

SPRAWOZDANIE 1

Podstawy Sztucznej Inteligencji

Przeszukiwanie. Heurystyki

Natalia Gadocha 304165
Geoinformatyka III rok

1. Obserwacje analizowanych obszarów dla poszczególnych algorytmów

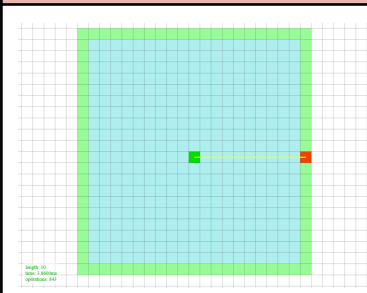
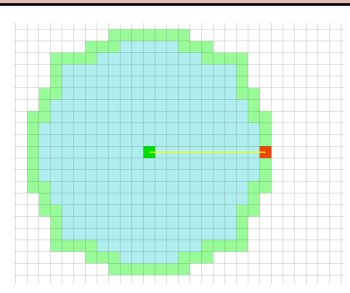
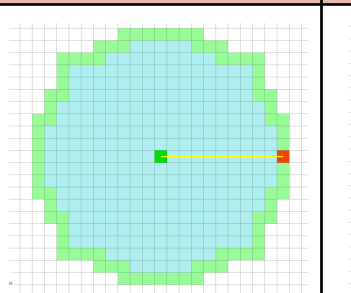
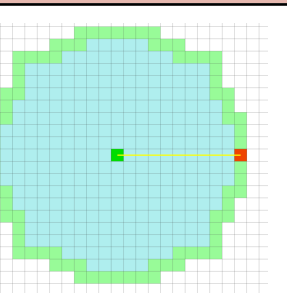
Każda badana długość (length) wynosiła 10 jednostek. Analizowany obszar jest optymalny dla domyślnych ustawień. Jest on jednak duży ze względu na jednokierunkowe rozbudowanie kwadracików (tylko z jednego miejsca jest szukana droga do drugiego celu). Dlatego też obszar zajmuje dużą część dostępnego miejsca.

Obszar zależy też w dużej mierze od wybranego algorytmu i używanych przez niego kształtów tworzonych wybranych dróg. Np algorytm Breadth-First-Search domyślnie analizuje obszar jako kwadrat, natomiast Dijkstra przybiera kształt zbliżony do koła, tak jak pozostałe badane przez nas dwa algorytmy (Best-First-Search oraz A*).

2. Dla domyślnych ustawień statystyki przedstawiają się następująco:

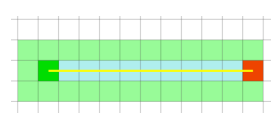
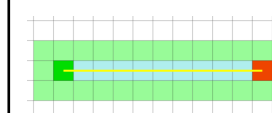
Algorytm	Długość ścieżki	Czas [ms]	Liczba operacji
Breadth-First-Search	10	6.4150	843
Dijkstra	10	6.1450	643
Best-First-Search	10	88.8500	646
A*	10	122.8650	468

Wizualnie, badane obszary wyglądają następująco:

Breadth-First-Search	Dijkstra	Best-First-Search	A*
			

3. Po dodaniu heurystyki dx

Algorytm	Długość ścieżki	Czas [ms]	Liczba operacji
Breadth-First-Search	X	X	X
Dijkstra	X	X	X
Best-First-Search	10	6.2900	47
A*	10	5.5750	47

Breadth-First-Search	Dijkstra	Best-First-Search	A*
X	X		

Na podstawie powyższych danych możemy jasno zauważyć, iż dodanie heurystyki dx znacząco poprawiło działanie algorytmów. W przypadku Best-First-Search działanie poprawiło się prawie piętnastokrotnie, natomiast dla A* poprawa była dwudziestodwukrotna. Długość wyznaczonej trasy jest taka sama, lecz zmniejszyła się powierzchnia badanego obszaru, co zaoszczędziło czas przeszukiwania oraz liczbę operacji.

4. Najpopularniejsze heurystyki dla wyszukiwania na płaszczyźnie:

- hill climbing
- best-first search
- uniform cost search
- A* oraz A* z iteracyjnym poprawianiem,
- simulated annealing

5. Heurystyki opisane w artykule:

a. Manhattan distance

Metoda ta używa czterech kierunków po poszukiwań najlepszej drogi. Możemy ją jednak w różny sposób optymalizować, tak aby algorytm mógł działać szybciej i bardziej efektywnie. Można to osiągnąć np. poprzez zwiększanie wartości D lub też zmniejszając stosunek najniższych i najwyższych kosztów brzegowych. Sama wartość D odpowiada za zużywanie mniejszych kosztów pomiędzy przyległymi kwadratami. Odległość dwóch punktów w tej metryce to suma wartości bezwzględnych różnic ich współrzędnych.

b. Diagonal distance

W tej metodzie natomiast jest używane osiem kierunków. Zakłada ona bowiem dodatkowy ruch po przekątnych, które mają dodatkowo optymalizować drogę oraz czas. Używa się w niej również specjalnych przekształceń, jeżeli nie jesteśmy w bezpośredni sposób utworzyć przekątnej. Co więcej, możemy w niej wyróżnić przypadki, które zawierają inne odległości; a mianowicie: Chebyshev oraz Octile distance.

c. Euclidean distance

Metoda ta mówi o tym, iż bardziej optymalną opcją jest wybranie drogi, która jest linią prostą. Jeżeli jest to oczywiście możliwe. Algorytm ten jednak nie jest odpowiedni dla metody A*, gdyż linia prosta niekorzystnie wpływa na wartość h.

d. Euclidean distance, squared

Metoda ta też jest bardziej problematyczna dla algorytmu A*. Bardzo łatwo otrzymać przeszacowaną heurystykę, za duży koszt lub nieoptymalne rozwiązania, zwłaszcza jeżeli chodzi o przypadek, kiedy to drogi są dłuższe. Zmniejszając heurystykę możemy osiągnąć odwrotny

problem: dla krótszych dróg metoda ta nie będzie efektywna i nie osiągniemy wymaganego rezultatu. Możemy jednak próbować niwelować nadmiarowe koszty poprzez przybliżenia pierwiastka metodą Euclidean distance albo użyć do tego odległości po przekątnej.

e. Multiple goals

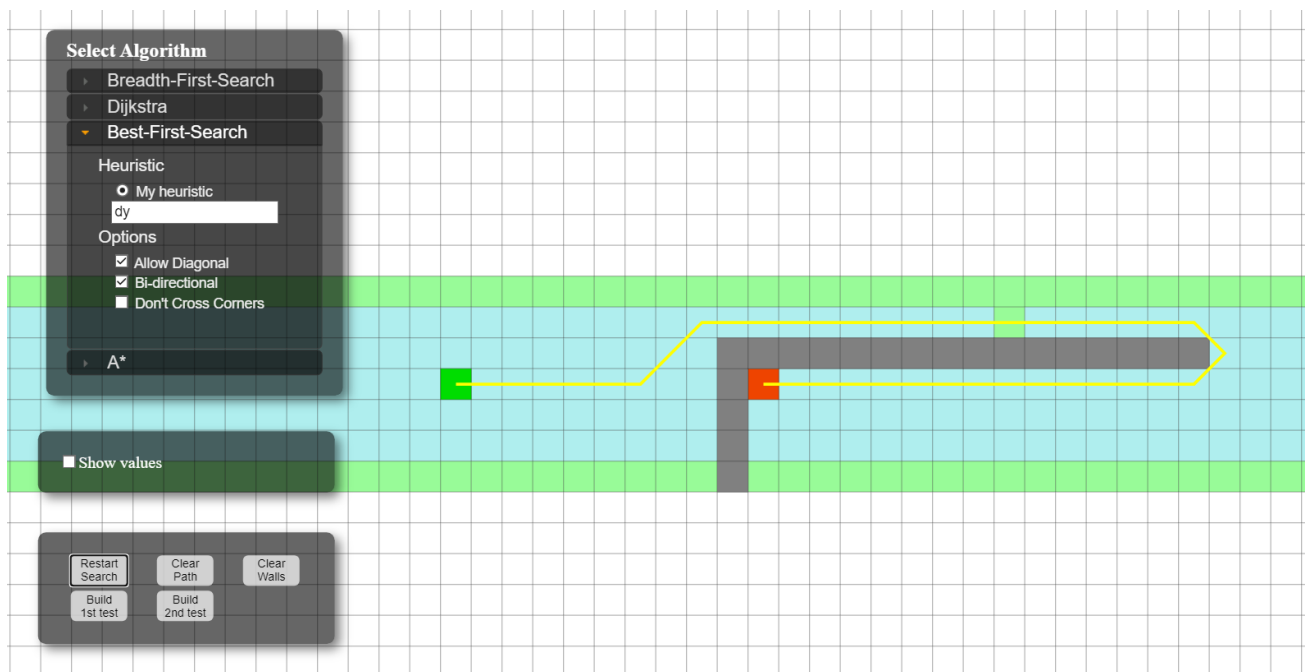
Metoda ta opiera się głównie na konstrukcji heurystyki w celu zwiększenia efektywności i optymalizacji działań. Możemy bowiem bez wkładu dodatkowych kosztów dodać krawędź ze skrajnych punktów. Algorytmy, które są mocno pomocne i używane to: A* (która jest bardziej efektywna w szukaniu przy jednym z krańców) oraz Dijkstra (która jest bardziej efektywna w szukaniu optymalnych powiązań) - w zależności od tego, dla jakiej ilości celów będziemy szukać dróg oraz na czym te powiązania powinny się skupiać.

f. Breaking ties

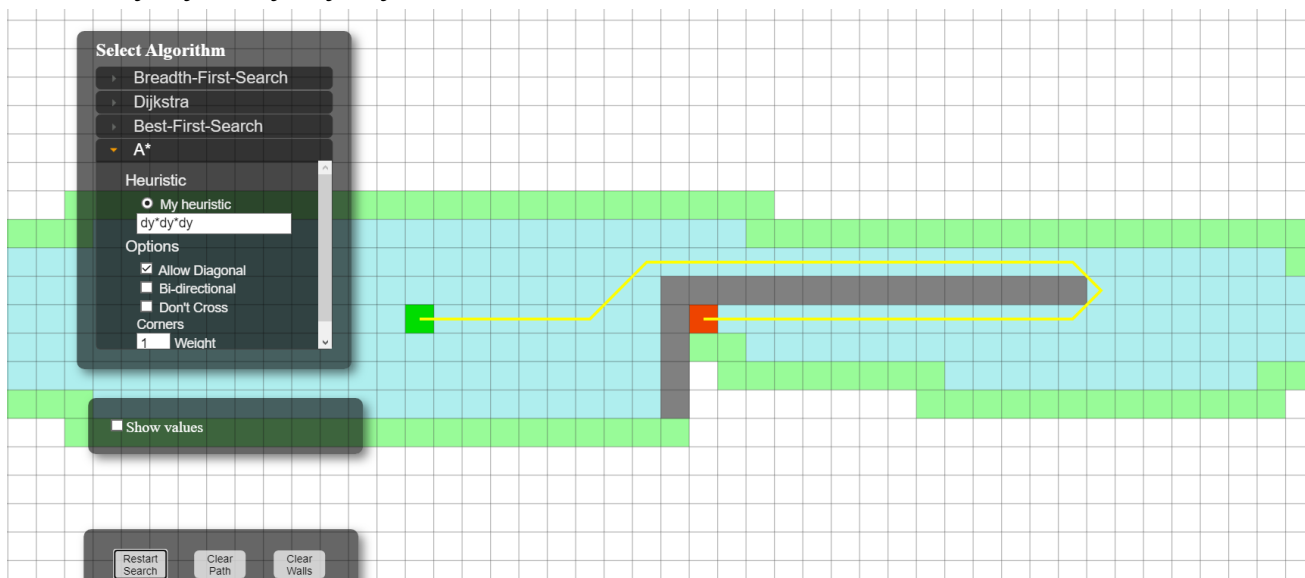
Problem, jaki może pojawić się przy wybieraniu najlepszej drogi to różne drogi o różnej optymalizacji, lecz o równej długości. Główną metodą, aby jak najbardziej efektywnie prowadzić drogi jest zmiana skali wartości h . Pozwala to na zaoszczędzenie czasu oraz ilości wykonywanych obliczeń. Innymi sposobami są m.in: porównanie wartości h z funkcji f , dodanie deterministycznej liczby losowej do heurystyki oraz obliczanie iloczynów wektorowych pomiędzy wektorem początkowym i końcowym a bieżącego punktu i końcowego (powstaje nam dzięki temu droga, która "preferuje" wybór linii prostych). Kolejny pomysł jaki się pojawił to stworzenie listy priorytetów, tak aby nowe wstawione wartości były położone w niej wyżej niż wartości dla starszych. Podobna pojawiająca się metoda to chęć zminimalizowania zakrętów. Zakłada ona względne poruszanie się po współrzędnych x, y w stosunku do początkowej wartości z początku. Ostatnim podejściem jest podejście "szachownicy". Chodzi w nim o to, by tworzone ścieżki były tworzone naprzemiennie wzdłuż pionowej i poziomej powierzchni (tworząc coś w rodzaju wcześniej wspomnianej szachownicy).

6. Poprowadzenie drogi w określony sposób

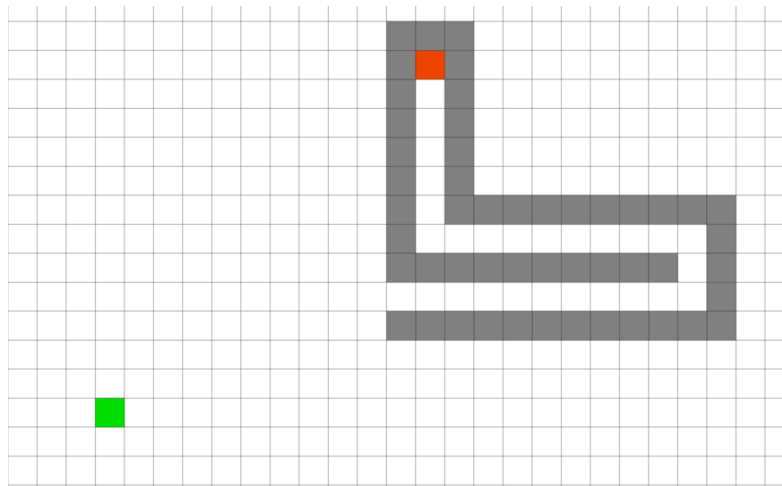
a. Best-First-Search, heurystyka: dy



b. A*, heurystyka $dy*dy*dy$



7. Próba znalezienia najszybszej drogi



W celu znalezienia najlepszej heurystyki dla podanego przykładu została stworzona tabelka wraz z niektórymi wynikami. Prezentuje się ona następująco:

Algorytm	Heurystyka	Czas [ms]	Odległość	Operacje
Best-First-Search	dx	128.1650	38.07	1311
	dy	307.9900	38.66	1472
	dy*dy*dy	227.2470	38.66	1472
	dx*dx*dx	174.0100	38.07	1311
	dx-dy	327.640	39.38	2395
	dy/dx	447.9450	36.9	4508
	dx*dx+dy*dy	62.7600	39.38	556
	100*(dx*dx+dy*dy)	46.3050	38.97	559
	(dx*dx+dy*dy)/1000	42.4450	38.97	509
	0	420.1550	36.9	3442
A*	dx*dx+dy*dy	66.9250	39.38	563
	0	231.6100	36.31	3408
	dx+dy	123.0750	36.7	1129

Natomiast tabelka z wartościami z wykorzystaną opcją bi-directional wygląda następująco:

Algorytm	Heurystyka	Czas	Odległość	Operacje
Breadth-First-Search	x	1.7000	36.9	287
Dijkstra	x	1.0050	36.9	285
Best-First-Search	$dx \cdot dx + dy \cdot dy$	49.5600	38.56	210
	dy	43.1950	38.07	289
	0	49.5990	36.9	285
	dx/dy	31.8200	51.38	349
A*	0	64.6100	37.31	308
	$dx \cdot dx + dy \cdot dy$	53.1250	38.56	204
	dx^2	24.5900	36.9	171

Z powyższych zależności wynika:

- Bardzo trudno jest uzyskać algorytm, który będzie jednocześnie optymalny czasowo, odległościowo oraz obliczeniowo. Często możemy dostać efekt np. niewielkiej odległości, lecz o długim czasie wyszukiwania.
- Całokształtowo, lepsze wyniki otrzymujemy podczas przeszukiwania dwukierunkowego. Znacząco jest skracany czas oraz ilość wykonanych operacji.
- Co można było dostrzec podczas prób heurystyk to to, iż większość algorytmów i wypróbowanych metod (w wyszukiwaniu jednokierunkowym) porusza się bardzo wolno wewnątrz labiryntu. Spowodowane jest to tym, iż dany algorytm przeszukuje wszystkie dostępne sąsiadujące kwadraty (wcześniej zostały opisane kierunki poruszania się po badanych obszarach). A trafiając na ścianę nie przerywa obliczeń, lecz sprawdza dostępne możliwości. Równocześnie jego uwaga jest też

skupiona na większym, nieograniczonym terenie, gdzie zdecydowanie szybciej i łatwiej może prowadzić niezbędne procedury.

- Porównując utworzone heurystyki dla wyszukiwania jedno- oraz dwukierunkowego:
 - stworzona heurystyka dla algorytmu First-Best-Search $[(dx*dx+dy*dy)/1000]$ jest o mniej więcej 4 ms szybsza od domyślnej heurystyki dla wyszukiwania dwukierunkowego (wartości kolejno: 42.4450 oraz 49.5990). Nie jest jednak bardziej optymalna pod względem ilości wykonywanych obliczeń oraz odległości.
 - Najkrótszą drogę otrzymałam dla heurystyki $[dx/dy]$ i jest ona równa 36.2. Niestety reszta parametrów ma całkowicie odwrotne wyniki. Do uzyskania tego rozwiązania przeprowadzono największą ilość operacji (ponad 4000) oraz była potrzeba do tego największa ilość czasu (ponad 400 ms). Odwrotnie jednak prezentują się statystyki dla przeszukiwania dwukierunkowego: otrzymujemy najdłuższą drogę lecz w najkrótszym czasie i najmniejszej liczbie obliczeń.
- Próby otrzymania najszybszego algorytmu z czasem mniejszym niż 50 ms oraz liczbą operacji mniejszą niż 200 w wyszukiwaniu jednokierunkowym:
 - Niestety nie udało mi się otrzymać jednego rozwiązania, które spełniałoby oba te warunki.
 - Nie udało mi się również otrzymać heurystyki, która nie przekraczałaby dwustu operacji. Żaden z wyników nie był bliski tej liczbie. Wartości o niższej liczbie operacji wynosiły około 550. Na pewno zabrakło mi doświadczenia oraz wyczucia, aby rozwiązać powyższy problem.
 - Heurystyką, która spełnia warunek czasowy jest $(dx*dx+dy*dy)/1000$ dla Best-First-Search. Czas wyznaczania ścieżki to ok 42 ms. Jednakże liczba potrzebnych do tego operacji to aż 509.

Uzyskana ścieżka wygląda następująco:

