Universidade Estadual do Oeste do Paraná

Desenvolvimento e Comparação de Algoritmos Dijkstra de Busca Cega e A* de Busca Informada

Cinthia Rodrigues da Silva
Khadije Elisa Lopes El Zein

Foz do Iguaçu
Abril/2018

Desenvolvimento e Comparação de Algoritmos Dijkstra de Busca Cega e A* de Busca Informada

Cinthia Rodrigues da Silva
Khadije Elisa Lopes El Zein
Universidade Estadual do Oeste do Paraná
Centro de Engenharia e de Ciências Exatas
Curso de Ciência da Computação
email : {khadyelzein,cinthiah.rodrigues}@gmail.com

Resumo. Em computação, um problema, como a menor distância entre duas cidades, pode ser considerado como um objetivo. Um conjunto de ações pode ser seguido para alcançar esse objetivo. Ao tentar resolver o problema por alcançar o estado final, ou seja buscar um objetivo estamos em determinado estado, o estado inicial é quando iniciamos a busca e o estado que satisfaz o objetivo é o estado final ou objetivo. Algoritmos de busca são usados para examinar o espaço de estados de um problema e encontrar um objetivo. Esses algoritmos podem ser divididos em técnicas de busca cega e técnicas de busca informada.Neste trabalho, são implementados dois algoritmos, um de busca cega e outro de busca informada, sendo eles Dijkstra e A*, respectivamente. Estes são então comparados baseando-se na média de seus tempos de execução e na quantidade de nós explorados. Resultados experimentais mostram que o algoritmo A* alcança o mesmo resultado que o Dijkstra em um tempo de execução menor e com menos nós sendo explorados.
Palavras-Chave: Algoritmos de Busca, Busca Cega, Busca Informada, Dijkstra, A*.

Sumário

Introdução **2**

Critério de Seleção dos Algoritmos **3**

Dijkstra **3**

4. A* **4**

5. Configuração dos experimentos **5**

6. Resultados e Discussão **5**

7. Considerações finais **6**

1. Introdução

Algoritmos de busca são usados para examinar o espaço de estados de um problema e encontrar um objetivo. Um algoritmo de busca determina a ordem correta de aplicação dos operadores para levar do estado inicial ao estado objetivo.
“Isto é feito por um processo de geração (de estados possíveis) e teste (para ver se o objetivo está entre eles)” (Lee,H; 2018). A solução da busca é um conjunto ordenado de operadores a serem aplicados.
Existem duas categorias para algoritmos de busca, busca exaustiva ou cega (não informada) ou busca heurística (informada).
A busca cega encontra uma solução para um problema através da geração sistemática de novos estados, estados esses constantemente comparados ao objetivo. (Lee, H;2018).
A busca heurística ou informada estima qual o melhor nó da fronteira a ser expandido com base em funções heurísticas, ou seja, se “utiliza conhecimento específico do problema na escolha do próximo nó a ser expandido”. Uma função heurística estima o custo do caminho do nó atual até o objetivo, indicando o “quanto um nó é promissor em relação a atingir a meta estabelecida”.(Lee, H).
Neste trabalho investigamos os algoritmos de busca cega, Dijkstra e o de busca informada, A*. Resultados experimentais provenientes da comparação desses algoritmos na questão de tempo de execução e quantidade de nós explorados mostram que o algoritmo A* obtém o mesmo resultado do Dijkstra em um tempo de execução menor e com menos nós sendo explorados.
Este trabalho está organizado do seguinte modo: Na Seção 2 é apresentado o critério de seleção dos algoritmos, na 3 e 4 são apresentados brevemente os algoritmos Dijkstra e A*. Na Seção 5 é descrita a implementação destes algoritmos. Na seção 5 é mostrado a configuração dos experimentos realizados. Resultados e discussão dos experimentos são apresentados na Seção 6 e considerações finais são apresentadas na Seção 7.

2. Critério de Seleção dos Algoritmos

Para fazer a comparação entre os algoritmos de busca cega e busca informada, foram definidos critérios como a informação de interesse é o caminho do estado inicial para o estado final. O algoritmo deve ser completo, ou seja,deve sempre encontrar uma solução quando essa existe e ótimo, ou seja, deve encontrar a melhor solução quando existem soluções diferentes.
Ao analisar os algoritmos Hill Climbing e Têmpera Simulada notou-se que eles não fornecem o caminho como solução, pois são algoritmos de busca local e a característica dos mesmos é guardar apenas a configuração final.
Ao analisar os algoritmos de Busca em Profundidade e Busca Gulosa notou-se que não são algoritmos completos nem ótimos.

Ao analisar o algoritmo de Busca em Largura e Aprofundamento iterativo foi evidenciado que estes apenas seriam ótimos se os operadores tivessem valores iguais, ou seja, custo igual.
Deste modo, foram selecionados os algoritmos de Custo Uniforme (Dijkstra) e A* por serem os únicos a cumprirem os critérios considerando operadores com valores distintos.

3. Dijkstra

O Algoritmo de Dijkstra (E.W. Dijkstra) é um dos algoritmos que calcula o caminho de custo mínimo entre vértices de origem e destino de um grafo através da estratégia de busca cega.Este algoritmo calcula o custo mínimo deste vértice para todos os demais vértices do grafo.
O algoritmo expande o nó da fronteira com menor custo de caminho até o momento e cada operador pode ter um custo associado diferente, determinado pela função g(n) que dá o custo do caminho da origem até o nó n (Lee, H). Dijkstra não garante, a exatidão da solução caso haja a presença de arcos com valores negativos.

4. A*

É um algoritmo para encontrar um melhor caminho de uma origem até um estado destino. Ele tenta minimizar o custo total da solução ao combinar o algoritmo de Busca Gulosa (simples, mas não é completo nem ótimo) e do algoritmo de custo uniforme (Dijkstra. É ineficiente mas completo e ótimo). Utiliza a função de avaliação f(n) = g(n) + h(n), sendo g(n) a distância do nó atual até o nó inicial e h(n) a distância estimada nó atual até o nó final. É a técnica de busca mais usada até o momento (Lee, H;2018).

5. Algoritmos implementados

Os algoritmos foram desenvolvidos no Ambiente de desenvolvimento IntelliJ IDEA Community Edition 2017.3.1, na linguagem de programação Java. O programa recebe um arquivo de entrada para executar os algoritmos.
O espaço de estados é constituído de todos os nós que compõem o grafo, lidos através do arquivo de entrada. O estado inicial e final correspondem, respectivamente ao primeiro e segundo nós lidos do arquivo de entrada.
A estratégia de controle é, para o algoritmo de Dijkstra, se o nó atual estiver com o menor custo então ele é colocado na lista de prioridades e, para o algoritmo A*, se o nó atual estiver com a função de avaliação (f(n) = g(n) + h(n)) com o menor valor então adiciona o nó a lista de prioridades.

6. Configuração dos experimentos

Ambos os algoritmos foram executados em um notebook Dell Inspiron 5577, com Sistema Operacional Microsoft Windows 10 Home versão 10.0.16299 Build 16299 X64, com processador Intel Core i7-7700HQ CPU 2.80 GHz, 2801 MHz, QuadCore com 8 processadores lógicos, com versão da Bios Dell Inc 1.0.0, com 8 GB de memória RAM DDR4, com Java versão 1.8.0_161, build 1.8.0_161-b12.
Foram escolhidos dois métodos para se comparar os algoritmos. Por tempo de processamento e por número de nós explorados.
Foram escolhidas essas métricas pois o algoritmo A*, apesar de baseado no Dijkstra, procura usar conhecimento específico do problema para estimar o custo do caminho do nó atual até o objetivo e expandir um nó apenas se ele parece promissor enquanto Dijkstra calcula o custo de todos os nós,logo A* explora menos nós e converge mais rápido do que Dijkstra. Essas métricas são as mais adequadas para comprovar tal afirmação.
Para se comparar os algoritmos por tempo de processamento foi usada a função System.nanoTime() que retorna o tempo do sistema em nanossegundos. A função foi chamada no início e final dos métodos e o tempo de processamento foi calculado pela diferença desses tempos. Os algoritmos foram executados 10 vezes,cada um, para cada arquivo de entrada. Foram usados dois arquivos de entrada distintos. Após o processo foram calculadas as médias, desvios padrões e coeficiente de variação dos tempos de processamento.
O número de nós explorados foi calculado usando a coleção de nós explorados (Set<Vertice> explorados), cada vez que um nó ia ser explorado e era retirado da fronteira o mesmo era adicionado à lista de nós explorados, após se alcançar o estado meta era obtido o tamanho dessa coleção e o mesmo era impresso.
Os arquivos de entrada utilizados foram : 1-entrada.txt obtido no site de Huei Diana Lee como parte das especificações para realização deste trabalho e entrada.txt, arquivo composto de 14 nós, representando o problema do mapa da Romênia, o estado inicial sendo Arad e o final Bucharest, as heurísticas e custos são os mesmos do problema do mapa da Romênia.

7. Resultados e Discussão

Para cada arquivo de entrada, foi realizada a busca cega e a informada, usando os dois modelos de algoritmos estudados neste trabalho, totalizando um total de 40 execuções.
A comparação do tempo de processamento foi realizada através da média de cada conjunto de 10 execuções, seu respectivo desvio padrão e coeficiente de variação.
As médias foram calculadas seguindo a seguinte fórmula: $MA = \sum_{i=1}^{n} xi/n$, sendo xi o dado.

O desvio padrão foi calculado seguindo a seguinte fórmula:

$DP = \sqrt{\sum_{i=1}^{n}(xi - MA^2)/n}$, sendo xi o dado e MA a média Aritmética

O coeficiente de variação foi calculado seguindo a seguinte fórmula: $(DP/MA) * 100$

A comparação do número de nós explorados foi feita através da fórmula : $(nMenor/nMaior) * 100$

As subseções a seguir mostram os resultados obtidos.

7.1 Comparação dos tempos de processamento.

De acordo com o arquivo 1-entrada.txt, os tempos de processamento obtidos executando Dijkstra foram, em nanossegundos: 920798, 990815, 1472184, 850417, 896730, 1070314,
981333,842029,821243 e 932468.

De acordo com o arquivo entrada.txt os tempos de processamento obtidos executando Dijkstra foram, em nanossegundos: 1115533, 954348, 1081254, 1009779, 867191, 1150542, 1065938, 1110793, 1034941, 912411.

De acordo com o arquivo 1-entrada.txt, os tempos de processamento obtidos executando A* foram, em nanossegundos: 944866, 829265, 802645, 1071408, 799362, 798269, 1052080, 861722, 900377,879590.

De acordo com o arquivo entrada.txt os tempos de processamento obtidos executando A* foram, em nanossegundos: 742109, 823431, 780035, 770533, 844581,789881,872661,824889, 824525, 769459.

As tabela 1 e 2 são tabelas comparativa, mostrando as médias, desvios padrões e coeficiente de variação do algoritmo Dijkstra e A*, respectivamente.

Tabela 1: Tempo de Processamento do algoritmo Dijkstra

Arquivo de Entrada	Média (ns)	Desvio Padrão (ns)	Coeficiente de variação (%)
1-entrada.txt	977833.1	189796.1162	19.40986823
entrada.txt	1030273	93560.44665	9.081131569

*Tabela 2: Tempo de Processamento do algoritmo A**

Arquivo de Entrada	Média (ns)	Desvio Padrão (ns)	Coeficiente de variação (%)
1-entrada.txt	804210.4	40156.88891	4.993331212%
entrada.txt	893958.4	100552.6577	11.24802426 %

No arquivo 1- entrada o algoritmo A* teve um tempo de processamento 82.2441% mais rápido e no arquivo entrada o algoritmo A* teve um tempo de processamento 86.7691% mais rápido, evidenciando que o algoritmo A* converge mais rapidamente, já que explora menos nós.

7.2 Comparação por número de nós explorados

A tabela 3 mostra a quantidade de nós explorados considerando os dois arquivos de entrada, para os dois algoritmos.

Tabela 3 : Comparação de Dijkstra e A pelo número de nós explorados.*

Arquivo de Entrada	Dijkstra	A*
1-entrada.txt	12	7
entrada.txt	14	6

No arquivo 1-entrada o algoritmo Dijkstra explorou 1,7143 vezes mais nós, ou seja, o algoritmo A* explorou 58,3333 % menos nós em comparação com Dijkstra.

No arquivo entrada o algoritmo Dijkstra explorou 2,3333 vezes mais nós, ou seja, o algoritmo A* explorou 42,8571 % menos nós em comparação com Dijkstra.

Esses resultados comprovam que o algoritmo A* explora menos nós que o Dijkstra quando a heurística é admissível.

7. Considerações finais

Neste trabalho foi apresentada o estudo e implementação de dois algoritmos de Busca, bem como um conjunto de métricas através das quais os algoritmos são comparados.

Os resultados obtidos através das execuções dos mesmos mostram que o algoritmo A*

é capaz de chegar a uma solução ótima e completa, assim como o Dijkstra em menor tempo de processamento e com menor número de nós explorados, pois ao considerar uma heurística que seja admissível procura os nós mais promissores.

Referências

Algoritmo de Dijkstra para cálculo do Caminho de Custo Mínimo. *UFSC. Disponível em: <http://www.inf.ufsc.br/grafos/temas/custo-minimo/dijkstra.html>. Acesso em: 14 abr. 2018.*

Material Didático Prof. Huei Diana Lee – Unioeste.