Variações do Algoritmo A*

Carlos Renato de Andrade Figueiredo
Engenharia da Computação
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos, Brasil
Email: carlos.figueiredo@ga.ita.br

Samara Ribeiro Silva
Engenharia da Computação
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos, Brasil
Email: samara.silva@ga.ita.br

***Resumo* - Este relatório descreve os resultados da simulação e comparação entre as seguintes variações do algoritmo A*: A* ponderado com peso estático e dinâmico, A* pwXD, A* pwXU e o T*. Observou que a escolha do melhor algoritmo deve-se levar em consideração o *trade-off* entre o tempo computacional gasto e o custo do caminho encontrado. O resultado obtido pelo A* pwXD é promissor se comparado com o A* pois possui um bom desempenho no quesito tempo sem penalizar demasiadamente o quesito custo do caminho. O algoritmo T* pode ser utilizado quando o caminho não precisa seguir o *grid* podendo assim explorar linhas retas mudando de direção majoritariamente nas quinas dos obstáculos.**

***Palavras-chave* - A*, Dijkstra, variações do A*, T*, A* ponderado**

I. Introdução

Os algoritmos de busca em grafos podem ser aplicados para encontrar solução de diversos problemas de engenharia e logística. Eles também são aplicado em jogos e navegação de robôs móveis para encontrar o melhor caminho entre um local e outro. Existem vários algoritmos para realizar a busca do melhor caminho em grafos, como a exploração em largura (*Breath-First Search (BFS)*) e profundidade (*Depth-First Search (DFS)*), o algoritmo de Dijkstra, busca gulosa (*Greedy Best-First-Search*), A* etc.

O algoritmo A* foi desenvolvido em 1968 [1] e combina as estratégias do algoritmo de Dijktra e a busca gulosa para encontrar o melhor caminho utilizando como função de estimativa de custo total $f(n)$ a soma do custo acumulado $g(n)$ e o valor da função heurística $h(n)$, conforme 1.

$$f(n) = g(n) + h(n) \tag{1}$$

O A* pode ser implementado seguindo os seguintes passos:

1) Inicialize as variáveis correspondentes ao nó inicial $g(n_{inicial}) = 0$ e portanto $f(n_{inicial}) = h(n_{inicial})$ e para os demais nós $f(n) = \infty$;
2) Inicialize uma fila de prioridade Q e insira o nó inicial;
3) Enquanto Q não estiver vazia:
   a) Extraia o nó n com menor valor de f e o marque como encerrado;
   b) Se n for o nó objetivo, retorne n;
   c) Caso contrário, para cada nó sucessor $n_{sucessor}$ de n que não esteja marcado como encerrado é verificado se $f(n_{sucessor} > g(n) + custo(n, n_{sucessor}) + h(n_{sucessor})$:
      i) n é marcado como o pai do $n_{sucessor}$;
      ii) atualiza-se os valores de $g(n_{sucessor})$ e $h(n_{sucessor})$; e
      iii) $n_{sucessor}$ é inserido em Q.

A introdução da função heurística $h(n)$ é a grande diferença entre o algoritmo A* e o algoritmo de Dijkstra. Ela permite uma melhora no desempenho do algoritmo adicionando informações específicas do domínio de cada problema e se modelada adequadamente garante uma solução ótima com o mesmo custo da solução encontrada com o uso do algoritmo de Dijkstra. Em [1] é possível encontrar a prova detalhada da garantia da solução ótima quando utiliza-se $h(n)$ consistente.

A distância euclidiana é comumente utilizada como função heurística para problemas encontrar melhor caminho em jogos, principalmente quando o *grid* é 8-conectado (é possível mover-se na diagonal). Quando o *grid* é 4-conectado normalmente utiliza-se a distância Manhattan, conforme 2.

$$h(n) = |n_x - sucessor_x| + |n_y - sucessor_y| \tag{2}$$

O algoritmo A* é muito versátil e possui muitas variações [2]. Este trabalho tem como objetivo abordar as seguintes das variações do algoritmo A*:

1) A* ponderado (*Weighted A**) estático e dinâmico;
2) pwXU;
3) pwXD; e
4) T*.

Com o intuito de otimizar a busca pelo melhor caminho alterou-se a função custo total do A*, conforme 3, multiplicado a função heurística por um peso constante $w \in (1, \infty)$, surgindo assim a versão ponderada do A*.

$$f(n) = g(n) + wh(n) \tag{3}$$

Observe que quando $w \to 0$ o algoritmo se aproxima do algoritmo Dijkstra e quando $w \to \infty$ o algoritmo terá uma abordagem gulosa se aproximando do *Greedy Best-First-Search*, ou seja, o incremento demasiado do valor de w pode não ser adequado e pode resultar em soluções subótimas, conforme [5].

Para evitar que o algoritmo A* ponderado termine sem encontrar a solução ótima, [5], pode-se utilizar a ponderação dinâmica onde o peso não será constante e dependerá do nó atual, onde $w(n)$ assume valor conforme 4 quando a profundidade da busca$d(n)$ é menor que limite superior da profundidade da busca N definida no início da busca e assume valor nulo para qualquer outra condição.

$$w(n) = 1 + \epsilon \left(1 - \frac{d(n)}{N}\right) \tag{4}$$

Note que no quanto mais próximo do nó objetivo $w(n) \to 1$ e portanto o algoritmo tende a comporta-se igual ao A* sem ponderação uma vez que $f(n) \approx g(n) + h(n)$.

Existem várias maneiras controlar a influência da função heurística na função custo total. Em [6] é possível encontrar duas abordagens: pwXD (*piecewise Convex Downward*) e pwXU (*piecewise Convex Upward*).

No pwXD a influência da função heurística $h(n)$ aumenta quando aproxima-se do nó objetivo. A função custo total $f(n)$ pode ser calculada conforme 5.

$$f(n) = \begin{cases} g(n) + h(n); \text{ se } g(n) < h(n) \\ \frac{g(n)+(2w-1)h(n)}{w}; \text{ se } g(n) > h(n) \end{cases} \tag{5}$$

Já para o pwXU o valor da função custo total $f(n)$ 6 sofre uma menor influência da função heurística $h(n)$ para posições mais próximas do nó objetivo.

$$f(n) = \begin{cases} \frac{g(n)}{(2w-1)} + h(n); \text{ se } g(n) < (2w-1)h(n) \\ \frac{g(n)+h(n)}{w}; \text{ se } g(n) > (2w-1)h(n) \end{cases} \tag{6}$$

A última variação abordada neste trabalho é o algoritmo T* que consiste em uma variação do A* para encontrar o melhor caminho quando o caminho não precisa seguir o *grid*. De acordo com [2], o desempenho do A* seria melhor se um gráfico com os pontos principais, como as quinas dos obstáculos, fosse conhecido. Para eliminar a necessidade de um pré-processamento para gerar esse gráfico utiliza-se o T* que analisa o mapa em *grid*, mas não encontra um caminho que segue esse *grid*.

A principal diferença dessa variação é que um nó pode ser sucessor de outro sem estar conectado a ele. É introduzido o conceito de linha de visão, se um nó ”enxerga” o outro (se não há obstáculos entre esses nós) não é necessário analisar os nós intermediários. Assim, esse algoritmo segue um caminho que tangencia as quinas dos obstáculos que estejam entre o nó inicial e o nó objetivo.

II. Implementação Desenvolvida

O código foi desenvolvido em linguagem Python e utilizou-se a biblioteca NumPy. Para a implementação das variações do A* seguiu-se os passos do pseudocódigo supracitado e utilizou-se como base a estrutura do código fornecido no segundo laboratório da disciplina de Inteligência Artificial para Robótica Móvel (CT-213) [7], fazendo as adaptações necessárias.

O projeto é composto de três arquivos, *grid.py*, *main_.py* e *path_planner.py*. O *grid.py* é responsável pela criação do mapa e estruturas auxiliares, o *main.py* é o arquivo que executa o planejamento do caminho, traça os gráficos dos caminhos encontrados e calcula a média e o desvio padrão do tempo computacional e do custo do caminho. Por fim, é no arquivo *path_planner.py* que estão foram implementados os métodos com os algoritmos tradicionais estudados em sala de aula(*dijkstra*, *greedy* e A*) e as cinco variantes do A*.

O A* ponderado estático foi implementado alterando o valor de $f(n)$ do A* conforme 3 e foi utilizado $w = 5$. Para A* ponderado dinâmico foi adicionado o cálculo de $w(n)$ conforme 4, utilizando $d(n) = N - h(n)$ assim quanto mais próximo do nó objetivo $w(n) \to 1$ e portanto $f(n) \approx g(n) + h(n)$.

Assim com no A* ponderado estático a implementação do pwXD e pwXU consistiu em apenas alterar o valor da função $f(n)$ conforme 5 e 6 respectivamente, com $w = 5$.

Para o T* foi necessário criar o método *line_of_sight*, que retorna um valor booleano verdadeiro se dois parâmetros estão em linha de visão sem bloqueio, e retorna falso caso dois elementos não estejam em linha de visão. O método *theta_star* age de forma parecida com o *a_star*, porém antes de fazer a ligação de um ponto com o seu sucessor é verificado se o antecessor daquele ponto possui linha de visão com o sucessor, caso positivo, o ponto pode ser descartado e é ligado o antecessor daquele ponto diretamente com o sucessor [8].

III. Resultados e Discussões

Para encontrar um mapa com a disposição dos obstáculos convenientes para testar os algoritmos alterou-se a *seed(16)* da função aleatório do código fornecido em [7] e optou-se por deixar nó objetivo fixo na posição $(x = 100, y = 40)$ nas simulações representadas nas figuras 1 a 8.

É possível observar nas figuras 4 a 7 o comportamento do caminho encontrado é uma mescla entre o A* e o *greedy* isso ocorre devido a mudança da influência da função heurística $h(n)$ em cada uma das funções custo total $f(n)$. Note, também, que o caminho da figura 8 correspondente ao algoritmo T* está de acordo com o esperado teoricamente, visto que não segue o *grid* e contorna o obstáculo buscando as quinas do mesmo.

Na tabela 1 estão representados os dados de tempo computacional e Custo do caminho para um simulação de Monte Carlo de 100 repetições. Para essa simulação a posição do nó objetivo não foi fixada, sendo assim gerada aleatoriamente.

Note que todas variações do A* abordadas neste trabalho obteve um desempenho melhor que o A* se considerarmos o tempo computacional. A escolha do melhor algoritmo deve-se estabelecer um *trade-off* entre o tempo computacional e o custo do caminho. Observe que o A* pwXD possui um excelente desempenho, visto que gasta um tempo computacional de aproximadamente 60% menor que o A* e encontra um caminho com o custo cerca de 13% maior. O bom desempenho do pwXD já era esperado teoricamente e pelos resultados encontrados por [6].

Tabela 1: Comparação entre os algoritmos de planejamento em termos de tempo computacional e custo do caminho

	Compute time	Cost
dijkstra	mean: 0.162 std: 0.093	mean: 79.37 std: 38.98
greedy	mean: 0.007 std: 0.004	mean: 110.19 std: 69.92
a_star	mean: 0.047 std: 0.043	mean: 79.37 std: 38.98
a_star_weight	mean: 0.010 std: 0.007	mean: 99.45 std: 56.04
a_star_dynamic_weight	mean: 0.009 std: 0.006	mean: 100.06 std: 55.81
a_star_pwXU	mean: 0.011 std: 0.006	mean: 100.817 std: 57.51
a_star_pwXD	mean: 0.020 std: 0.020	mean: 89.730 std: 47.81
theta_star	mean: 0.030 std: 0.022	-

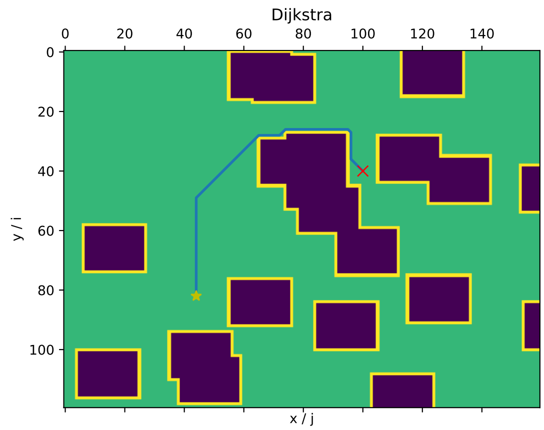

Fig. 1. Caminho calculado pelo algoritmo de Dijkstra. $Tempo = 0.277$ e $Custo = 109.598$. Fonte: autor.

O custo do caminho do T* não foi representado na Tabela 1, visto que como este algoritmo não segue a *grid* o seu custo não segue os mesmo parâmetros dos demais algoritmos. Uma aproximação do seu custo seria a distância percorrida. Como já discutido anteriormente a escolha desse algoritmo dependerá dos requisitos da aplicação.

IV. Conclusão

Este trabalho abordou algumas variações do algoritmo A* analisando o tempo computacional e o custo do caminho encontrado. Foi possível observar que com algumas alterações na função custo total pode-se diminuir expressivamente o tempo computacional gasto com um aumento relativamente pequeno no custo do caminho.

Entre os algoritmos estudados destaca-se o A* pwXD que obteve uma marca de tempo computacional de aproximadamente 60% menor com um aumento de cerca de 13% no custo do caminho se comparado com os números obtidos pelo algoritmo A*. Vale ressaltar também o T*, que pode ser utilizado em casos no qual o caminho não precisa seguir o *grid* e não está disponível um gráfico com as informações de

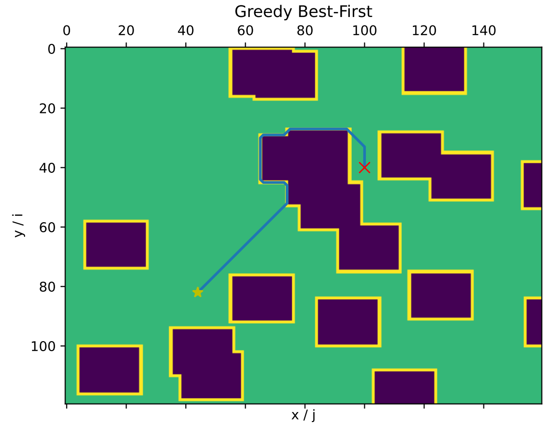

Fig. 2. Caminho calculado pelo algoritmo greedy. $Tempo = 0.016$ e $Custo = 180.882$. Fonte: autor.

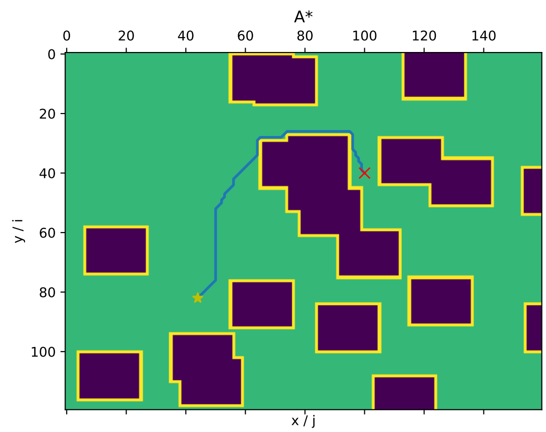

Fig. 3. Caminho calculado pelo algoritmo do A*. $Tempo = 0.169$ e $Custo = 109.598$. Fonte: autor.

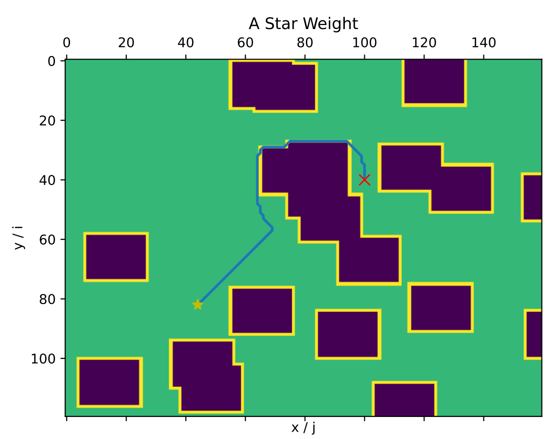

Fig. 4. Caminho calculado pelo algoritmo do A* com peso estático. $Tempo = 0.0534$ e $Custo = 144.639$. Fonte: autor.

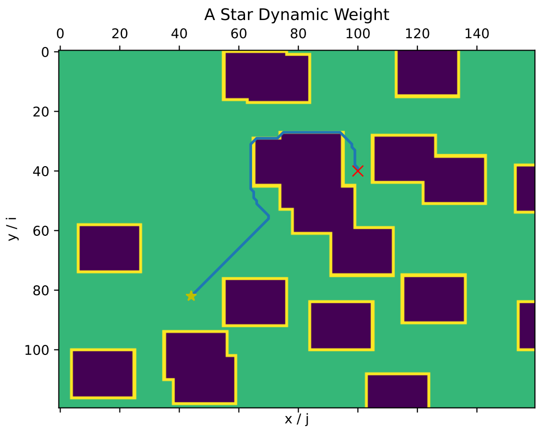

Fig. 5. Caminho calculado pelo algoritmo do A* com peso dinâmico. $Tempo = 0.062$ e $Custo = 144.468$. Fonte: autor.

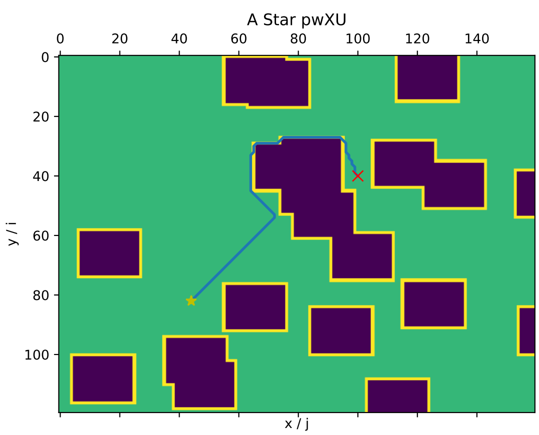

Fig. 6. Caminho calculado pelo algoritmo do A* pwXU. $Tempo = 0.047$ e $Custo = 148.125$. Fonte: autor.

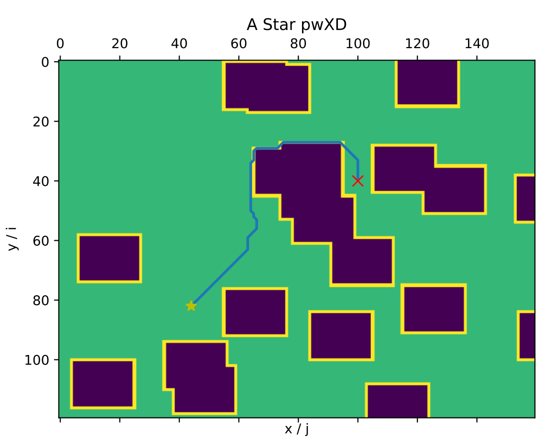

Fig. 7. Caminho calculado pelo algoritmo do A* pwXD. $Tempo = 0.073$ e $Custo = 144.154$. Fonte: autor.

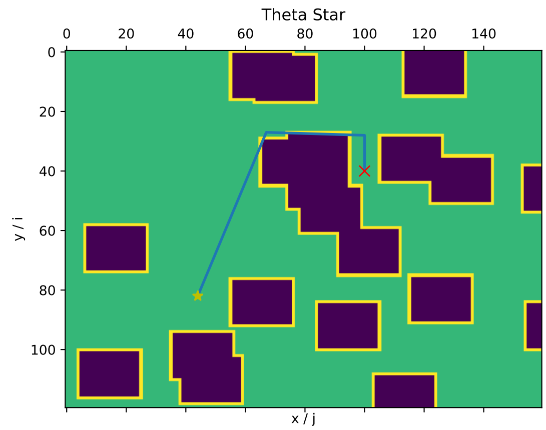

Fig. 8. Caminho calculado pelo algoritmo do T*. $Tempo = 0.100$. Fonte: autor.

pontos importantes do mapa como as quinas dos obstáculos. O uso do T* dispensa a necessidade de pré-processamento do mapa para obter esse gráfico.

REFERENCES

[1] P. E. Hart, N. J. Nilsson and B. Raphael, ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968, doi: 10.1109/TSSC.1968.300136.

[2] A. Patel. Variants of A*, [online] Available: http://theory.stanford.edu/~amitp/GameProgramming/Variations.html.

[3] A. Patel. Introduction to A*, [online] Available: https://www.redblobgames.com/pathfinding/a-star/introduction.html.

[4] C. Wilt, W. Ruml. When does Weighted A* Fail?, [online] Available: https://www.cs.unh.edu/~ruml/papers/wted-astar-socs-12.pdf.

[5] I. Pohl. The Avoidance Of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting And Computational Issues In Heuristic Problem Solving, [online] Available: https://www.cs.auckland.ac.nz/courses/compsci709s2c/resources/Mike.d/Pohl1973WeightedAStar.pdf.

[6] J. Chen., N. Sturtevant. Necessary and Sufficient Conditions for Avoiding Reopenings in Best First Suboptimal Search with General Bounding Functions, [online] Available: https://webdocs.cs.ualberta.ca/~nathanst/papers/chen2021general.pdf.

[7] M. Máximo. Laboratório 2 - Busca Informada. CT-213 - Inteligência Artificial para Robótica Móvel. Divisão de Engenharia da Computação. Instituto Tecnológico de Aeronáutica. São José dos Campos.

[8] K. Daniel, A. Nash, S. Koenig, A. Felner. Theta*: Any-Angle Path Planning on Grids, [online] Available: https://arxiv.org/ftp/arxiv/papers/1401/1401.3843.pdf.