Sztuczna inteligencja. Uzupełnienie do Wykładów 3 i 4. A*

Paweł Rychlikowski

Instytut Informatyki UWr

30 marca 2022

Algorytm A*

Definicje

- $g(n)$ – koszt dotarcia do węzła n
- $h(n)$ – szacowany koszt dotarcia od n do (najbliższego) punktu docelowego ($h(s) \geq 0$)
- $f(n) = g(n) + h(n)$

Algorytm

Przeprowadź przeszukanie, wykorzystując $f(n)$ jako priorytet węzła (czyli rozwijamy węzły od tego, który ma najmniejszy f).

Kontury A*

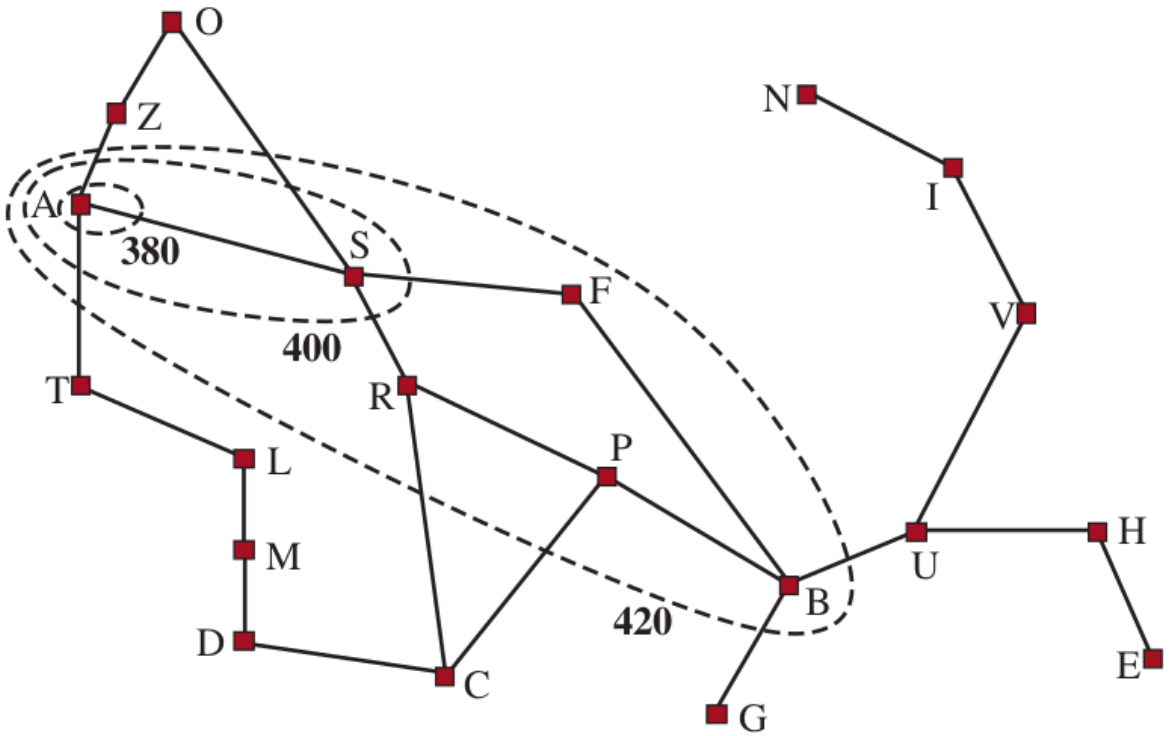


Figure 3.20 Map of Romania showing contours at $f = 380$, $f = 400$, and $f = 420$, with Arad as the start state. Nodes inside a given contour have $f = g + h$ costs less than or equal to the contour value.

Heurystyki dla ósemki

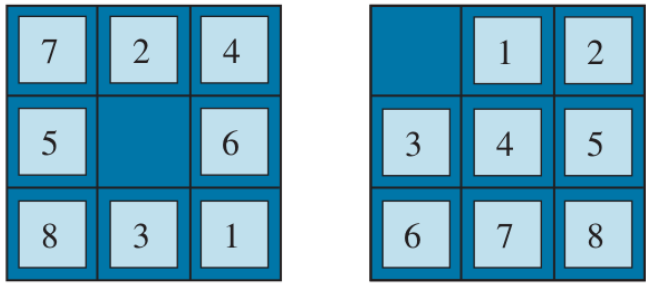


Figure 3.25 A typical instance of the 8-puzzle. The shortest solution is 26 actions long.

Heurystyki dla ósemki

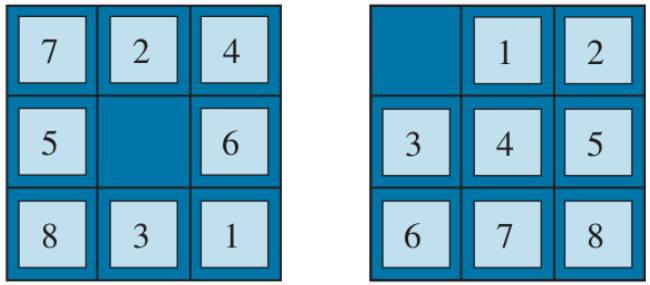


Figure 3.25 A typical instance of the 8-puzzle. The shortest solution is 26 actions long.

Pomysł 1

Jak coś jest nie na swoim miejscu, to musi się ruszyć o co najmniej 1 krok. Zliczajmy zatem, ile kafelków jest poza punktem docelowym ($h_1(s) = 8$)

Pomysł 2

Jak coś jest nie na swoim miejscu, to musi pokonać cały dystans do punktu docelowego. Zliczajmy zatem, ile kroków od celu jest każdy z kafelków ($h_2(s) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$)

Efektywność w praktyce

	Search Cost (nodes generated)			Effective Branching Factor		
d	BFS	A*(h_1)	A*(h_2)	BFS	A*(h_1)	A*(h_2)
6	128	24	19	2.01	1.42	1.34
8	368	48	31	1.91	1.40	1.30
10	1033	116	48	1.85	1.43	1.27
12	2672	279	84	1.80	1.45	1.28
14	6783	678	174	1.77	1.47	1.31
16	17270	1683	364	1.74	1.48	1.32
18	41558	4102	751	1.72	1.49	1.34
20	91493	9905	1318	1.69	1.50	1.34
22	175921	22955	2548	1.66	1.50	1.34
24	290082	53039	5733	1.62	1.50	1.36
26	395355	110372	10080	1.58	1.50	1.35
28	463234	202565	22055	1.53	1.49	1.36

Figure 3.26 Comparison of the search costs and effective branching factors for 8-puzzle problems using breadth-first search, A* with h_1 (misplaced tiles), and A* with h_2 (Manhattan distance). Data are averaged over 100 puzzles for each solution length d from 6 to 28.

Rezygnacja z optymizmu

Weighted A^* search

$f(n) = g(n) + W \times h(n)$, dla $W > 1$

Rezygnacja z optymizmu

Weighted A^* search

$f(n) = g(n) + W \times h(n)$, dla $W > 1$

Różne warianty W, dla podsumowania

- A^*: $g(n) + h(n)$, czyli $W = 1$

Rezygnacja z optymizmu

Weighted A^* search

$f(n) = g(n) + W \times h(n)$, dla $W > 1$

Różne warianty W, dla podsumowania

- A^*: $g(n) + h(n)$, czyli $W = 1$
- **Uniform-cost search**: $g(n)$, czyli $W = 0$

Rezygnacja z optymizmu

Weighted A^* search

$f(n) = g(n) + W \times h(n)$, dla $W > 1$

Różne warianty W, dla podsumowania

- A^*: $g(n) + h(n)$, czyli $W = 1$
- **Uniform-cost search**: $g(n)$, czyli $W = 0$
- **Greedy best-first search**: $h(n)$, czyli $W = \infty$

Rezygnacja z optymizmu

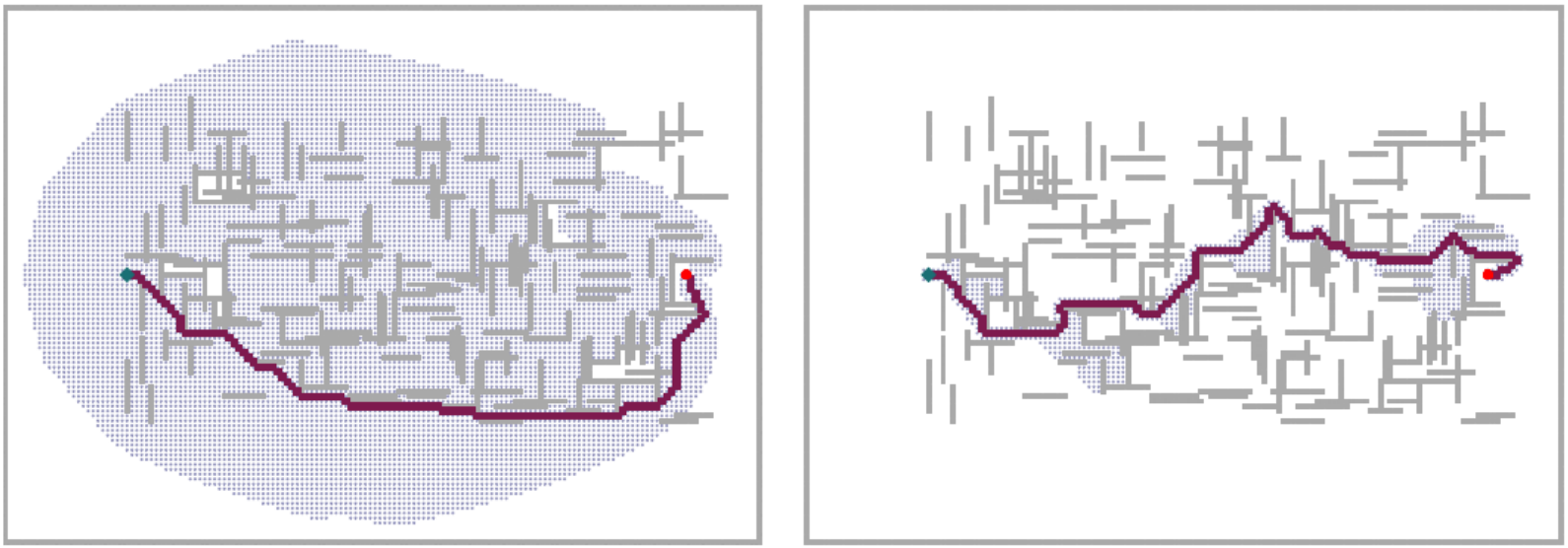
Weighted A^* search

$f(n) = g(n) + W \times h(n)$, dla $W > 1$

Różne warianty W, dla podsumowania

- A^*: $g(n) + h(n)$, czyli $W = 1$
- **Uniform-cost search**: $g(n)$, czyli $W = 0$
- **Greedy best-first search**: $h(n)$, czyli $W = \infty$
- **Weighted A^* search**

Przykładowe wyniki Weighted A*



(a) (b)

Figure 3.21 Two searches on the same grid: (a) an A* search and (b) a weighted A* search with weight $W = 2$. The gray bars are obstacles, the purple line is the path from the green start to red goal, and the small dots are states that were reached by each search. On this particular problem, weighted A* explores 7 times fewer states and finds a path that is 5% more costly.

Ograniczenia pamięci

Możemy lepiej kontrolować używanie pamięci

Ograniczenia pamięci

Możemy lepiej kontrolować używanie pamięci

Jeden z najprostszych wariantów: **Iterative Deepening A***
rozwijamy tylko wartości f mniejsze od pewnego progu

Ograniczenia pamięci

Możemy lepiej kontrolować używanie pamięci

Jeden z najprostszych wariantów: **Iterative Deepening A***
rozwijamy tylko wartości f mniejsze od pewnego progu
W kolejnej iteracji mniejsze od najmniejszej odwiedzonej, nierozwiniętej.

Ograniczenia pamięci

Możemy lepiej kontrolować używanie pamięci

Jeden z najprostszych wariantów: **Iterative Deepening A***
rozwijamy tylko wartości f mniejsze od pewnego progu
W kolejnej iteracji mniejsze od najmniejszej odwiedzonej, nierozwiniętej.

Wiele innych wariantów: Beam search, zostawianie tylko niektórych węzłów (niezbyt odległych od lidera F), RBFS, MA*, SMA*, ...

Wyszukiwanie dróg (Google Maps, etc)

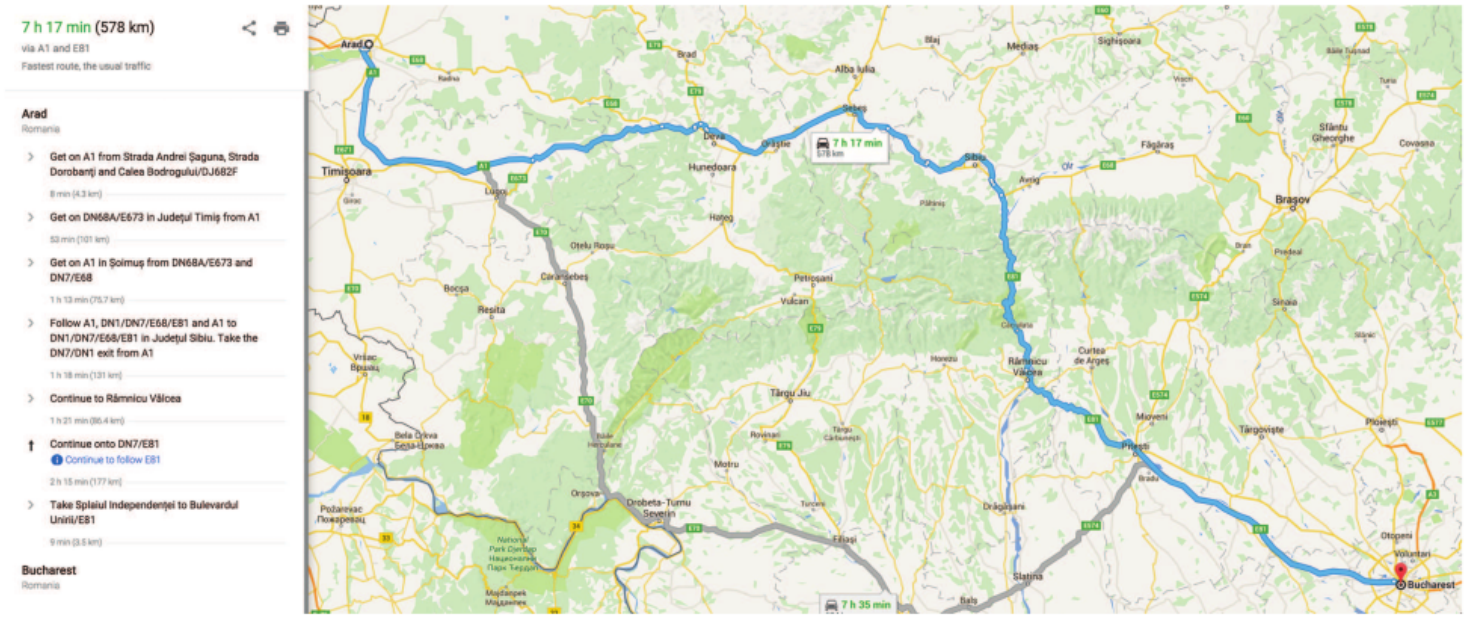


Figure 3.28 A Web service providing driving directions, computed by a search algorithm.

Charakterystyka:

- Duży graf, dany explicite
- Wynik potrzebny od zaraz
- Możliwy preprocessing (bo zapytania mogą być podobne do siebie)

Punkty orientacyjne

Landmarks

Zakładamy istnienie pewnej liczby **punktów orientacyjnych**, będziemy się do nich odwoływać przy liczeniu heurystycznej odległości między węzłami

Punkty orientacyjne

Landmarks

Zakładamy istnienie pewnej liczby **punktów orientacyjnych**, będziemy się do nich odwoływać przy liczeniu heurystycznej odległości między węzłami

Zakładamy, że policzyliśmy odległości między **każdym** **p.o.** a **każdym** węzłem.

Heurystyka

$h(n) = \min_{L \in \text{Landmarks}}(C^*(n, L) + C^*(L, \text{goal}))$

Punkty orientacyjne

Landmarks

Zakładamy istnienie pewnej liczby **punktów orientacyjnych**, będziemy się do nich odwoływać przy liczeniu heurystycznej odległości między węzłami

Zakładamy, że policzyliśmy odległości między **każdym** **p.o.** a **każdym** węzłem.

Heurystyka

$h(n) = \min_{L \in \text{Landmarks}}(C^*(n, L) + C^*(L, \text{goal}))$

Ogólnie nie jest optymistyczna, chyba że ścieżka przechodzi przez *jakiś* punkt orientacyjny!

Differential heuristic

Differential heuristic

$h(n) = \max_{L \in \text{Landmarks}}(C^*(n, L) - C^*(L, \text{goal}))$

Differential heuristic

Differential heuristic

$h(n) = \max_{L \in \text{Landmarks}}(C^*(n, L) - C^*(L, \text{goal}))$
(jak wyjdzie ujemna, to dajemy 0)

Differential heuristic

Differential heuristic

$h(n) = \max_{L \in \text{Landmarks}}(C^*(n, L) - C^*(L, \text{goal}))$
(jak wyjdzie ujemna, to dajemy 0)

- Myślimy, że punkt orientacyjny jest **za celem** (jadąc do L mijamy cel po drodze)
- Jak cel jest trochę z boku drogi, to tracimy dokładność, ale nie optymizm.

Wybór punktów orientacyjnych

- Można po prostu wylosować.

Wybór punktów orientacyjnych

- Można po prostu wylosować.
- Dowolna zachłanno/losowa procedura, starająca się by punkty były odległe (euklidesowo)

Wybór punktów orientacyjnych

- Można po prostu wylosować.
- Dowolna zachłanno/losowa procedura, starająca się by punkty były odległe (euklidesowo)
- Wybrać centrum i losować daleko od niego

Wybór punktów orientacyjnych

- Można po prostu wylosować.
- Dowolna zachłanno/losowa procedura, starająca się by punkty były odległe (euklidesowo)
- Wybrać centrum i losować daleko od niego
- Uwzględniać „dane ze świata” (czyli na przykład premiować duże węzły komunikacyjne)

Uczenie się heurystyk

- Pomyślmy o ósemce i o tym, że będziemy liczyć wartość heurystyki jako $h_c(n) = \alpha_1 h_1(n) + \alpha_2 h_2(n) + \alpha_3 h_3(n)$

Uczenie się heurystyk

- Pomyślmy o ósemce i o tym, że będziemy liczyć wartość heurystyki jako $h_c(n) = \alpha_1 h_1(n) + \alpha_2 h_2(n) + \alpha_3 h_3(n)$
- Jak wyznaczyć wartości α?
- Możemy wygenerować pewną liczbę par $(n, h_{\text{opt}(n)})$ i **nauczyć się** wartości α w ten sposób, by h_c przybliżało h_{opt}