Análise do Desempenho de Algoritmos Clássicos de Busca de Caminho em Ambiente de Navegação

Rafael Castro G. Silva[1], **Rafael S. Parpinelli**[1]

[1]Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina (UDESC) - Joinville - SC - Brasil

rafaelcgs10@gmail.com, rafael.parpinelli@udesc.br

Abstract. *The literature shows a wide variety of pathfinding algorithms for navigation environments. These algorithms are widely used in games, robotics, GPS and many other applications. This paper aims to analyze how the classics algorithms Breadth-First Search, Search with Uniform Cost, A^* and Weighted A * behave in an environment with various costs. It presents an analysis of the objectives of minimizing the path cost and amount of expanded nodes.*

Resumo. *A literatura apresenta uma grande variedade de algoritmos de busca de caminho em ambientes de navegação. Esses algoritmos são amplamente utilizados em jogos, robôs, GPS e diversas outras aplicações. Este trabalho visa analisar como os algoritmos clássicos Busca Cega em Largura, Busca com Custo Uniforme, A^* e Weighted A^* desempenham em um cenário com diversos custos. Apresenta-se uma análise nos objetivos de minimização do custo do caminho e de quantidade de nós expandidos.*

1. Introdução

O problema de encontrar o menor caminho em um ambiente é recorrente em diversas aplicações e problemas cotidianos, por exemplo, a menor rota até o destino que deve ser encontrada por um GPS, um robô que deve procurar um caminho para sair de um labirinto e um personagem em um jogo que deve se aproximar de um inimigo.

As abordagens clássicas como Busca Cega em Largura e Busca com Custo Uniforme são bem conhecidas para busca em grafos. A ideia é expandir os nós da árvore de decisão com base em algum critério. No caso da Busca Cega em Largura os nós são explorados seguindo o conceito de *FIFO* (*First In First Out*), ou seja, os primeiros nós a chegarem são os primeiros a serem explorados. A Busca com Custo Uniforme é uma variação que considera o custo acumulado durante a expansões dos nós, de maneira a dar preferência para os nós com menor custo até então. O algoritmo A* [P. E. Hart and Raphael 1968] é similar a Busca com Custo Uniforme, porém se diferencia pelo uso de uma heurística que poda a árvore de expansão.

Quatro décadas depois do surgimento do A*, os algoritmos sofreram diversos avanços e frequentemente são apresentadas novas abordagens. [Algfoor et al. 2015] apresenta uma revisão sobre o estado da arte dos algoritmos de busca e demonstram que há uma grande variedade de abordagens. Cada abordagem costuma lidar com um caso específico do problema, como espaços de busca tridimensionais, ambientes dinâmicos, variadas topologias de terreno, ponto destino em movimento, tempo limitado de processamento e espaços de busca complexos. Esses tratamentos de casos geralmente são otimizações do A* que aproveitam características em particular do cenário.

O que faz o A* ser uma busca mais direcionada é a função heurística, a qual permite a eliminação de caminhos julgados não promissores. Na procura por reduzir ainda mais o espaço de busca é possível modificar a função heurística por meio de pesos ϵ, como apresentado por [Pohl 1969]. Um ponto negativo a abordagem de pesos é não ser sempre ótima. O custo é limitado por ϵ, ou seja, será no máximo ϵ vezes pior que o caminho ótimo [Pohl 1969].

O peso na função heurística pode ser usado em algoritmos *Anytime*, os quais encontram uma primeira solução sub-ótima rapidamente e continuam trabalhando até que o tempo disponível acabe. [Likhachev et al. 2004] apresenta o algoritmo ARA*, uma versão *Anytime* do A* que calcula o primeiro caminho usando um valor alto para o peso ϵ e gradualmente reduz esse valor, assim cada vez encontra um caminho melhor que o anterior. Se o tempo for suficiente, então o caminho ótimo pode ser provadamente encontrado.

Este trabalho apresenta uma avaliação de como o valor ϵ influencia na qualidade do resultado, ou seja, quão longe é do ótimo. Essa avaliação é feita com duas métricas: a quantidade de nós expandidos na árvore de decisão e o custo do caminho encontrado. Espera-se que os resultados desta avaliação demonstrem o quão eficiente a função modificada pode ser. Os algoritmos analisados foram o algoritmo de Busca Cega em Largura, Busca com Custo Uniforme, A* e *Weighted* A*. O cenário escolhido para os testes é uma matriz 42x42 com cada célula contendo um valor de custo. O objetivo é verificar o desempenho de algoritmos clássicos de busca em um ambiente que simula diferentes tipos de terrenos.

Este trabalho apresenta a seguinte estrutura: na seção 2 são apresentados os algoritmos e as características para o problema a ser resolvido. Na seção 3, apresenta-se a problemática e o cenário onde os experimentos serão realizados. Por fim, na seção 4 os experimentos são apresentados e discutidos.

2. Algoritmos de Busca

Apresenta-se quatro algoritmos de busca de caminho: Busca Cega em Largura (*bfs*), Busca com Custo Uniforme (bfs_u), A* ($astar_m$) e *Weighted* A* ($astar_w$) [Russell and Norvig 2003]. Todos esses algoritmos apresentam a característica de serem completos, isto é, sempre encontram uma solução para o problema. Entretanto, apenas o bfs_u encontra sempre a solução ótima e o A* encontra a ótima se certas restrições forem satisfeitas.

Uma simples otimização é evitar a exploração de nós já explorados. Considera-se que todos os algoritmos em discussão utilizam a não repetição de nós.

2.1. Busca Cega em Largura

A Busca Cega em Largura (*bfs*) é um algoritmo básico que é comumente utilizado para busca de caminho em grafos. Primeiro o nó raiz é visitado, então todos os sucessores da raiz são visitados e todos os sucessores desses nós, e assim por diante. A exploração dos nós acontece como uma fila. Na árvore de expansão, os nós a serem explorados são sempre as folhas (nós de fronteira).

2.2. Busca com Custo Uniforme

A Busca Cega com Custo Uniforme (*bfs*$_u$) é uma variação do algoritmo *bfs*, porém considera o custo acumulado, dado por $g(n)$, até o nó n de fronteira e sempre escolhe explorar primeiro o nó com o menor custo acumulado até o momento. Como consequência, a primeira vez que o nó objetivo for encontrado, esse nó também será o nó com menor custo. Isso garante que este algoritmo é ótimo, logo sempre encontra o caminho de menor custo. O custo utilizado é o valor associado a cada célula no cenário.

2.3. A*

O algorimto A* é o resultado da junção de um algoritmo guloso, o qual escolhe primeiro a melhor solução aparente no momento e o algoritmo de Busca com Custo Uniforme cuja solução é sempre ótima. O que define a melhor escolha aparente é uma estimativa $h(n)$ do custo até o objetivo. No caso que será tratado, uma estimativa da distância até o nó alvo. A função heurística $h(n)$ garante uma redução na quantidade de nós explorados, pois direciona a busca até o objetivo.

Ainda que *bfs*$_u$ seja ótimo, isso não implica que o A* também seja. Como dito anteriormente, há algumas condições para o A* ser ótimo. Primeiro, a estimativa deve ser admissível, que significa nunca ser superior ao valor real do custo ótimo. Este é o ponto que diferencia o A* do *Weighted* A*. Segundo, a estimativa deve ser consistente. De acordo com [Russell and Norvig 2003, p. 95] uma heurística $h(n)$ é consistente se, para todo nó n e para todo nó n' de n gerado por qualquer ação a, o custo estimado de alcançar o objetivo a partir de n nunca é maior que o custo de chegar até n' mais o custo de alcançar o objetivo a partir de n".

A função heurística utilizada é uma estimativa da distância do nó atual até o nó objetivo. Pode-se utilizar qualquer distância neste algoritmo, o importante é que a distância escolhida não superestime o real valor do custo ótimo. Como o robô não pode movimentar-se diagonalmente, então foi escolhido utilizar a distância de manhattan, utilizada no *astar*$_m$ e *astar*$_w$, dada pela equação 1.

$$dz = |dx| + |dy| \tag{1}$$

2.4. *Weighted* A*

Superestimar o custo da distância é uma maneira de direcionar ainda mais a busca, uma vez que o o algoritmo passa a prestar menos atenção no custo já acumulado e mais nos nós que tem uma menor distância até o objetivo [Millington and Funge 2009].

Ainda que o uso do peso quebre a condição de ser admissível, existe um limite para quão distante do ótimo a solução encontrada está. Se o peso for um valor ϵ, então o custo será no máximo ϵ vezes pior que o custo ótimo [Pohl 1969].

3. Problemática

O problema consiste em encontrar o melhor caminho entre dois pontos arbitrários em uma matriz nxn, onde há diversos tipos de terrenos com diferentes custos. Os tipos de terrenos são:

- Sólido e plano (verde) – Custo: 1

- Montanhoso (marrom) – Custo: 5
- Pântano (azul) – Custo: 10
- Fogo (vermelho) – Custo: 15

A Figura 1 é um exemplo de cenário de dimensão 42x42 e é o que será utilizado nos testes.

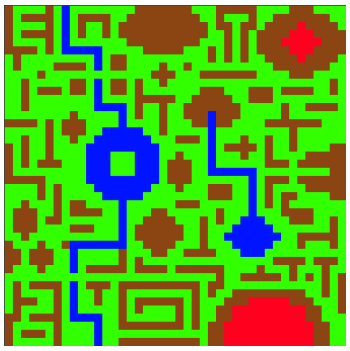


Figura 1. Cenário do experimento

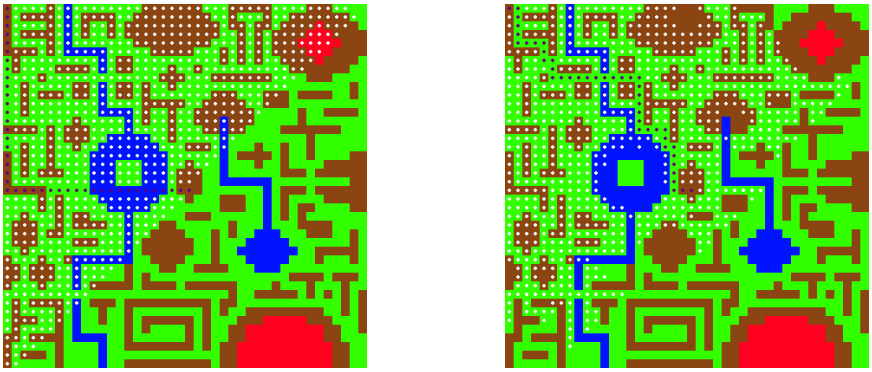
O objeto que irá se deslocar pelo cenário é um robô que pode apenas se movimentar na horizontal e na vertical. Ou seja, não é permitido movimentos diagonais.
A exploração dos nós vizinhos ao robô acontece seguindo a seguinte ordem: norte, leste, sul e oeste. O objetivo é encontrar o melhor caminho possível, se não o próprio caminho ótimo.

4. Experimentos e Resultados

O hardware utilizado nos experimentos foi um *Intel Core i*5-3450 3.10*GHz*, *Kingston* 2x4*GB DDR*3 1600*MHz* e os algoritmos foram todos implementados na linguagem *Python* 2.7.

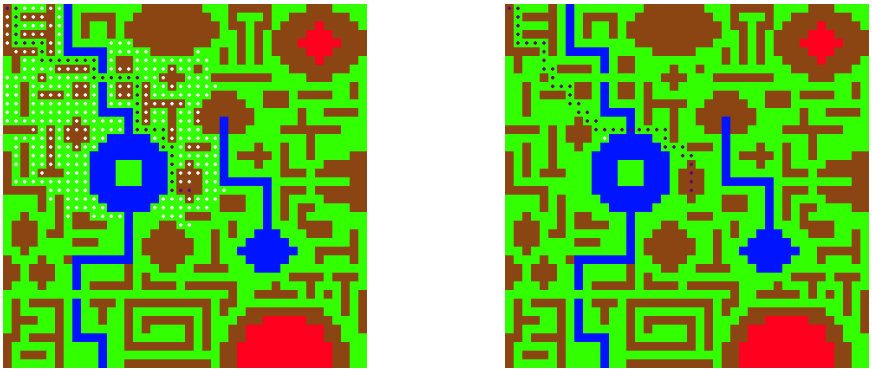
Por meio de testes empíricos determinou-se que o melhor valor de ϵ, para o cenário apresentado, é 5. Assim, a função heurística do *Weighted* A* (*astar*$_w$) é $5h(n)$, sendo $h(n)$ o custo estimado fornecido pela distância de manhattan.

Ilustrações dos algoritmos *bfs*, *bfs*$_u$, *astar*$_m$ e *astar*$_w$ podem ser observada nas Figuras 2 e 3.Em todas as figuras a posição de partida é o ponto superior esquerdo $(0, 0)$ e o de chegada é o centro do cenário (21, 21). Em branco podem ser observados os nós expandidos e em preto está ressaltado o caminho final gerado por cada abordagem. A figura 2 mostra que as buscas não heurísticas não são direcionadas e expandem muitos caminhos não improváveis. Por outro lado, os algoritmos A* da figura 3 são bem mais direcionados, por causa da função heurística. Especialmente o *astar*$_w$, o qual expande poucos nós além dos que fazem parte do caminho obtido.



(a) *bfs* (b) bfs_u

Figura 2. Resultados do algoritmos básicos



(a) astar_m (b) astar_w

Figura 3. Resultados dos Algoritmos A*

Outros experimentos foram realizados para analisar como os custos e a quantidade de nós expandidos crescem conforme a distância entre as posições de início e fim é ampliada. Uma sequência de pares de posições foi gerada da seguinte maneira: a sequência começa com 100 pares aleatórios e únicos de posições cuja distância de manhattan é 0, os próximos 100 pares de posições têm distância de manhattan 2 e assim por diante. A distância cresce com passo 2 de 0 até 40. Logo, ao todo são 2100 pares de pontos aleatórios e únicos utilizados nos testes.

A Figura 4 apresenta as médias dos custos com desvio padrão (eixo-y) e a distância (eixo-x). As curvas bfs_u e $astar_m$ aparecem sobrepostas pois ambos algoritmos são ótimos, possuindo sempre o mesmo custo. O fato interessante que este gráfico apresenta é que o *Weighted* A* ($astar_w$) se distancia cada vez mais do custo ótimo, mas em média nunca chega a ser 5 vezes maior que o ótimo, indicando que é possível direcionar mais a busca sem prejudicar muito o custo.

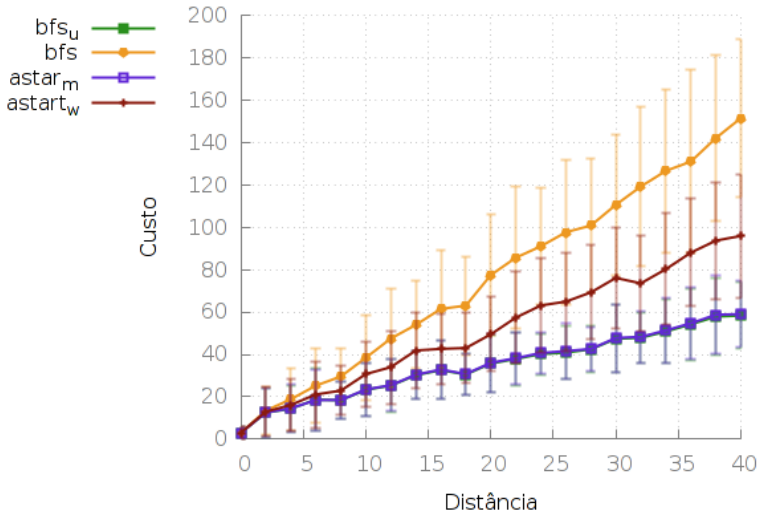


Figura 4. Gráfico dos custos por distância

Algo que se espera do *Weighted* A* (*astar*$_w$) é que ele sempre explore poucos nós, já que é mais direcionado. A figura 5 mostra no eixo-y a quantidade média de nós explorados e o desvio padrão, e a distância no eixo-x, confirma essa expectativa e mostra que há pouca variação de nós explorados para uma mesma distância. A discrepância do *astar*$_w$ em relação aos demais é bastante grande, pois em média sempre a mesma quantidade de nós são visitados.

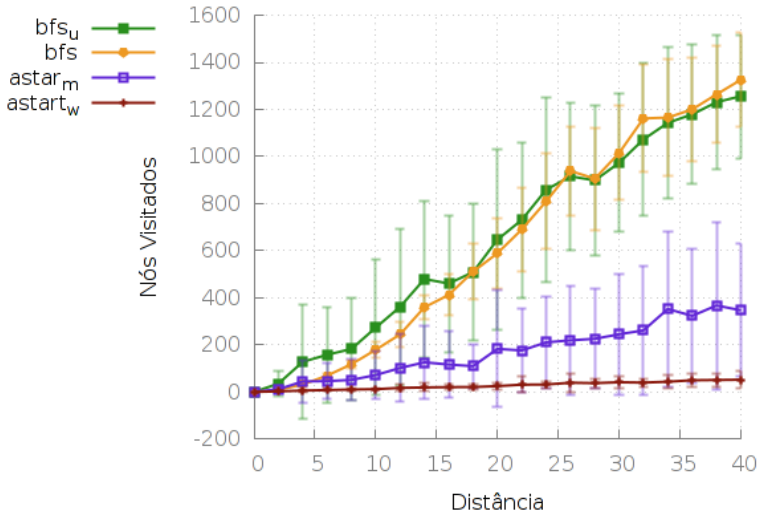


Figura 5. Gráfico dos nós visitados por distância

Uma outra maneira de analisar os testes é por meio do gráfico de Pareto apresentado na figura 6, o qual relaciona a média de nós explorados (eixo-y) com a média dos custos (eixo-x) para uma distância de valor 40. Diz-se que um algoritmo domina outro se o primeiro for melhor do que o segundo em todos os objetivos considerados. Na Figura 6 os objetivos considerados são minimizar o custo do caminho (eixo-x) e minimizar a quantidade de nós expandidos (eixo-y). Sendo assim, é possível visualizar que, dentre

os algoritmos que utilizam alguma informação para guiar a busca, não existe algum que domine todos os outros em ambos objetivos. Desta forma, o conjunto ótimo de Pareto é formado pelos algoritmos bfs_u, $astar_m$ e $astar_w$. O único algoritmo dominado por todos os outros é o *bfs*. Dentre os algoritmos não-dominados, o $astar_m$ se mostrou com o melhor compromisso em atender ambos objetivos.

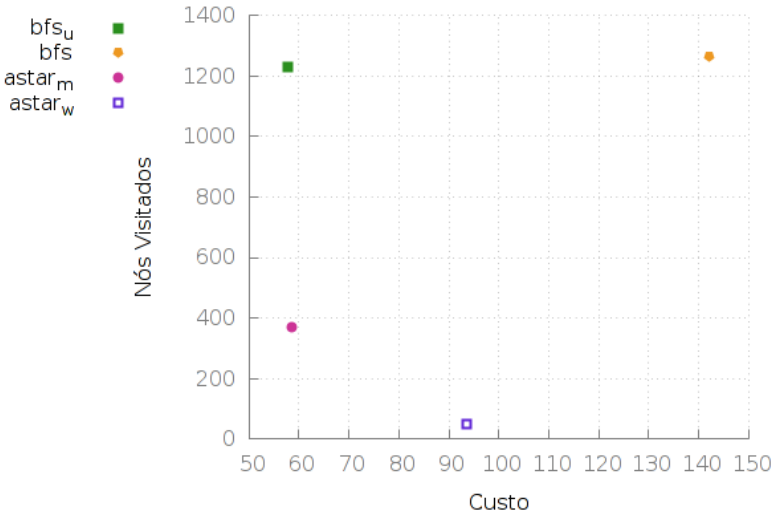


Figura 6. Gráfico de Pareto para distância 40

5. Conclusão

Este trabalho explicou as principais características de quatro algoritmos clássicos de busca, apresentou uma problemática comum a diversas aplicações e avaliou os algoritmos nos objetivos de minimização de custo e quantidade de nós expandidos.

Os testes mostraram dois resultados importantes. O primeiro, que não há um único algoritmo, dos apresentados neste artigo, que domine todos os outros em ambos objetivos. E segundo, que a função heurística é um ponto chave para a otimização dos algoritmos de busca. A melhor estimativa da distância está relacionada com o cenário do problema a ser tratado. Ainda que o uso do peso na função heurística tenha o efeito colateral de resultados sub-ótimos, é visível que para os teste realizados os caminhos encontrados eram bastante satisfatórios e o baixo custo computacional pode justificar o seu uso.

Como trabalhos futuros pode-se citar a aplicação do algoritmo A* como determinante do comportamento de um agente em jogos e a plicação em cenários mais complexos, por exemplo, com a existência de obstáculos.

Referências

Algfoor, Z. A., Sunar, M. S., and Kolivand, H. (2015). A comprehensive study on pathfinding techniques for robotics and video games. *Int. J. Comput. Games Technol.*, 2015.

Likhachev, M., Gordon, G. J., and Thrun, S. (2004). Ara* : Anytime A* with provable bounds on sub-optimality. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 767–774. MIT Press.

Millington, I. and Funge, J. (2009). *Artificial Intelligence for Games, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition.

P. E. Hart, N. J. N. and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107.

Pohl (1969). *First Results on the Effect of Error in Heuristic Search*. MIP-R-. Edinburgh University, Department of Machine Intelligence and Perception.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.