 -1 -1
-1  1  1  0  1  1  1  1 -1
-1  1  0  1  1  1  0  1 -1
-1  0  0  1  0  1  0  0 -1
 3  1  1  0  1  0  1  1  4
-1  1  0  1  1  1  0  1 -1
-1  1  0  1  1  1  0  1 -1
-1  1  1  0  1  1  1  1 -1
-1 -1 -1 -1  5 -1 -1 -1 -1
Custo: 8
Tempo: 14775
Gerados: 5088
Avaliados: 11
```



O output exibe sempre o número da instância, bem como a instância inicial (definida no enunciado), o resultado obtido (o melhor caso), o seu custo, tempo de execução, número de cidades geradas e de cidades avaliadas.

4.2. Tabela de resultados

	Instância	1	2	3	4	5	6	7	8	9	10
A Star	Avaliações	4	5	5	5	5	3	11	4	7	5
	Gerações	388	381	393	541	1837	799	5088	3400	5395	5469
	Custo	3	4	4	4	4	2	8	3	6	4
	Tempo (msec)	179	100	126	331	3614	1229	14775	14375	15499	23520
	Tempo (sec)	0,18	0,10	0,13	0,33	3,61	1,23	14,78	14,38	15,50	23,52

A tabela de resultados acima exibe, para cada instância, o número de avaliações, gerações, custo, tempo em milissegundos e em segundos, obtidos com o algoritmo desenvolvido.

4.3. Análise de resultados

Analisando os resultados obtidos pelo algoritmo para as diferentes instâncias, podemos observar o seguinte:

Avaliações: O número de avaliações representa a quantidade de nós que foram avaliados durante a execução do algoritmo. É interessante notar que, de um modo geral, as instâncias apresentam um número baixo de avaliações, o que indica que o algoritmo está sendo eficiente ao explorar o espaço de procura.

Gerações: O número de gerações representa a quantidade de nós gerados durante a execução do algoritmo. Observamos que as instâncias variam em termos de quantidade de gerações, o que sugere que a complexidade do problema pode influenciar a expansão do espaço de procura. Instâncias com maior número de gerações podem indicar uma maior complexidade do problema.

Custo: O custo representa a medida de qualidade da solução encontrada pelo algoritmo. Em geral, observamos que as instâncias têm um custo baixo, o que indica que o algoritmo foi capaz de encontrar soluções com um número reduzido de trocas de zonas habitacionais.

Tempo: O tempo de execução do algoritmo, medido em milissegundos, mostra o esforço computacional necessário para obter as soluções. De notar que, à medida que o tamanho das instâncias aumenta, o tempo de execução também aumenta significativamente. Isso sugere que o algoritmo pode enfrentar desafios computacionais maiores para problemas mais complexos.

Em resumo, os resultados mostram que o algoritmo A* combinado com a procura em profundidade foi capaz de encontrar soluções com baixo custo em um número razoável de avaliações, podemos inferir então que, de facto, tendo em conta a proximidade dos valores destas duas variáveis de análise (custo e avaliações), podemos concluir, que o algoritmo errou poucos caminhos até chegar a solução ótima.

5. BIBLIOGRAFIA

Russell, S. J., Norvig, P., & Davis, E. (2010). Artificial intelligence: A modern approach (3rd ed). Prentice Hall.

A* search algorithm. In Wikipedia. Obtido em 18 de Maio de 2023, de https://en.wikipedia.org/wiki/A*_search_algorithm

Depth-first search. In Wikipedia. Obtido em 18 de Maio de 2023, de https://en.wikipedia.org/wiki/Depth-first_search

FIM