*Algorytm A**

Wykonali:
Jakub Junkiert
Monika Pawlak

Politechnika Poznańska

Poznań 2021

Spis treści

1.

1. Wstęp

Algorytm A* jest heurystycznym algorytmem służącym do znajdowania najkrótszej ścieżki w grafie. Jest to algorytm zupełny i optymalny, co oznacza, że znajduje ścieżkę, jeśli tylko taka istnieje, i przy tym jest to ścieżka najkrótsza. W metodzie tej istnieje zorganizowany model pamięciowy, który gwarantuje, że każdy punkt może zostać odwiedzony. A* jest przykładem metody „najpierw najlepszy". Algorytm A* działa najlepiej, gdy przestrzeń przeszukiwań jest przestrzenią drzewiastą.

Działanie algorytmu oparte jest na minimalizacji funkcji celu *f(x)*, zdefiniowanej jako suma funkcji kosztu *(g(x))* oraz funkcji heurystycznej *(h(x))*:

$$f(x) = g(x) + h(x)$$

W każdym kroku algorytm A* przedłuża już utworzoną ścieżkę o kolejny wierzchołek grafu, wybierając taki, w którym wartość funkcji będzie najmniejsza.

Funkcja g(x) określa rzeczywisty koszt dojścia do punktu x (suma wag krawędzi, które należą już do ścieżki oraz wagi krawędzi łączącej aktualny węzeł z x).

Funkcja h(x) jest to funkcja zwana funkcją heurystyczną. Szacuje ona (zawsze optymistycznie) koszt dotarcia od punktu x do wierzchołka docelowego.

Prawidłowa funkcja heurystyczna musi spełniać dwa warunki:

- warunek dopuszczalności : $g(x) + h(x) \leq g(x_t)$
- warunek monotoniczności: $g(x_j) + h(x_j) \geq g(x_i) + h(x_i)$, gdzie $j > i$, a x_t oznacza punkt początkowy

Warunek dopuszczalności to wymieniony już wcześniej nadmierny optymizm. Funkcja heurystyczna to taka „dobra wróżka”, która niedoszacowuje gdy minimalizujemy koszt, a przeszacowuje gdy maksymalizujemy zysk.

Warunek monotoniczności mówi o tym, że im bardziej przybliżamy się do rozwiązania, tym oszacowanie musi być coraz mniej optymistyczne.

2. Cechy algorytmu A*

Algorytm A* jest kompletny – w każdym przypadku znajdzie optymalną drogę i zakończy działanie, jeśli taka droga istnieje.

Algorytm A* jest optymalny dla danej heurystyki, co znaczy, że nie istnieje inny algorytm, który z pomocą tej samej heurystyki odwiedziłby mniej wierzchołków niż A*.

Dlaczego algorytm A* jest dopuszczalny oraz optymalny obliczeniowo?

Zakładając, że używana w algorytmie heurystyka jest dopuszczalna, możemy stwierdzić, że A* jest dopuszczalny. Oznacza to, że zawsze poda nam prawidłowe rozwiązanie, jeżeli takowe istnieje. Co więcej, algorytm ten podczas działania przeszukuje mniej węzłów niż jakikolwiek inny dopuszczalny algorytm przeszukiwania z taką samą heurystyką. Dzieje się tak, ponieważ A* tworzy „optymistyczne” oszacowanie kosztu ścieżki przez każdy węzeł, który bierze pod uwagę – optymistyczny w tym sensie, że prawdziwy koszt ścieżki przez dany węzeł do węzła końcowego będzie co najmniej tak duży jak oszacowanie.

Kiedy A* kończy przeszukiwanie, ma (z definicji) znalezioną ścieżkę, której koszt jest mniejszy niż szacowany koszt jakiejkolwiek innej ścieżki. Ponieważ oszacowania są optymistyczne, algorytm A* może bezpiecznie zignorować te inne ścieżki. Innymi słowy, A* nigdy nie „przeoczy” ścieżki o niższym koszcie i dlatego jest dopuszczalny.

3. Zastosowanie

Algorytm A* w analizie składniowej PCFG

Algorytm A* znajduje zastosowanie w analizie składniowej bezkontekstowych gramatyk probabilistycznych (PCFG) w celu przyspieszenia analizy przy zachowaniu poprawności wyniku – w przeciwieństwie do niektórych metod PCFG, algorytm A* zwróci zawsze drzewo Viterbiego, czyli o maksymalnym możliwym prawdopodobieństwie, w przeciwieństwie do metod „best-first" oraz „finite-beam", które tego nie gwarantują.

Algorytm A* w tym zastosowaniu zaproponowali Dan Klein i Christopher Manning.

4. Schemat działania programu

1. Zaczynamy w punkcie startowym, dodajemy go do listy otwartych
2. Szukamy dostępnych przyległych pól i liczymy dla nich g(x) oraz h(x).
3. Pole startowe dodajemy do listy zamkniętych.
4. Wybieramy spośród wszystkich dostępnych punktów ten o najmniejszym współczynniku f. i dodajemy go do listy otwartych. Rodzica dodajemy do listy zamkniętych.
5. Kontynuujemy poszukiwania, postępując wg. algorytmu.
6. Jeżeli węzeł w otoczeniu rodzica jest już na liście zamkniętych, to sprawdzamy czy z aktualnego węzła nie wygenerujemy lepszej (niższy koszt) ścieżki dla niego.
7. Algorytm kończy działanie w momencie kiedy punkt końcowy zostanie osiągnięty lub gdy lista otwartych będzie pusta (brak ścieżki do celu).
8. Ścieżka zostaje wyznaczona poprzez ruch od punktu końcowego, po kolejnych rodzicach, aż do punktu początkowego.

5. Graficzna prezentacja działania algorytmu

Rysunek przedstawia graficzną reprezentację działania algorytmu. Na zielono zaznaczono węzeł startowy, liczby w kolejnych węzłach oznaczają długość drogi dzielącej węzeł startowy i aktualny węzeł.