



Ćwiczenie 1

Planowanie trasy robota mobilnego w siatce kwadratów pól - Algorytm A^*

0.022	0.198	 0.509	
0.008	0.068	0.175	
0.001	0.006	0.016	

Zadanie do wykonania

- 1) Tworzymy na pulpicie katalog w formacie Imię_nazwisko, w którym umieszczamy wszystkie pliki związane z ćwiczeniem.
- 2) Czytamy teorię związaną z algorytmem A^* , w razie problemów ze zrozumieniem, analizujemy przykład na kartce.
- 3) Generujemy mapę z przeszkodami za pomocą programu map_generator.exe.
- 4) Implementujemy w C++ kroki algorytmu A^* na wygenerowanej mapie. Wartości 0 oznaczają dostępne na mapie miejsca. Wartości 5 oznaczają przeszkody. Wartością 3 zaznacz wyliczoną trasę. Przyjmujemy, że Start ma współrzędne (0,0) (pierwszy wiersz od dołu, pierwsza kolumna), Cel ma współrzędne (19,19) (dwudziesty wiersz od dołu, dwudziesta kolumna). Jako heurystykę przyjmujemy odległość Euklidesową. Koszt pojedynczego ruchu wynosi 1. Możliwe ruchy to góra, dół, lewa, prawa (ruch na skos jest zabroniony). Kolejność przeglądania pól to góra, dół, lewa, prawa. Konflikty rozwiązywane są hierarchicznie lub losowo. W wariantcie hierarchicznym wybieramy najwcześniej lub najpóźniej napotkaną wartość spośród wartości konfliktujących.
- 5) Jeżeli nie programujemy biegle w C++, zapoznajemy się z programem demonstracyjnym C++, na stronie <http://wmii.uwm.edu.pl/~artem> w zakładce Dydaktyka/Elementy Robotyki i Automatyki. Program umieszczamy w swoim katalogu na pulpicie.

Algorytm A^* - część teoretyczna

Algorytm A^* jest jednym z najefektywniejszych algorytmów wyszukiwania optymalnej drogi od ustalonego punktu startowego do docelowego. Pierwsze wzmianki o algorytmie A^* jako rozszerzeniu algorytmu Dijkstry, nastąpiły w 1968 roku.

Omówmy działanie algorytmu w siatce kwadratów pól (którą będziemy określali mianem gridu). Algorytm A^* interesuje nas głównie pod kątem szerokiego zastosowania w planowaniu ruchu robotów mobilnych.

Metoda A^* używa ideologii pierwszy-najlepszy, szuka ścieżki do celu o najmniejszym koszcie w sensie wartości funkcji f definiowanej w rozważanych do odwiedzenia pozycjach poz następująco,

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

gdzie poz jest rozważaną pozycją, $g(poz)$ kosztem dojścia od pozycji startowej do poz , natomiast $h(poz)$ jest heurystyką szacującą odległość do celu z pozycji poz .

Heurystyka $h(poz)$ nie jest określana jednoznacznie, może przyjmować różne warianty, koniecznym założeniem jest jej dopuszczalność, czyli nie może przeszacowywać odległości do celu. Heurystyka dopuszczalna powinna spełniać następujący warunek: dla każdej możliwej do odwiedzenia pozycji poz ,

$$h(poz) \leq \text{koszt optymalnej drogi do celu z pozycji } poz,$$

Gdy ten warunek nie jest spełniony, istnieje możliwość, że w pewnych konfiguracjach pozycji startowej, pozycji docelowej i pewnych ustawieniach przeszkód, algorytm A^* wybierze nieoptymalną drogę do celu.

Przejdźmy do opisu podstawowych założeń i kroków, które składają się na algorytm A^* , opis dotyczy działania w siatce kwadratów pól,

Krok 1) ustalamy sposób poruszania się inteligentnego agenta, koszt jego poszczególnych ruchów oraz kolejność przeszukiwania pól wokół aktualnej pozycji,

Krok 2) dobieramy heurystykę h , spełniającą warunek dopuszczalności,

Krok 3) uzgadniamy sposoby rozwiązywania konfliktów, powstających podczas eksploracji siatki,

Krok 4) wskazujemy punkt startowy i docelowy,

Krok 5) tworzymy dwie pomocnicze listy, listę otwartą inicjowaną zborem pustym, która będzie zawierała kratki aktualnie rozważane jako pola do ekspansji oraz listę zamkniętą inicjowaną polem startowym, do której trafiają pola odwiedzone, znikające jednocześnie z listy otwartej,

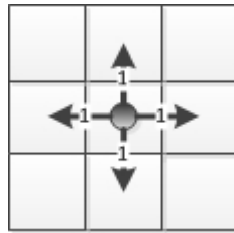
Krok 6) kratki otaczające pole ostatnio dodane do listy zamkniętej, które możemy odwiedzić trafiają na listę otwartą i zachowują informację o kratce rodzicu, za pośrednictwem, której trafiły na listę otwartą; wyliczana jest dla nich wartość funkcji f , w przypadku gdy już posiadają tę wartość obliczoną za pośrednictwem innego pola dodanego wcześniej do listy zamkniętej, porównujemy wyliczaną wartość z aktualną i zmieniamy rodzica w przypadku gdy oszacowany koszt f dotarcia do celu jest mniejszy od poprzedniego,

Krok 7) do listy zamkniętej trafia kratka z listy otwartej, której wartość f jest najmniejsza (konflikty rozwiązywane są hierarchicznie, wybieramy spośród kratek o tym samym oszacowaniu f , pole odwiedzone najpóźniej),

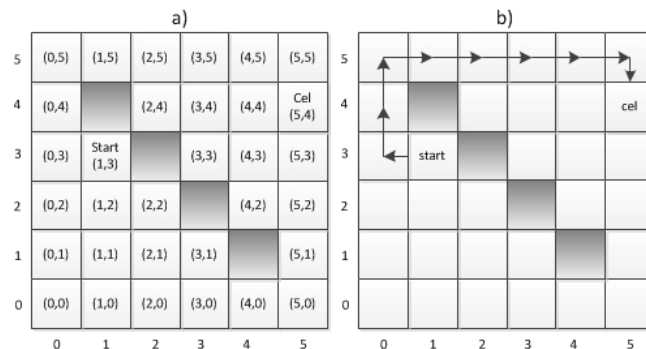
Krok 8) jeżeli cel nie został osiągnięty i lista otwarta zawiera przynajmniej jeden element wykonujemy skok do Kroku 6),

Krok 9) algorytm może zakończyć się dwoma scenariuszami: a) jeżeli cel został osiągnięty, wracamy wstecz po kolejnych kratkach rodzicach do kratki start, wyznaczając w ten sposób optymalną ścieżkę od startu do celu, b) gdy lista otwarta nie zawiera żadnego elementu a cel nie został osiągnięty, zwracany jest komunikat o niemożliwości dotarcia do celu.

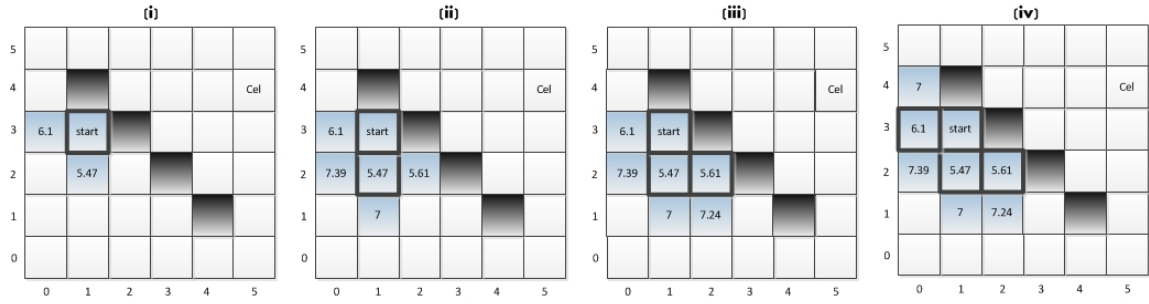
Przykład działania algorytmu A* dla ustalonej heurystyki dopuszczalnej, jako odległości Euklidesa możemy zobaczyć na Rys. 2. Przedstawiamy w punkcie a) Rys. 2 siatkę ze współrzędnymi krutek, a w punkcie b) Rys. 2 drogę jaka jest determinowana przez A* od kratki start do kratki cel. Dla uproszczenia zakładamy, że robot może poruszać się po pustych kratkach w kierunkach, dół, lewa, góra, prawa, w tej samej kolejności przeszukuje kratki wokół odwiedzanego pola oraz nie może poruszać się po przekątnych kratkach - patrz. Rys. 1. Kolejne kroki działania algorytmu A* w siatce, możemy prześledzić na Rys. 3, 4, 5, 6 oraz 7.



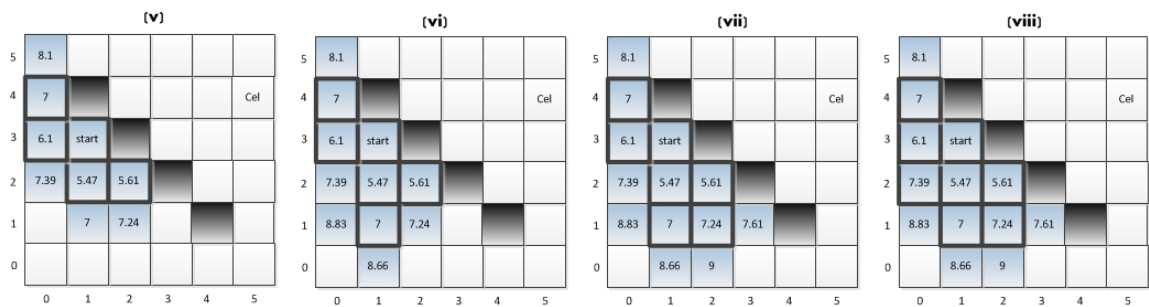
Rysunek 1: Wybrana strategia poruszania się inteligentnego agenta po siatce z ustalonym kosztem pojedynczego ruchu, przy założeniu, że ruch po skosie jest wykluczony. Kolejność przeglądania krutek wokół aktualnej pozycji ustalamy na: dół, lewa, góra, prawa



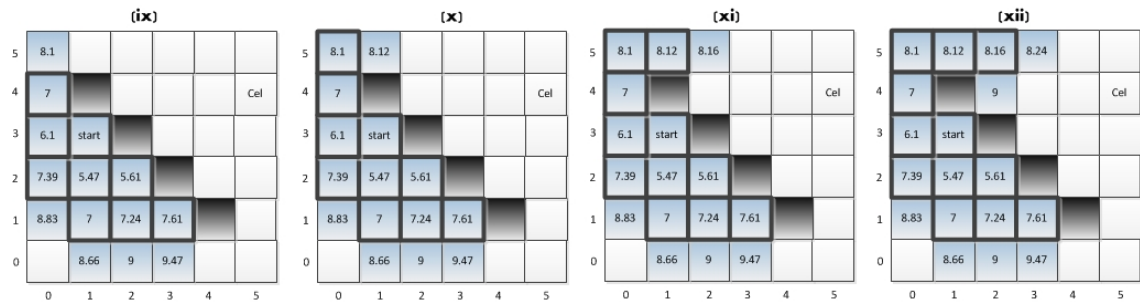
Rysunek 2: Działanie A* na podstawie heurystyki dopuszczalnej jako metryki Euklidesa $h((poz_x, poz_y)) = \sqrt{(poz_x - cel_x)^2 + (poz_y - cel_y)^2}$, (poz_x, poz_y) to współrzędne kratki dla której wyliczamy heurystykę, (cel_x, cel_y) są współrzędnymi celu. W podpunkcie a) mamy siatkę ze współrzędnymi krutek, w podpunkcie b) optymalną drogę wyznaczoną algorytmem A*. Na Rys. 3, 4, 5, 6 oraz 7, przedstawiamy działanie algorytmu w szczegółach



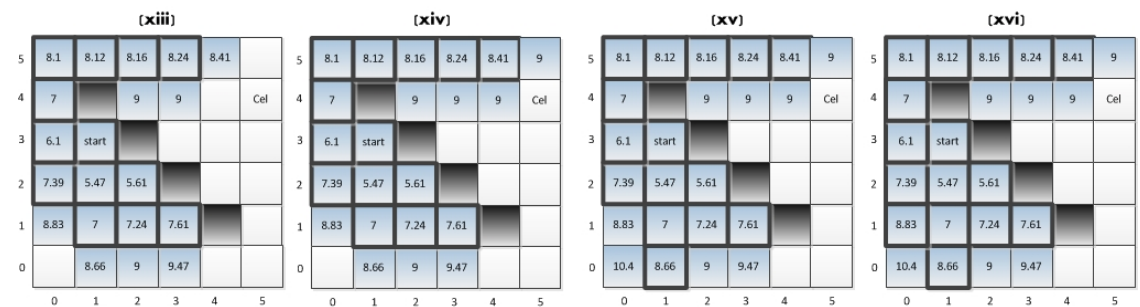
Rysunek 3: Działanie algorytmu A* na podstawie heurystyki jako metryki Euklidesa, strategii poruszania się i kosztu ruchu z Rys. 1. Opiszmy pierwsze kroki działania algorytmu A*. w punkcie (i) kratka start trafia do listy zamkniętej, kratki możliwe do odwiedzenia w kolejności dół, lewa trafiają do listy otwartej z informacją, że ich rodzicem jest kratka start i wyliczamy dla nich wartość funkcji f następująco, $f((1,2)) = 1 + \sqrt{(1-5)^2 + (2-4)^2} = 1 + \sqrt{20} \approx 5.47$, $f((0,3)) \approx 6.1$, w kolejnych krokach pamiętamy o przypisywaniu kratkom pojawiającym się na liście otwartej rodziców za pośrednictwem których znalazły się na liście, z uaktualnieniami rodzica w przypadku gdy jest możliwość dotarcia do tych kraterk w bardziej optymalny sposób, w kroku (ii) wybieramy kratkę z listy otwartej, która ma najmniejszy koszt f, czyli (1,2), umieszczamy ją na liście zamkniętej, a na liście otwartej kratki przyległe. Wyliczamy $f((1,1)) = 7$, $f((0,2)) \approx 7.39$, $f((2,2)) \approx 5.61$, w podpunkcie (iii) do listy zamkniętej trafia kratka o współrzędnych (2,2), do listy otwartej kratka (2,1), obliczamy $f((2,1)) \approx 7.24$, w podpunkcie (iv) do listy zamkniętej trafia kratka (0,3), do listy otwartej (0,4), $f((0,4)) = 7$, kratka (0,2) jest już na liście otwartej, wyliczona wartość f nie zmienia się przy obliczaniu z perspektywy dojścia z (0,3) stąd rodzic kratki (0,2) pozostaje bez zmian.



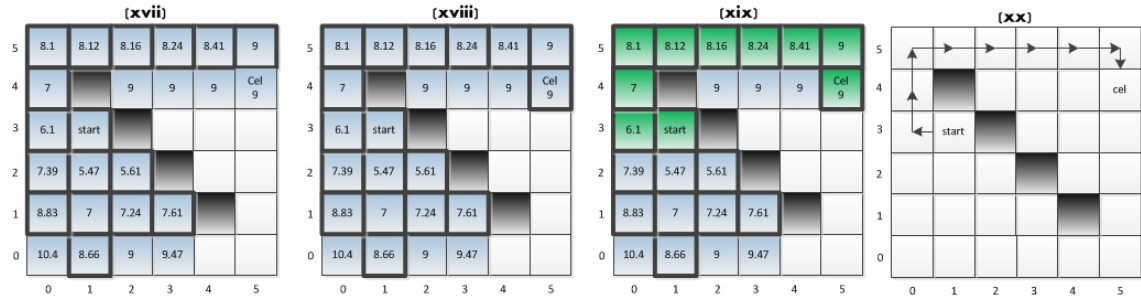
Rysunek 4: Działanie algorytmu A* cd.; metryka Euklidesa; w podpunkcie (v) do listy zamkniętej mamy dwie kandydujące kratki z listy otwartej, kratkę (1,1) i kratkę (0,4), korzystając ze strategii wyboru, kratki która została odkryta najpóźniej, do listy zamkniętej dodajemy kratkę (0,4), a do otwartej (0,5), $f((0,5)) \approx 8.1$, w punkcie (vi) do listy zamkniętej trafia kratka (1,1), do otwartej (1,0), $f((1,0)) \approx 8.66$, oraz (0,1), $f((0,1)) \approx 7.39$, koszt (2,1) nie zmienia się stąd jej rodzicem pozostaje kratka (2,2), w punkcie (vii) do listy zamkniętej dodajemy kratkę (2,1), do listy otwartej (2,0), $f((2,1)) = 9$ oraz (3,1), $f((3,1)) \approx 7.61$, w kroku (viii) do listy zamkniętej trafia kratka (0,2), stan listy otwartej nie zmienia się, kratka (0,1) nie zmienia rodzica



Rysunek 5: Działanie algorytmu A* cd.; metryka Euklidesa; w podpunkcie (ix) do listy zamkniętej dokładamy kratkę (3,1), do listy otwartej (3,0), $f((3,0)) \approx 9.47$, w kroku (x) do listy zamkniętej trafia kratka (0,5), do otwartej (1,5), $f((1,5)) \approx 8.12$, w kroku (xi) do listy zamkniętej trafia kratka (1,5), do otwartej (2,5), $f((2,5)) \approx 8.16$, w punkcie (xii) do listy zamkniętej trafia kratka (2,5), do listy otwartej kratka (2,4), $f((2,4)) = 9$ oraz (3,5), $f((3,5)) \approx 8.24$



Rysunek 6: Działanie algorytmu A* cd.; metryka Euklidesa; w podpunkcie (xiii) na liście zamkniętej pojawia się kratka (3,5), na liście otwartej kratka (3,4), $f((0,1)) = 9$ oraz (4,5), $f((4,5)) \approx 8.41$, w kroku (xiv) do listy zamkniętej dopisujemy (4,5), do otwartej (4,4), $f((4,4)) = 9$ oraz (5,5), $f((5,5)) = 9$, w kroku (xv) dodajemy do listy zamkniętej kratkę (1,0), do otwartej $f((0,0)) \approx 10.4$, kratka (2,0) nie zmienia rodzica, w punkcie (xvi) dodajemy do listy zamkniętej kratkę (0,1), rodzic kratki (0,0) pozostaje bez zmian



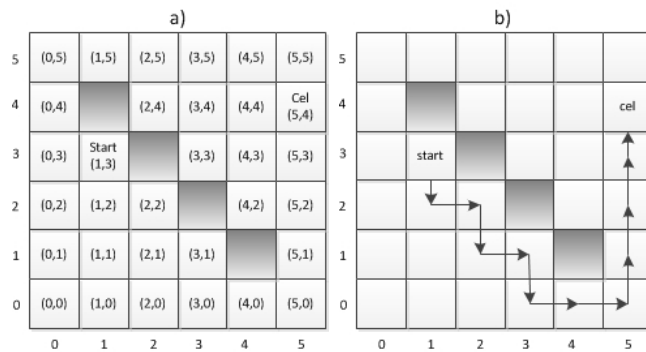
Rysunek 7: Działanie algorytmu A* cd.; metryka Euklidesa; w podpunkcie (xvii) mamy na liście otwartej pięć kratek z minimalnym kosztem 9, do listy zamkniętej wybieramy kratkę, która trafiła do listy otwartej jako ostatnia czyli pole (5,5), do listy otwartej trafia kratka cel o współrzędnych (5,4) z $f((5,4)) = 9$, w punkcie (xviii) kratka cel trafia do listy zamkniętej i w kroku (xix) algorytm przechodzi wstecz od celu do kolejnych kraterów rodziców, aż do osiągnięcia kratki start, zapamiętując tę sekwencję, w kroku (xx) mamy pokazane kolejne ruchy, które wyznaczają optymalną drogę od startu do celu. Algorytm kończy działanie

Dla zobrazowania problemu, który może się pojawić, gdy nasza heurystyka jest nie-dopuszczalna, przedstawiamy na Rys. 8 w podpunkcie b), sekwencję ruchów do celu na podstawie metryki rzeka definiowanej następująco,

$$h(poz) = poz_y + |cel_x - poz_x| + cel_y$$

gdzie (poz_x, poz_y) , są współrzędnymi aktualnej pozycji, a (cel_x, cel_y) są współrzędnymi celu.

Jak widzimy, przy zastosowaniu powyższej heurystyki cel został osiągnięty w sposób nieoptymalny, obliczona ścieżka jest o dwa kroki dłuższa od optymalnej.



Rysunek 8: Działanie algorytmu A* na bazie heurystyki określonej na podstawie metryki rzeka, ustalonej jako kratki o drugiej współrzędnej zerowej