Análise e Comparação dos Algoritmos de Dijkstra e A-Estrela na Descoberta de Caminhos Mínimos em Mapas de Grade

Marcel L. Rios[1,2], Francisco S. S. Neto[2], José F. Magalhães Netto[2]

[1]Instituto Federal de Rondônia - Campus Porto Velho Calama (IFRO) - Porto Velho, RO - Brasil

[2]Programa de Pós Graduação em Informática - Universidade Federal do Amazonas (UFAM) - Manaus, AM - Brasil

marcel.rios@ifro.edu.br, laertesnetho@gmail.com, jfmnetto@gmail.com

Abstract. *This paper presents a comparison between two algorithms for finding shortest paths in grid maps. First of them is Dijkstra, a well known greedy algorithm for finding the shortest path between vertex in a given graph. And the second one is A* (A-star), an algorithm that uses heuristics to predict their behavior, also for going through a graph and finding the shortest path between vertex. For each algorithm a stop condition has been added, showing their pseudocode, analyzing their complexity and showing promising results of A-Star compared to Dijkstra.*

Resumo. *Este trabalho apresenta uma comparação entre dois algoritmos de descoberta de caminhos mínimos em mapas de grade. O primeiro deles é o Dijkstra, um algoritmo guloso muito conhecido por encontrar o caminho mais curto entre vértices em um dado grafo. O segundo é o A* (A-Estrela), um algoritmo que utiliza heurística para prever seu comportamento, também percorrendo um grafo e encontrando o menor caminho entre vértices. Para cada algoritmo foi adicionada uma condição de parada, demonstrando os respectivos pseudocódigos, análisando a complexidade dos mesmos e apresentando resultados promissores do A-Estrela em comparação ao Dijkstra.*

1. Introdução

Os algoritmos de caminho mínimo, por definição, buscam encontrar, a partir de uma posição inicial, o caminho de custo mínimo até uma posição final (alvo). Segundo [Djojo and Karyono 2013] o problema do caminho mínimo é um dos problemas de otimização mais frequentemente estudado e adaptado em várias aplicações.

Segundo [Russell and Norvig 2013] os algoritmos de descoberta de caminhos mínimos podem ser classificados de duas formas: Busca Não Informada, quando o algoritmo não faz uso de heurísticas para descobrir o menor caminho entre origem e destino. E Busca Informada, quando o algoritmo faz uso de heurísticas para estimar o caminho de custo mínimo.

Nessa perspectiva, o presente trabalho busca analisar o comportamento e a complexidade de dois importantes algoritmos: Dijkstra (Busca Não-Informada) e A-Estrela (Busca Informada), ambos aplicados em mapas de grade, cujas arestas possuem custos uniformes. Os resultados demonstraram a eficiência do A-Estrela em face do Dijkstra, evidenciando a importância da heurística adotada.

2. Abordagem e Métodos

O algoritmo de Dijkstra, segundo [Cormen et al. 2009], resolve o problema de caminhos mais curtos de origem única em um grafo orientado ponderado $G = (V, E)$ para o caso no qual todos os pesos de arestas são não negativos.

Para a implementação do algoritmo de Dijkstra em um mapa de grades, conforme demonstra a Figura 1, foi necessário acrescentar uma condição de parada (linha 14), onde o algoritmo, a cada vértice examinado, verifica se chegou ao nó de destino.

```
1  Program Dijkstra(G,w,s,a) ;
2   Begin
3        d[s] = ∅;
4        For cada v ∈ V[G]-{s} do
5            d[v] = ∞;
6            π[v] = NIL;
7        Q = V[G];
8        While Q ≠ ∅ do
9            u = ExtraiMin(Q);
10           For cada v ∈ Adj[u] do
11               if d[v] < d[u]+w(u,v) then
12                   d[v] = d[u]+w(u,v);
13                   π[v] = u;
14               if v == a then
15                   return constroiCam(π[], v);
16       return failure;
17  End.
```

Figura 1. Algoritmo de Dijkstra com Condição de Parada.

Além disso, caso o nó verificado seja o nó de destino, então o algoritmo chama uma função de construção do caminho mínimo (linha 15) entre os dois vértices. É possível observar que esta função, conforme demonstra a Figura 2, pode levar o tempo de $O(V)$, considerando sempre o pior caso.

```
1  Function constroiCam(π[],v) ;
2   Begin
3        caminhoFinal = v;
4        While v ∈ π[value] do
5            v = π[v];
6            caminhoFinal = caminhoFinal ∪ {v};
7        return caminhoFinal;
8   End.
```

Figura 2. Algoritmo de Construção do Caminho Mínimo.

Levando em consideração que, tradicionalmente, o algoritmo de Dijkstra, segundo [Cormen et al. 2009], possui o custo de tempo $O(ElogV)$, com a inserção da função supracitada, o Dijkstra recebeu $O(max(V + ElogV))$, mas permanecendo em $O(ElogV)$.

O algoritmo A* (A-Estrela), segundo [Russell and Norvig 2013], é a forma de solução mais amplamente conhecida da busca de melhor escolha, ela avalia os nós através da combinação de $g(n)$, o custo para alcançar o nó visitado, e $h(n)$, o custo para ir do nó visitado ao nó de destino.

Segundo [Djojo and Karyono 2013] o algoritmo A-Estrela realiza a busca de caminhos mínimos utilizando funções heurísticas, ou seja, a seleção dos nós é baseado na

distância a partir do nó de ínicio mais a distância aproximada até ao destino. Essa estimativa de aproximação pode ser representada pela função $f(n) = g(n) + h(n)$.

Segundo [Eraghi et al. 2014] cada célula do mapa de grades tem uma coordenada que é mostrada em (X,Y). Essa coordenada é utilizada por funções heurísticas que realizam a soma das distâncias ortogonais entre dois pontos, sendo X a posição do nó na coluna e Y a posição do nó na linha.

Para a implementação do algoritmo A-Estrela é necessário utilizar uma lista aberta (para os nós não examinados) e uma lista fechada (para os nós já examinados), além da função heurística: $f(n) = g(n) + h(n)$ existente para estimar a distância até o destino, conforme demonstra a Figura 3.

```
1  Program A-Estrela(G,w,inicio,alvo);
2   Begin
3        ls_fechada = Ø;
4        ls_aberta = inicio;
5        g[inicio] = Ø;
6        f[inicio] = g[inicio] + h(inicio, alvo);
7        While ls_aberta ≠ Ø do
8            u = ExtraiMin(ls_aberta);
9            ls_fechada = u;
10           For cada v ∈ Adj[u] do
11                if v ∈ ls_fechada then
12                else
13                    geraG = g[u]+w(u,v);
14                    if v ∉ ls_aberta or geraG < g[v] then
15                        π[v] = u;
16                        g[v] = geraG;
17                        f[v] = g[v] + h(v, alvo);
18                        if v == goal then
19                             return constroiCam(π[],v);
20                        if v ∉ ls_aberta then
21                             ls_aberta = v;
22       return failure;
23  End.
```

Figura 3. Algoritmo A-Estrela.

O algoritmo A-Estrela também faz uso da função de construção do caminho mínimo (Figura 2). Além disso, o A-Estrela deve utilizar uma heurística admissível. Tendo em vista a utilização de mapas de grade, a heurística adotada foi a Distância de Manhattan, com movimentos na horizontal e na vertical, conforme demonstra a Figura 4.

```
1  Function h(v,alvo) ;
2   Begin
3        int D = 1;
4        int dx = abs(v[x]-alvo[x]);
5        int dy = abs(v[y]-alvo[y]);
6        return (D.(dx+dy));
7   End.
```

Figura 4. Algoritmo da Heurística Distância de Manhattan.

Apesar da função heurística ser $O(1)$, o algoritmo A-Estrela também possui a função de construção do caminho mínimo $O(V)$, que adicionado a sua complexidade de tempo ficaria $O(max(V + VlogV))$, não aumentando a complexidade inicial $O(VlogV)$.

3. Resultados e Conclusões

Os experimentos evidenciaram o comportamento dos algoritmos de Dijkstra e A-Estrela durante a expansão dos nós. Os resultados em um mapa de grade (7x7), onde o nó de partida (verde) e o nó de destino (vermelho) estão nos extremos, são mostrados na Figura 5.

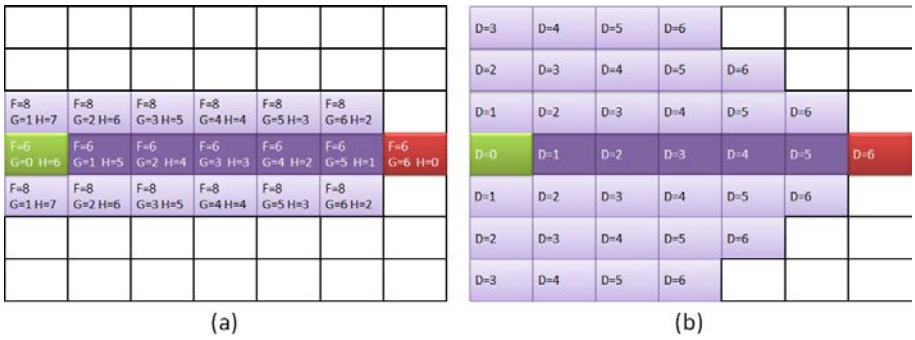

Figura 5. Comportamento dos Algoritmos A-Estrela (a) e Dijkstra (b).

A quantidade de operações realizadas em cada algoritmo diverge significativamente de acordo com o tamanho dos mapas de grade, conforme demonstra a Figura 6.

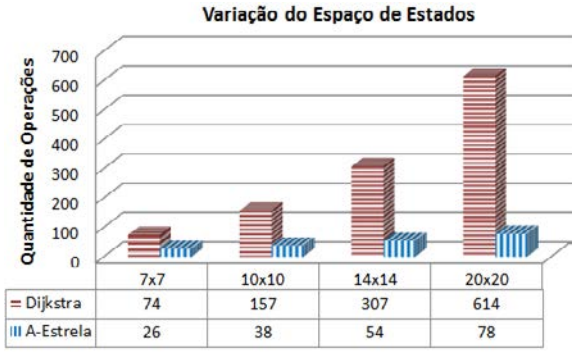

Figura 6. Gráfico Comparativo das Operações em Distintos Mapas de Grade.

Dessa forma, apesar dos algoritmos estudados possuírem, no pior caso, a mesma complexidade de tempo, é possível constatar que o A-Estrela, utilizando um heusrística admissível, resolve o problema caminho mínimo de forma mais eficiente que o Dijkstra.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Djojo, M. and Karyono, K. (2013). Computational load analysis of dijkstra, a*, and floyd-warshall algorithms in mesh network. In *Robotics, Biomimetics, and Intelligent Computational Systems (ROBIONETICS), 2013 IEEE International Conference on*, pages 104–108.

Eraghi, N., Lopez-Colino, F., de Castro, A., and Garrido, J. (2014). Path length comparison in grid maps of planning algorithms: Hctnav, a*; and dijkstra. In *Design of Circuits and Integrated Circuits (DCIS), 2014 Conference on*, pages 1–6.

Russell, S. J. and Norvig, P. (2013). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.