Politechnika Wrocławska

Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka (AIR)

PRACA DYPLOMOWA MAGISTERSKA

TYTUŁ PRACY:

Porównanie algorytmów planowania ścieżki opartych na Q–learningu

AUTOR:

Przemysław Jaskuła

PROMOTOR:

dr inż. Wojciech Domski

WROCŁAW 2023

Streszczenie

Celem pracy było porównanie algorytmów inicjalizacji macierzy Q w algorytmie Q–learningu dla zadania planowania ścieżki. Na początku pracy przedstawiono wybrane algorytmy wykorzystywane w celu planowania ścieżki, w tym również algorytm Q–learningu. Następnie zostały dogłębnie omówione wykorzystane algorytmy służące do inicjalizacji macierzy Q. Omówione zostały też podjęte próby zaimplementowania sieci neuronowych inicjalizujących macierz Q. Zarówno algorytm Q–learningu, jak i poszczególne algorytmy inicjalizacji zostały zaimplementowane w języku Python. Na końcu pracy zostały porównane zaimplementowane algorytmy na podstawie czasu wykonywania algorytmu, czasu inicjalizacji, ilości kroków, długości wygenerowanej ścieżki oraz gładkości ścieżki. Porównane zostały one zarówno z wykorzystaniem wcześniej przygotowanych map, jak i map wygenerowanych losowo. Zauważono również, że czas inicjalizacji nie jest zależny od mapy, a suma kątów jest zależna od długości ścieżki.

Summary

The major objective of this thesis was comparison of Q-matrix initialization algorithms in the Q-learning algorithm for path planning. At the beginning of the thesis various algorithms used for the purpose of path planning were introduced, including the Q-learning algorithm. Next the algorithms used for initialization of the Q-matrix were described in depth. Attempts to use neural networks for the purpose of initialiazation were also discussed. The Q-learning algorithm, as well as the initialiazation algorithms were coded in Python. At the end of the thesis the implemented algorithms were compared on the basis of execution time of the Q-learning algorithm, initialiazation time, number of steps taken by the algorithm, lenght of generated path and path smoothness. The algorithms were compared using prepared maps, as well as randomly generated ones. It was noticed that the initialization time is not dependent on the map and that the sum of angles is dependent on the path length.

Słowa kluczowe: planowanie ścieżki, Q-learning, uczenie maszynowe, roboty mobilne

Keywords: path planning, Q-learning, machine learning, mobile robots

Spis treści

Rozdział 1

Wstęp

W obecnych czasach jedną z najszybciej rozwijających się gałęzi robotyki jest robotyka mobilna. Dzięki stosunkowo małej wielkości roboty mobilne są wykorzystywane w rozmaitych dziedzinach. Wiele magazynów wykorzystuje obecnie te platformy do autonomicznego rozmieszczania oraz poboru towarów zastępując przy tym wielu magazynierów. Małe rozmiary tych robotów są również zaletą przy dojeździe do miejsc trudno dostępnych, co pozwala zarówno na bardzo dokładne wykonywanie zdalnej inspekcji urządzeń przemysłowych, jak i na dogląd procesów [3]. Powszechność robotów mobilnych można nawet zauważyć w wielu domostwach, gdzie pełnią rolę autonomicznych odkurzaczy. W trakcie przejazdu mapują one swoje otoczenie i po dokładnym jego zbadaniu generują trasę, która dokładnie pokrywa wytworzoną mapę. Po dokończeniu zadania czyszczenia roboty te wracają do swojej stacji dokującej w celu naładowania. W tym celu muszą zaplanować ścieżkę, której punkt końcowy znajduje się w tym samym miejscu co stacja dokująca.

Planowanie ścieżki jest jednym z zagadnień często poruszanych w robotyce [15]. W zadaniu tym przyjmujemy, że znane jest całe środowisko, po którym porusza się robot oraz wszystkie ruchome i statyczne przeszkody w nim występujące. Celem tego planowania jest wyznaczenie pewnej listy punktów tworzącej ścieżkę od punktu początkowego do punktu końcowego przy jednoczesnym braku kolizji z występującymi przeszkodami. Bardzo często występują również dodatkowe ograniczenia wynikające z dynamiki robota oraz jego fizycznych możliwości. W robotyce istnieje również zagadnienie powiązane z planowaniem ścieżki, czyli planowanie trajektorii. W planowaniu trajektorii dana ścieżka jest powiązana z czasem, co wymusza dodatkowe ograniczenia na robocie mobilnym.

Zadanie planowania ścieżki jest jednym z fundamentalnych zadań robotyki, wobec tego istnieje wiele algorytmów rozwiązujących to zadanie. Jednym z najbardziej znanych jest algorytm A*. Jest on powszechnie używany np. w grach wideo w celu planowania ścieżki dla przeciwników. Można go postrzegać jako rozwinięcie algorytmu Dikstry, jednak A* do oceny trasy używa funkcji heurystycznej, czyli funkcji, która aproksymuje koszt przejścia daną ścieżką. Dużą wadą tego algorytmu jest jego złożoność pamięciowa [5]. Algorytmy planowania ścieżki czerpią również ze zjawisk fizycznych jak np. algorytm Symulowanego wyżarzania (z ang. *Simulated Anneling*), czy też metoda sztucznych pól potencjałowych (z ang. *Artificial Potential Field*). Pierwszy z nich inspirowany jest zjawiskiem wyżarzania występującym w metalurgii. Polega on na rozgrzaniu metalu do odpowiedniej temperatury oraz utrzymaniu go w niej przez pewien czas. Następnie jest on powolnie studzony, co zmienia jego właściwości fizyczne [9]. Drugi z wymienionych algorytmów opiera się na zjawisku przyciągania lub odpychania cząsteczek w zależności od ich potencjału. Robot mobilny jest w nim traktowany jako cząsteczka z pewnym ładunkiem. Następnie przyjmowane jest, że przeszkody posiadają ładunek o takim samym znaku, a cel ładunek o znaku

przeciwnym. Dzięki temu cząsteczka jest odpychana przez pola przeszkód, a przyciągana przez punkt końcowy [8]. Inne algorytmy są oparte na zjawiskach zachodzących w biologii. Algorytm genetyczny wykorzystuje krzyżowanie oraz mutację występujące w ewolucji [16]. Niektóre sposoby planowania ścieżki opierają się na zachowaniach różnych zwierząt. Algorytm świetlików(z ang. *Firefly Algorithm*) opiera się na zachowaniu robaczków świętojańskich w tropikach oraz ich sposobie migania [17]. Z kolei algorytm mrówkowy (z ang. *Ant Colony Optimization*) czerpie inspirację z gniazd mrówek, a konkretnie losowego przemieszczania się mrówki w celu znalezienia pokarmu oraz zostawiania za sobą śladu feromonów podczas powrotu do kolonii.

Popularne w obecnych czasach uczenie maszynowe również zostało wykorzystane do rozwiązywania problemu planowania ścieżki [19]. Jednym z nurtów uczenia maszynowego jest uczenie przez wzmocnienie (z ang. *Reinforcment Learning*). Polega ono na symulowaniu zachowań agenta w danym środowisku oraz zmaksymalizowaniu otrzymanej nagrody. Środowisko często jest przedstawiane jako proces decyzyjny Markowa, jednak nie zawsze jesteśmy w stanie w ten sposób je zamodelować. Q-learning jest bezmodelową metodą uczenia przez wzmocnienie, dzięki czemu nie musimy posiadać modelu środowiska. Tworzona jest macierz Q, która zawiera wszystkie możliwe stany oraz wszystkie możliwe akcje w danych stanach. Podczas uczenia, agent wykonuje akcje w danym stanie, a macierz Q jest aktualizowana za pomocną równania Bellmana z odpowiednią nagrodą lub karą oraz wartością stanu otrzymanego w wyniku danej akcji [7]. W planowaniu ścieżki można przyjąć, że ruch, który spowodowałby kolizję z przeszkodzą otrzymuje karę, natomiast osiągniecie celu byłoby nagradzane.

Istotnym aspektem Q–learningu jest inicjalizacja macierzy Q. Często wykorzystywana jest inicjalizacja zerowa, tzn. wszystkie elementy macierzy są na początku ustawiane jako zero. Wadą takiego podejścia jest początkowa losowość w wyborze następnej akcji oraz stosunkowo duża liczba kroków potrzebna do otrzymania rozwiązania. Inną metodą jest inicjalizacja losowa, która również cierpi na podobne problemy, jednak w niektórych przypadkach potrafi przynieść lepsze rezultaty. Istnieją sposoby inicjalizacji macierzy Q pozwalające na znacznie szybsze otrzymanie odpowiedniego rozwiązania oraz pozwalają na znaczące zmniejszenie liczby kroków.

1.1 Teza

Wykorzystanie różnych algorytmów do inicjalizacji macierzy Q w zagadnieniu planowania ścieżki znacząco poprawia czas wykonywania algorytmu w porównaniu do inicjalizacji zerowej.

1.2 Podział pracy

W rozdziale drugim znajduje się ogólny opis zadania planowania ścieżki. Przedstawione zostały różne algorytmy wykorzystywane w celu rozwiązania tego zadania. Została również opisany algorytm Q–learningu, na którym opierana jest reszta pracy. W następnym rozdziale dokładnie przedstawione są metody inicjalizacji macierzy Q, które zostały zaimplementowane w celach porównania. Rozdział czwarty zawiera opis wykorzystanego oprogramowania oraz sprzętu symulacyjnego, a także prezentuje otrzymane wyniki oraz wnioski z nich wynikające. W rozdziale znajduje się również opis poszczególnych map wykorzystanych do porównania poszczególnych metod inicjalizacji. Ostatni rozdział stanowi krótkie podsumowanie całej pracy.

Rozdział 2

Planowanie ścieżki

Istnieje wiele algorytmów służących do rozwiązania zadania planowania ścieżki. Często różnią się one opisem środowiska. Ważne jest, czy algorytm zajmuje się zdyskretyzowaną reprezentacją mapy, czy też przestrzeń rozwiązań jest ciągła. Kolejnym ważnym aspektem algorytmów planowania ścieżki jest ich złożoność obliczeniowa. Mimo że w zadaniu przyjmowana jest pełna wiedza o otoczeniu robota mobilnego, ciągle zależy nam na minimalizacji czasu wykonywania, szczególnie gdy wykorzystujemy tylko sprzęt będący częścią robota. Innym aspektem, na który należy zwrócić uwagę, jest długość otrzymanej ścieżki. Jest ona jednym z prostszych sposobów oceny jakości ścieżki i często służy ona jako podstawa w porównywaniu algorytmów.

2.1 A*

Algorytm A* jest jednym z najpopularniejszych algorytmów stosowanych do rozwiązania problemu planowania ścieżki. Algorytm ten jest heurystyczny i wykorzystuje on informację na temat punktu docelowego w celu zmniejszenia liczby węzłów do przeszukania. Do wykorzystania tego algorytmu mapa powinna zostać zdyskretyzowana do postaci siatki punktów, na której można wyróżnić przeszkody oraz punkty początkowe i końcowe. Algorytm traktuje tę mapę jako graf, w którym każdy wierzchołek u_i jest punktem na mapie. Sąsiednie punkty są połączone ze sobą krawędziami $c(u_i, u_j)$ dla $i \neq j$, które posiadają pewien koszt przejścia $w(u_i, u_j)$ z punktu u_i do punktu u_j. Ważną częścią algorytmu jest heurystyka $h(u)$, która służy do estymacji kosztu ścieżki z wierzchołka u do wierzchołka docelowego u_d. Algorytm iteracyjne wybiera kolejne wierzchołki spośród zbioru dostępnych wierzchołków na podstawie

$$f(u) = g(u) + h(u), \tag{2.1}$$

gdzie $g(u)$ oznacza koszt dotarcia do punktu u z punktu startowego, a $h(u)$ jest omawianą wcześniej heurystyką. Wierzchołek jest w zbiorze dostępnym jeśli nie został wcześniej przetworzony oraz jeżeli istnieje krawędź łącząca go z punktem będącym wcześniej przetworzonym. Wybierany jest wierzchołek, dla którego funkcja $f(u)$ jest najmniejsza. Jeżeli wierzchołek ten jest docelowy, algorytm kończy pracę. W przeciwnym razie punkt jest dodawany do zbioru punktów przetworzonych oraz obliczana jest funkcja $f(u)$ dla wszystkich punktów osiągalnych z nowo dodanego punktu, które następnie są wprowadzane do zbioru punktów dostępnych. Jeżeli wierzchołek znajduje się już w tym zbiorze, porównywana jest nowo wyliczona wartość funkcji $f(u)$ z wartością otrzymaną wcześniej. Jeśli nowa wartość jest mniejsza, algorytm aktualizuje funkcję $f(u)$ zapisaną w pamięci.

2.2 Symulowane wyżarzanie

Algorytm symulowanego wyżarzania czerpie inspirację ze zjawiska wyżarzania występującego w metalurgii. Polega ono na podniesieniu temperatury metalu, utrzymaniu jej przez pewien czas w tym stanie oraz następnie powolnym studzeniu metalu. Sam algorytm jest probabilistyczny i służy do przybliżenia globalnego minimum danej funkcji [1]. Można powiedzieć, że każde rozwiązanie funkcji s jest analogią stanu, a funkcja minimalizowana $E(s)$ jest analogią energii wewnętrznej. Algorytm próbuje więc przejść z dowolnego stanu, do stanu z najmniejszą energią. W algorytmie na początku przyjmujemy dowolny stan oraz ustawiamy maksymalną wartość temperatury. Następnie losowane jest nowy stan s', bliski poprzedniemu. Później stan jest przyjmowany lub nie, z prawdopodobieństwem określonym tzw. funkcją akceptacji $P(E(s), E(s'), T)$. Przykładowa funkcja akceptacji wygląda następująco

$$P(E(s), E(s'), T) = \begin{cases} 1, & \text{gdy } E(s') < E(s) \\ e^{\frac{-\Delta E}{T}}, & \text{w przeciwnym wypadku,} \end{cases} \tag{2.2}$$

$$\Delta E = E(s') - E(s). \tag{2.3}$$

Oznacza to, że algorytm nie zawsze przyjmuje lepsze rozwiązanie, co pozwala aproksymować minimum globalne. Po dokonaniu tej zmiany następuje zmniejszenie temperatury. Sposób tego zmniejszenia określa się jako strategię chłodzenia, lecz często wykorzystuje się przemnożenie przez pewną stałą wartość $\alpha \in (0, 1)$. Dla zadania planowania ścieżki stanem i może być np. zbiór punktów $s_i = \left\{P^i_{start}, P^i_{start+1}, \ldots, P^i_{end-1}, P^i_{end}\right\}$, a funkcją oceny sumaryczny dystans. W celu uniknięcia przeszkód, przejście przez przeszkody często jest częścią funkcji oceny uwzględnione w następujący sposób

$$E(s) = \sum_{j=start}^{end-1} ||P^i_j - P^i_{j+1}|| + \beta \cdot L_c, \tag{2.4}$$

gdzie β jest pewnym dodatnim parametrem, a L_c jest liczbą odcinków przecinającą przeszkody. Algorytm kończy swoje działanie gdy temperatura osiąga określoną wcześniej temperaturę minimalną T_{min}.

2.3 Metoda sztucznych pól potencjałowych

Metoda sztucznych pól potencjałowych w celach planowania ścieżki została pierwszy raz zaproponowana w 1985 [8]. Metoda ta proponuje przyjęcie, że wszystkie przeszkody wytwarzają pole odpychające, podczas gdy cel wytwarza pole przyciągające. Zarówno pola odpychające, jak i przyciągające stają się silniejsze, im bliżej robot jest odpowiednio przeszkody lub punktu końcowego. Pole działające na robota jest sumą pola przyciągającego oraz wszystkich pól odpychających. Przyjmując pozycję robota jako $s = (x, y)$ oraz punkt docelowy jako s_d, wzór na pole przyciągające wygląda następująco

$$U_p = \frac{1}{2} k_p ||s - s_d||^2, \tag{2.5}$$

gdzie k_p jest współczynnikiem przyciągania. Pole to oddziałuje na robota z siłą $F_p = -\nabla U_p = k_p||s - s_d||$ Pole odpychające i-tej przeszkody jest zapisana jako

$$U^i_o = \begin{cases} \frac{1}{2} k^i_o \left(\frac{1}{||s - s^i_o||} - \frac{1}{d_0}\right)^2, & \text{gdy} ||s - s^i_o|| < d_0 \\ 0, \text{gdy} & ||s - s^i_o|| > d_0 \end{cases}, \tag{2.6}$$

gdzie k_o^i oznacza współczynnik odpychania i-tej przeszkody, s_o^i jej położenie, a d_0 jest maksymalnym dystansem oddziaływania pola. Analogicznie do siły przyciągania, siła odpychania i-tej przeszkody wynosi

$$F_o^i = -\nabla U_o^i = \begin{cases} -k_o^i \left(\frac{1}{||s-s_o^i||} - \frac{1}{d_0}\right) \frac{1}{||s-s_o^i||^2} & \text{, gdy } ||s-s_o^i|| < d_0 \\ 0 & \text{, gdy } ||s-s_o^i|| > d_0 \end{cases} \tag{2.7}$$

Robot poruszać się będzie w kierunku wyznaczanym przez siłę wypadkową $F = F_p + \sum F_o^i$. Metoda ta jest często wykorzystywana dzięki swojej prostocie, jednak istnieje szansa wpadnięcia w minimum lokalne przed osiągnięciem celu, gdzie siła wypadkowa będzie równa zeru, a robot przestanie się poruszać.

2.4 Algorytm świetlikowy

Algorytm świetlikowy jest metaheurystycznym algorytmem zainspirowanym miganiem robaczków świętojańskich. Insekty te poprzez proces bioluminescencji wytwarzają światło i służą jako sygnał do przyciągnięcia innych osobników. Jednym z czynników decydujących o sile zachęcania jest intensywność wytwarzanego światła. W algorytmie analogiem światła jest podawana funkcja minimalizacji. [14] Algorytm ten ma trzy podstawowe zasady:

- wszystkie świetliki są bezpłciowe oraz przyciągają się niezależnie od płci,
- poziom atrakcyjności świetlika jest proporcjonalny do ich jasności i dla jakiejkolwiek pary, osobnik świecący mniej będzie poruszał się w stronę jaśniejszego,
- funkcja celu służy do ocenienia jasności robaczków świętojańskich.

W algorytmie na początku generujemy losową populację z dostępnej puli rozwiązań minimalizowanej funkcji, wyznaczamy współczynnik absorpcji γ oraz współczynnik przyciągania β. Następnie dla każdego robaczka x obliczamy jasność $I = f(x)$. Potem w każdej iteracji t, dla każdej pary robaczków x_i oraz x_j, jeżeli $I_j < I_i$ robak i porusza się w stronę robaka j za pomocą równania

$$x_i^{t+1} = x_i^t + \beta e^{-\gamma r_{ij}^2}(x_i^t - x_j^t) + \alpha_t \varepsilon, \tag{2.8}$$

gdzie r_{ij}^2 jest odległością między świetlikami, α jest parametrem kontrolującym długość kroku, a ε jest losowym wektorem z pewnej dystrybucji. Po wykonaniu tego działania dla każdej pary ponownie jest wyliczana jasność dla każdego osobnika w populacji. Gdy ilość iteracji kończy się, rozwiązaniem będzie osobnik z najniższą wartością I. Dla zadania planowania ścieżki możemy planować prostą drogę do celu, aż do momentu zbliżenia się do przeszkody na pewną odległość. W tym miejscu do ominięcia przeszkody można wykorzystać algorytm świetlikowy, z funkcją celu minimalizującą odległość do celu przy zachowaniu minimalnej odległości od przeszkody. W każdej iteracji algorytmu najlepsze rozwiązanie staje się kolejnym punktem na trasie aż do osiągnięcia pewnej odległości od przeszkody, gdy znów zaczynamy rysować prostą drogę do celu. Zaletą tego algorytmu jest prostota jego implementacji.

2.5 Q-learning

Q-learning jest pewną metodą uczenia maszynowego, będącą przykładem uczenia przez wzmocnienie. W uczeniu przez wzmocnienie (z ang. *Reinforcment Learning*) agent wpływa na środowisko wykonując akcję oraz otrzymując nagrody lub kary w zależności od

jej efektu. Ta grupa algorytmów jest często wykorzystywana zarówno gdy mamy pełną informację o środowisku, jak i gdy informacja o środowisku jest niepełna. Z powodu skuteczności przybliżania polityki optymalnej uczenie przez wzmocnienie jest również wartościowym sposobem implementacji kontrolerów ruchu dla robotów mobilnych. Q-learning jest jedną z możliwych implementacji uczenia przez wzmocnienie. Wykorzystuje ona macierz Q o wymiarach $m \times n$, gdzie m jest ilością możliwych stanów systemu, a n jest liczbą akcji, które mogą zostać podjęte w każdym stanie. W macierzy tej największa wartość komórek macierzy w danym stanie s oznacza najlepszą możliwą akcję do podjęcia. W algorytmie na początku inicjalizowana jest macierz Q oraz ustawiany jest stan początkowy s_p. Następnie w każdej iteracji algorytmu agent podejmuje najlepszą dotychczasową akcję w danym stanie lub podejmuje akcję losową z prawdopodobieństwem ε. W przypadku gdy dwie akcje posiadają tę samą wartość, wybierana jest jedna z nich w sposób losowy. Po wykonaniu akcji agent otrzymuje nagrodę lub karę r oraz przechodzi do kolejnego stanu s'. Następnie aktualizuje się wartość wykonanej akcji w macierzy Q, posługując się tzw. równaniem Bellmana

$$Q(s,a) = Q(s,a) + \alpha \left(r + \gamma \max_a Q(s',a) - Q(s,a) \right), \tag{2.9}$$

gdzie α jest współczynnikiem uczenia, γ jest współczynnikiem dyskontowania, r jest nagrodą otrzymaną w stanie s, a $\max_a Q(s',a)$ jest maksymalną wartością akcji w stanie s'. Wybór akcji oraz wykonywanie ruchu jest powtarzane aż do osiągnięcia stanu końcowego s_k. Zarówno współczynnik uczenia, jak i dyskontowania są wartościami w zakresie $(0,1)$. Można by powiedzieć, że α odpowiada za ważność poprzedniej wartości w macierzy Q, a γ za ważność potencjalnej przyszłej nagrody. Parametr ε jest ważny ze względu na balans między eksploracją a eksploatacją i często jest zmniejszany wraz z postępem iteracji. Eksploracją nazywane jest osiąganie nowych, nieodwiedzonych wcześniej stanów, a eksploatacją wykorzystywanie uprzednio zdobytej wiedzy. Eksploracja pozwala nam na wyjście z minimów lokalnych oraz osiągnięcie polityki optymalnej. Aby wykorzystać Q-learning w zadaniu planowanie ścieżki, na początku należy zdyskretyzować mapę oraz liczbę możliwych do wykonania akcji. Każdy możliwy stan jest punktem na mapie, a akcją próba przemieszczenia się do punktu sąsiedniego. Dla tego zadania możliwą karą może być próba wyjścia poza obszar mapy lub uderzenie w przeszkodę, a nagrodą osiągnięcie punktu docelowego. Algorytm ten dąży do rozwiązania optymalnego, lecz często wymaga dużej ilości iteracji, by osiągnąć zbieżność ilości wykonywanych kroków. Inną wadą algorytmu jest stosunkowo duża ilość wykonywanych kroków w początkowej fazie eksploracji. Jednym z rozwiązań tego problemu są różne metody inicjalizacji macierzy Q.

Rozdział 3

Algorytmy inicjalizacji macierzy Q

Jednym z najważniejszych aspektów wykorzystania metody Q–learningu w zadaniu planowania ścieżki jest inicjalizacja macierzy Q. Wpływa ona bowiem zarówno na czas wykonywania samego algorytmu, jak i na otrzymaną ścieżkę. Z tego powodu istnieje wiele metod inicjalizacji macierzy Q, które wykorzystują wiedzę o środowisku będącą częścią zadania planowania ścieżki.

3.1 Tradycyjne metody inicjalizacji

Jednym z podstawowych oraz najczęściej wykorzystywanych metod inicjalizacji macierzy Q jest inicjalizacja zerowa. Polega ona na zapełnieniu macierzy wartościami zerowymi. Prowadzi to do całkowitej losowości w początkowych fazach eksploracji, co w większości przypadków skutkuje dużą ilością kroków, a tym samym wykonanych obliczeń. Jest ona często wykorzystywana, ponieważ nie wymaga ona żadnej wcześniejszej wiedzy o środowisku, ani dodatkowych obliczeń w trakcie inicjalizacji. Nie jest to jednak najlepsze rozwiązanie w przypadku zadania planowania ścieżki, ponieważ z założenia dostępna jest pełna wiedza o środowisku. Inną tradycją metodą inicjalizacji jest inicjalizacja losowa. Jak sama nazwa wskazuje polega ona na przypisaniu pewnej losowej wartości każdej akcji w macierzy Q. Inicjalizacja taka bardzo często ma potencjał być lepszą, ponieważ algorytm Q–learningu posiada dużo większą wiedzę o środowisku. Wiedza ta jest jednak niepewna i o ile komórki macierzy prowadzące do przeszkód szybko osiągną odpowiednią, ujemną wartość to komórki odpowiadające pustym stanom mogą być przez długi czas omijane, mimo że są częścią optymalnej trasy do celu. W dalszym porównaniu algorytmów inicjalizacji pod uwagę wzięta została tylko inicjalizacja zerowa.

3.2 Metoda sztucznych pól potencjałowych

Z powodu prostoty implementacji metody sztucznych pól potencjałowych często jest ona wykorzystywana wspólnie z innymi algorytmami planowania ścieżki lub też używana jest do ich modyfikacji. Idea wykorzystania metody pól potencjałowych do inicjalizacji macierzy Q polega na nadaniu komórkom macierzy Q wartości odpowiadającej polu potencjałowemu w danym stanie. Metoda da pozwala na uniknięcie początkowego etapu losowości w wyborze akcji, jak również sprzyja omijaniu przeszkód oraz dążeniu do punktu docelowego. Wykorzystanie metody sztucznych pól potencjałowych w celu inicjalizacji macierzy Q zostało zaproponowane oraz wykorzystane już kilkukrotnie [4], [2]. W tej pracy w celach porównawczych wykorzystana została metoda zaproponowana przez Tenq

Luo [12]. Polega ona na uproszczeniu obliczeń przez pominięcie odpychającego pola przeszkód oraz zmianę równania pola przyciągającego. Wadą tradycyjnej funkcji obliczającej pole przyciągające (2.5) jest to, że wartość pola zmniejsza się przy zbliżaniu się do punktu docelowego. Zmodyfikowana funkcja wykorzystywana do inicjalizacji wygląda następująco

$$V(s) = k_p e^{-\frac{1}{2}[(x-\mu_1)^2+(y-\mu_2)^2]}, \tag{3.1}$$

gdzie k_p jest współczynnikiem przyciągania, s jest pewnym stanem na mapie,(x, y) to koordynaty celu, a μ_1, μ_2 to koordynaty stany s. Wartość macierzy Q w stanie s dla każdej akcji a jest równa wynikowi otrzymanemu z równania (3.1). Zaletą tego rozwiązania jest zwiększanie się pola w ramach zbliżania się do punktu docelowego, jednak z powodu braku pola odpychającego, metoda ta jest podatna na słabą efektywność w radzeniu sobie z przeszkodami, które w tradycyjnej metodzie APF powodowałyby minima lokalne.

3.3 Optymalizacja oparta na zachowaniu wielorybów

Algorytm optymalizacji oparty na zachowaniu wielorybów (z ang. *Whale Optimization Algorithm*) jest kolejnym algorytmem metaheurystycznym inspirowanym naturą. Został on zaproponowany w roku 2016 oraz opiera się na zachowaniu długopłetwowców oceanicznym zwanych również humbakami [13]. Imituje on stadny sposób polowania tych zwierząt. Algorytm wykorzystuje zbiór agentów zwanych również „wielorybami" w celu znalezienia rozwiązania globalnie optymalnego. Na początku przyjmowane jest losowe rozwiązanie, które następnie jest aktualizowane aż do spełnienia kryterium stopu. W zadaniu inicjalizacji macierzy Q każdy agent reprezentuje pewną pozycję na mapie, a jakość danej pozycji jest określana przez symulację każdego możliwego ruchu w danej pozycji oraz aktualizację macierzy Q poprzez równanie Bellmana. W algorytmie można rozróżnić trzy kroki: otaczanie ofiary, strategia sieci bąbelkowej oraz szukanie ofiary.

3.3.1 Otaczanie ofiary

W trakcie tego kroku każdy agent dokładnie zna położenie ofiary oraz ją otacza. Ponieważ pozycja optymalnej wartości nie jest wcześniej znana, za położenie celu przyjmowana jest pozycja agenta z największą wartością macierzy Q. Podczas otaczania ofiary, agenci aktualizują swoją pozycję wykorzystując położenie celu. Matematyczny model tego zachowania prezentuje się następująco

$$\vec{D} = |\vec{C} \cdot \vec{G}(t) - \vec{X}(t)|, \tag{3.2}$$

$$\vec{X}(t+1) = \vec{G}(t) - \vec{A} \cdot \vec{D}. \tag{3.3}$$

W powyższych równaniach $\vec{X}(t)$ jest pozycją danego agenta w iteracji t, $\vec{G}(t)$ jest najlepszym rozwiązaniem w danej iteracji oraz $\vec{D}$ reprezentuje odległość między danym agentem, a obecnym najlepszym rozwiązaniem. Wektory $\vec{A}$ oraz $\vec{C}$ reprezentują wektory współczynników. Jeżeli istnieje lepsze rozwiązanie, wektor $\vec{G}(t)$ powinien zostać zaktualizowany w obecnej iteracji. Wektory współczynników obliczane są w następujący sposób

$$\vec{A} = 2\vec{a} \cdot \vec{r_a} - \vec{a}, \tag{3.4}$$

$$\vec{C} = 2 \cdot \vec{r_c}. \tag{3.5}$$

W powyższych równaniach $\vec{r_a}$ oraz $\vec{r_c}$ są losowymi wektorami w zakresie $[0, 1]$, a $\vec{a}$ jest liniowo zmniejszany od 2 do 0 w trakcie danej iteracji.

3.3.2 Strategia sieci bąbelkowej

W trakcie polowania wieloryby poruszają się wokół ofiary po spiralnej ścieżce, jednocześnie wypuszczając bańki powietrza w celu w skierowania ofiary w kierunku powierzchni. Takie zachowanie znane jest jako strategia sieci bąbelkowej. Mechanizm otaczania oraz zacieśniania spirali są osiągane poprzez zmniejszenie wartości $\vec{a}$ w równaniu (3.4). Zakres wektora $\vec{A}$ zmniejsza się wraz ze zmniejszeniem $\vec{a}$. Zgodnie z równaniem (3.4), wektor $\vec{A}$ posiada losowe wartości w zakresie $[-a, a]$. Ustawienie $\vec{A}$ jako losowej wartości w zakresie $[-1, 1]$ powoduje losowy ruch agenta pomiędzy obecną pozycją optymalną a poprzednią pozycją. W takim wypadku nowa pozycja agenta jest obliczana przez

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos 2\pi l + \vec{G}(t), \tag{3.6}$$

$$\vec{D'} = |\vec{G}(t) - \vec{X}(t)|. \tag{3.7}$$

W powyższych równaniach $\vec{D'}$ reprezentuje dystans pomiędzy danym agentem a obecnym najlepszym agentem, l jest losową liczbą w przedziale $[-1, 1]$, a b jest współczynnikiem decydującym o kształcie spirali logarytmicznej i powinien być dostosowywany w zależności od wymogów danego zadania.

W trakcie każdego kroku danej iteracji. pozycja agenta jest aktualizowana poprzez otaczanie ofiary lub strategię sieci bąbelkowej. Model aktualizacji pozycji danego agenta w kroku t prezentuje się następująco

$$\vec{X}(t+1) = \begin{cases} \vec{G}(t) - \vec{A} \cdot \vec{D}, \text{gdy } p < 0.5 \\ \vec{G}(t) + \vec{D'} \cdot e^{bl} \cdot \cos 2\pi l, \text{gdy } p >= 0.5 \end{cases}, \tag{3.8}$$

gdzie p jest pewną liczbą z przedziału $[0, 1]$

3.3.3 Poszukiwanie ofiary

Oprócz wykorzystywania strategi sieci bąbelkowej w trakcie poszukiwania ofiary, agenci muszą również szukać rozwiązania losowo w procesie eksploracji. Matematyczny model tego zachowania prezentuje się następująco

$$\vec{D} = |\vec{C} \cdot \vec{X_{rand}}(t) - \vec{X}(t)| \tag{3.9}$$

$$\vec{X}(t+1) = \vec{X_{rand}}(t) - \vec{A} \cdot \vec{D}, \tag{3.10}$$

gdzie $\vec{X_{rand}}(t)$ jest pozycją losowego agenta z populacji.

W celu zapewnienia odpowiedniej eksploracji oraz zbieżności, gdy $|\vec{A}| \geqslant 1$ wykorzystuje się losowego agenta jako pozycję celu. W przeciwnym wypadku ($|\vec{A}| < 1$) pod uwagę brana jest pozycja najlepszego dotychczasowego rozwiązania [10].

3.4 Algorytm zapylania kwiatów

Zapylanie jest procesem transferu pyłku danego kwiatu do innego kwiatu. W około 90% kwiatów dochodzi do zapylania krzyżowego, gdzie pyłki kwiatów są przenoszone przez np. wiatr lub pszczoły, a w 10% kwiatów zachodzi samozapylenie, w którym pyłek pochodzi z tego samego kwiatu lub innego kwiatu tej samej rośliny. Algorytm zapylania kwiatów (z ang. *Flower Polination Algorithm*) jest kolejnym algorytmem metaheurystycznym zainspirowanym naturą, który został wykorzystany w celach inicjalizacji macierzy Q [11]. Został on początkowo przedstawiony przez Yang'a w 2012 roku [18]. Algorytm ten upraszcza proces zapylania wykorzystując 4 zasady:

- zapylanie globalne wykorzystuje zapylanie krzyżowe. W trakcie przenoszenia pyłku zapylacz podąża dystrybucją lotów Levy'ego,
- zapylanie lokalne wykorzystuje proces samozapylania,
- prawdopodobieństwo rozmnażania zależy od podobieństwa między dwoma kwiatami,
- prawdopodobieństwo $p \in [0, 1$ służy do kontroli wystąpień zapylania lokalnego oraz globalnego.

Pierwszą oraz trzecią zasadę dotyczące zapylania globalnego można matematycznie zaprezentować jako

$$x_i^{t+1} = x_i^t + \lambda L(\delta)(x_i^t - g^*), \tag{3.11}$$

gdzie x_i^t jest pyłkiem i w iteracji t, g^* jest najlepszym dotychczasowym rozwiązaniem, γ jest czynnikiem skalowania odpowiadającym za długość kroku, a $L(\delta)$ odpowiada sile zapylania.

Dobrym sposobem modelowania owadów zapylających, które poruszają się z różnymi długościami kroków przez długie dystanse jest wykorzystanie lotów Levy'ego. Dystrybucja Levy'ego opisana jest poprzez

$$L \sim \frac{\lambda \Gamma \sin \pi\lambda/2}{\pi} \frac{1}{s^{1+\delta}}, \tag{3.12}$$

gdzie s jest pewną podstawową wartością kroku większą od 0, a Γ to funkcja gamma.

Druga oraz czwarta zasada odnoszą się do zapylania lokalnego i modelowane są przez

$$x_i^{t+1} = x_i^t + \varepsilon \left(x_j^t - x_k^t \right), \tag{3.13}$$

gdzie ε jest liczbą losową z zakresu $[0, 1]$, a x_j^t oraz x_k^t są losowo wybranymi kwiatami z populacji.

Podobnie jak w algorytmie opartym na zachowaniu wielorybów każdy kwiat reprezentuje pewną pozycję na mapie oraz jest oceniany poprzez wykonanie wszystkich możliwych ruchów w danej pozycji i aktualizację macierzy Q używając równania Bellmana. Jakością danego kwiatu jest najwyższa wartość macierzy Q w jego pozycji.

3.5 Połączenie metod

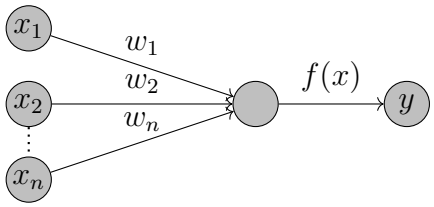
Zarówno algorytm zapylania kwiatów, jak i algorytm wielorybów reprezentują poszczególnych agentów w populacji jako stany na mapie. W trakcie oceny poszczególnych agentów obie metody wykorzystują wykonanie każdego możliwego ruchu z pozycji agenta oraz aktualizację macierzy Q równaniem Bellmana. Następnie wartością funkcji oceny jest maksymalna wartość macierzy Q w danej pozycji. Algorytmy te posiadają jednak tę samą wadę, jak oryginalne wykorzystanie Q–learningu – w obu przypadkach macierz Q jest początkowo inicjalizowana zerami co wydłuża czas obliczeń danych algorytmów. Wadę tę można by jednak zniwelować wykorzystując prostą metodę inicjalizacji, która wykorzystuje posiadane wiadomości o mapie. Proponowana metoda polega wstępnej inicjalizacji z wykorzystaniem metody sztucznych pól potencjałowych, a następnie wykorzystanie tak zainicjalizowanej macierzy Q jako początkowej macierzy w algorytmie zapylania kwiatów lub algorytmie wielorybów. Taka fuzja pozwoliłaby na poprawę czasu obliczeniowego w obu metodach oraz zniwelowałaby występujący problem minimów lokalnych w inicjalizacji metodą sztucznych pól potencjałowych.

3.6 Sieć neuronowa

W obecnych czasach zagadnienia związane z uczeniem maszynowym są powszechnie wykorzystywane w wielu celach. Jednym z takich zagadnień jest sieć neuronowa. Jest to algorytm zainspirowany działaniem ludzkiego mózgu. Sieć ta składa się z połączonych ze sobą węzłów zwanych sztucznymi neuronami. Każdy neuron składa się z wejść, wag oraz wyjść. Wejściami neuronu są dane otrzymywane z innych neuronów. Wagi neuronu to liczby określające wpływ danego wejścia na wyjście neuronu. Wyjście neuronu jest liczone poprzez przemnożenie wejść neuronu przez ich wagi oraz dodanie dodatkowej wartości niezależnej od wejść zwanej bias. Obliczona wartość jest przekazywana do funkcji aktywacji, która determinuje ostateczne wyjście neuronu. Graficzna reprezentacja neuronu znajduje się na rysunku 3.1. Równanie opisujące działanie pojedynczego neuronu wygląda następująco

$$y = f\left(\sum_{i=1}^{n} x_i \cdot w_i + b\right), \tag{3.14}$$

gdzie x_i oznacza wejście i , w_i jest wagą tego wejścia, b jest biasem danego neuronu, $f(x)$ jest wybraną funkcją aktywacji, a y jest wyjściem neuronu.



Rysunek 3.1: Reprezentacja jednego neuronu w sieci neuronowej

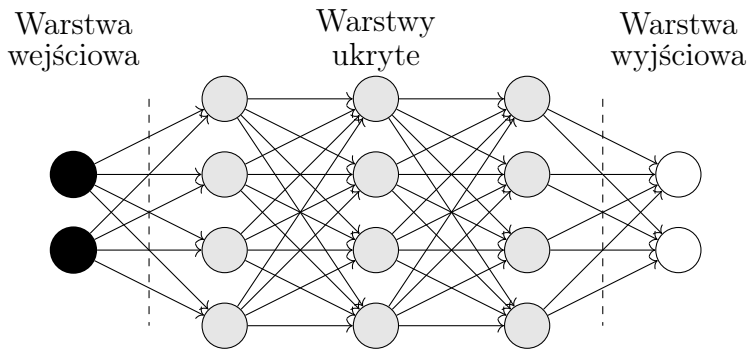
Neurony te ułożone są w poszczególne warstwy. Warstwa wejściowa przyjmuje dane. Warstwy środkowe, zwane również ukrytymi służą do odpowiedniego przetwarzania danych, a warstwa wyjściowa jest ostatecznym wynikiem działania sieci neuronowej. Poglądowy rysunek sieci neuronowej znajduje się na rysunku 3.2. Wagi wszystkich neuronów są aktualizowane w procesie uczenia sieci neuronowej. Do tego procesu potrzebne są określone pary wejść oraz oczekiwanych wyjść. Następnie wyjście sieci neuronowej jest porównywane z oczekiwanym wyjściem oraz obliczany jest błąd. Bardzo często wykorzystuje się błąd średniokwadratowy opisywany następująco

$$MSE = \frac{1}{n}\sum_{i}^{n}\left(Y_i - \hat{Y_i}\right)^2, \tag{3.15}$$

gdzie MSE oznacza błąd średniokwadratowy, Y_i oznacza i-tą wartość przewidzianą przez sieć neuronową, $\hat{Y_i}$ oznacza wartość rzeczywistą, a n jest liczbą wyjść sieci neuronowej.

Sieci neuronowe, w których każdy neuron jest połączony z każdym neuronem następnej warstwy, nazywane są sieciami gęstymi. Są one również wykorzystywane w zadaniu planowania ścieżki np. w algorytmie Deep Q-learnignu. W algorytmie tym sieć neuronowa służy zastąpieniu tradycyjnej macierzy Q. Wejściem takiej sieci jest stan agenta, a wyjściem jest akcja, która powinna zostać wykonana oraz jej estymowana wartość w macierzy Q.

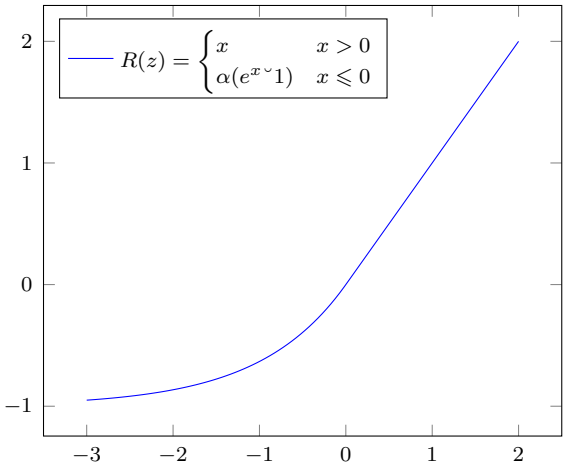
W pracy została również podjęta próba wykorzystania gęstej sieci neuronowej w celach inicjalizacji macierzy Q. Próba ta została podjęta w początkowych stadiach pracy. W związku z tym mapa posiadała wymiarowość 21 × 21 zamiast 64 × 64, a wejściem wstępnie zaprojektowanej sieci neuronowej był 445 elementowy wektor, gdzie 441



Rysunek 3.2: Poglądowa reprezentacja sieci neuronowej

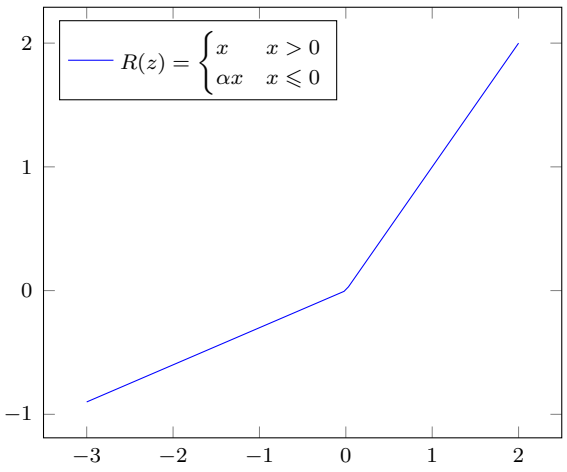
pierwszych pozycji odpowiadało spłaszczonej mapie zajętości, a ostatnie 4 elementy odpowiadały koordynatom punktu początkowego oraz punktu końcowego. Sieć składała się z 4 warstw, gdzie pierwsze dwie posiadały 445 neuronów, a następne dwie 1764 neurony, co odpowiadało wymiarom spłaszczonej macierz Q, która przed spłaszczeniem posiadała wielkość $21 \times 21 \times 4$. Funkcją aktywacji dla wszystkich warstw była wykładniczo-liniowa funkcja jednostkowa (z ang. *Exponential Linear Unit*), która jest zaprezentowana na rysunku 3.3. Funkcją błędu był omówiony wcześniej w równaniu (3.15) błąd średniokwadratowy. Dane do uczenia powyższej sieci były otrzymywane poprzez wygenerowanie losowej mapy, a następnie zastosowanie algorytmu Q–learningu. Otrzymana po szkoleniu macierz Q była wykorzystywana jako oczekiwana wartość w procesie uczenia. Dane szkoleniowe zawierały 16 384 próbki.

Niestety podejście to nie przyniosło oczekiwanych rezultatów. Pomimo dużej ilości próbek szkoleniowych zastosowanie macierzy Q utworzonej przez sieć neuronową skutkowało brakiem znalezienia ścieżki w liczbie kroków spełniającej ograniczenie 20 milionów kroków w jednej iteracji algorytmu. Mogło to wyniknąć ze zbyt dużej ilości połączeń między neuronami, a co za tym idzie, zbyt dużej ilości wag do dostrojenia. Kolejnym powodem mógł być brak zachowania informacji przestrzennej o mapie będącej wejściem sieci, przez co pomijane były ważne informacje wejściowe. Inną możliwością jest przeuczenie sieci – oznacza to, że sieć neuronowa zbyt dokładnie odwzorowywała wartości oczekiwane w procesie szkolenia, przy czym traciła ona możliwość przewidywania wartości, które nie pojawiły się wcześniej w zbiorze szkoleniowym.



Rysunek 3.3: Wykładniczo-liniowa funkcja jednostkowa

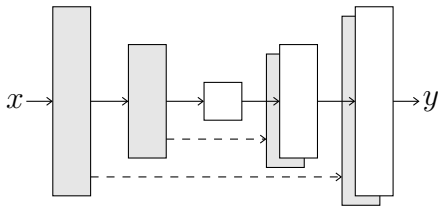
W celu poprawy wcześniej zidentyfikowanych błędów zostały wprowadzone następujące zmiany. Największą oraz najważniejszą z nich była zmiana architektury sieci na sieć konwolucyjną. Sieci konwolucyjne lub też splotowe są obecnie jedną z najpopularniejszych architektur sieci neuronowych wykorzystywanych w celach przetwarzania obrazów. Największą różnicą między gęstymi sieciami neuronowymi, a sieciami konwolucyjnymi jest liczba połączeń między kolejnymi warstwami danej sieci. Sieci splotowe ograniczają liczbę połączeń, a tym samym liczbę wag do dostrojenia za pomocą wykorzystania warstw konwolucyjnych. Warstwy te są stosowane przy danych wejściowych będących macierzą dwuwymiarową, na przykład obrazem. W trakcie operacji takiej warstwy po macierzy przesuwany jest filtr, również będący macierzą dwuwymiarową. W taki sposób otrzymujemy dwie macierze, jedną z nich są piksele obrazu przysłonięte przez filtr, a drugą jest sam filtr. Następnie wartości o tych samych współrzędnych są przez siebie mnożone, a wyniki mnożeń są sumowane. Suma ta jest wartością piksela obrazu będącego wyjściem warstwy konwolucyjnej. Taka operacja posiada kilka zalet: obliczenia uwzględniają relacje przestrzenne pomiędzy poszczególnymi pikselami obrazu wejściowego, wymiar obrazu wyjściowego tej warstwy jest zmniejszony oraz warstwa konwolucyjna z filtrem o wymiarach $x \times y$ posiada tylko $x \cdot y$ wag. Kolejnym zastosowanym krokiem była normalizacja wsadowa. Działa ona na zasadzie uśrednienia wartości średnich i odchyleń standardowych dla wszystkich aktywacji warstwy i wykorzystania uzyskanych wyników do normalizacji. Na wyniki tej warstwy została nałożona również rektyfikowana funkcja liniowa ReLU, która została zaprezentowana na rysunku 3.4.



Rysunek 3.4: Rektyfikowana funkcja liniowa

W sieci została wielokrotnie zaimplementowana kombinacja warstwy konwolucyjnej, normalizacji wsadowej oraz rektyfikowanej funkcji liniowej. Następnie stosowana jest seria warstw kolejno wykonujących operację splotu transponowanego, normalizacji wsadowej, warstwy porzucenia oraz ponownie rektyfikowanej funkcji liniowej. Warstwa porzucenia (z ang. *Dropout*) polega na losowym ustawieniu części wyjść na 0. Służy ona do walki z przeuczeniem, gdyż co każde przejście losowo wyłączane są połączenia, co zapobiega zbyt szybkiej nauce „na pamięć" danej sieci. Cała architektura sieci została oparta na generatorze wykorzystanym w sieci *Pix2Pix*, który z kolei jest zmodyfikowaną architekturą sieci *U-net* [6]. Graficzna reprezentacja sieci znajduje się na rysunku 3.5. Ostatnią zmianą było wykorzystanie pełnowymiarowej mapy 64 × 64 oraz próba wytrenowania sieci zarówno na macierzach Q otrzymanych przez algorytm Q–learningu z inicjalizacją zerową, jak i z połączeniem metod inicjalizacji. Podobnie jak poprzednio wykorzystany błędem

był błąd średniokwadratowy. Niestety, wykorzystanie macierzy Q uzyskanych przez sieci neuronowe po raz kolejny poskutkowało niemożliwością uzyskania ścieżki przez algorytm Q–learningu przy wykonaniu maksymalnie 20 milionów kroków w jednej iteracji.



Rysunek 3.5: Poglądowa reprezentacja zastosowanej sieci konwolucyjnej

Rozdział 4

Porównanie algorytmów

Q-learning oraz wszystkie omówione algorytmy inicjalizacji macierzy Q zostały zaimplementowane w języku Python. W znacznej części implementacji wykorzystywana była biblioteka Numpy w celu ułatwienia obliczeń oraz reprezentacji macierzy Q, która została zaimplementowana jako tablica $64 \times 64 \times 4$. W celu wytworzenia obrazów map oraz ścieżek wykorzystana została biblioteka Matplotlib. Sama mapa posiada dwie reprezentacje. Jedna określa zbiór przeszkód występujących na mapie, a druga jest mapą zajętości, w której 0 oznacza pole puste, a 1 pole zajęte przez przeszkodę. W celu stworzenia mapy zajętości, mapa jest najpierw rysowana za pomocą biblioteki Pillow jako czarno-biały obraz, a następnie obraz jest z powrotem przekształcany w tablicę Numpy. Testy z implementacją głębokiej sieci neuronowej, jak i konwolucyjnej sieci neuronowej wykorzystywały popularną bibliotekę TensorFlow w wersji 2.12. Całość obliczeń wykonana została na serwerze Linux z systemem operacyjnym Ubuntu 18.04, 16GB pamięci RAM oraz dwóch czterordzeniowych procesorach Intel Xeon E312xx.

Algorytmy porównywane były ze względu na ogólny czas wykonywania metody Q-learning, czas wykonywania samej inicjalizacji macierzy Q, długość uzyskanej ścieżki, średniej liczby wykonanych kroków w trakcie Q–learningu oraz gładkość uzyskanej ścieżki. Gładkość ścieżki ewaluowana jest jako suma kątów obrotu wykonanych przez robota mobilnego w trakcie pokonywania uzyskanej ścieżki. Wzór na obliczenie tej sumy jest następujący

$$\theta = \sum \left[\text{atan2} \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right) - \text{atan2} \left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}} \right) \right], \tag{4.1}$$

gdzie (x_i, y_i) reprezentuje i-ty punkt na otrzymanej ścieżce. Można intuicyjnie stwierdzić, że im mniejsza suma kątów obrotu, tym ścieżka jest gładsza, gdyż robot nie musi często zmieniać swojego kierunku ruchu. Można więc stwierdzić, że gładkość ścieżki jest odwrotnością sumy kątów obrotu ścieżki.

W celach porównawczych zostały przygotowane cztery mapy zawierające rzadko rozmieszone przeszkody, przeszkody rozmieszczone symetrycznie oraz przeszkodę o kształcie podkowy zawierającą minimum lokalne. Ponadto algorytmy zostały również przebadane za pomocą losowo wygenerowanych map, z gęstym rozmieszczeniem przeszkód. Każda próba wyznaczenia ścieżki została wykonana 10 razy dla każdego algorytmu. Wszystkie mapy posiadają wymiary 64 na 64. Na wszystkich mapach punkt startowy jest zaznaczony kolorem niebieskim, a punkt końcowy kolorem czerwonym.

Pierwszą mapą wykorzystaną do wstępnego przebadania algorytmów jest mapa z jedną przeszkodą o kształcie koła, występującą w równej odległości pomiędzy punktem startowym a końcowym. Na rysunku 4.1 zaprezentowane jest porównanie przykładowych ścieżek otrzymanych przy różnych metodach inicjalizacji macierzy Q. W celach zwiększenia

przejrzystości opisu algorytmy reprezentowane są następującymi skrótami wywodzącymi się z angielskich nazw algorytmów:

- Zero – Inicjalizacja zerowa,
- APF – Metoda sztucznych pól potencjałowych,
- FPA – Algorytm zapylania kwiatów,
- WOA – Algorytm wielorybów,
- APF-FPA – Połączenie metody sztucznych pól potencjałowych z algorytmem zapylania kwiatów,
- APF-WOA – Połączenie metody sztucznych pól potencjałowych z algorytmem wielorybów.

4.1 Mapa z okrągłą przeszkodą

Pierwszą przebadaną mapą była trywialna mapa z jedną przeszkodą pomiędzy punktem startowym, a punktem końcowym. Przykładowe ścieżki otrzymane przez poszczególne metody zaprezentowane są na rysunku 4.1. Istnieje kilka cech, które rozróżniają charakterystyki poszczególnych algorytmów. Inicjalizacja zerowa charakteryzuje się gwałtownymi zmianami oraz losowością otrzymanej ścieżki. Wynika to z braku jakiejkolwiek początkowej wiedzy o mapie, a w szczególności jest to spowodowane początkową losowością ruchów w fazie eksploracji. Ścieżka otrzymana inicjalizacją metodą sztucznych pól potenjałowych charakteryzuje się prostymi odcinkami oraz mocnym przyleganiem do przeszkód występujących na drodze do punktu końcowego. Powodem takiej charakterystyki jest prosta ocena pól względem odległości od punktu końcowego, skłaniająca agenta Q-learnigu do próby wyznaczenia ścieżki będącej prostą linią z punktu początkowego do punktu końcowego. Inicjalizacja algorytmem zapylania kwiatów skutkuje ścieżką z mniejszą chaotycznością, w porównaniu do inicjalizacji zerowej i większą preferencją dla prostych odcinków. Inicjalizacja macierzy Q poprzez algorytm wielorybów prowadzi do ścieżek, które często posiadają odcinki wzdłuż krawędzi mapy. Takie ścieżki są spowodowane próbą „otaczania" obecnego punktu optymalnego w trakcie inicjalizacji, co w przypadkach, gdy punkt ten jest odpowiednio zbliżony do krawędzi mapy, skutkuje zmianą pozycji agenta na pozycję wybiegającą poza określone granice mapy. W takim przypadku pozycja agenta jest ustawiana na najdalszą możliwą pozycję do osiągnięcia, to znaczy na krawędzi mapy. Przy inicjalizacji macierzy Q wykorzystując połączenie metody sztucznych pól potencjałowych z algorytmem zapylania kwiatów lub algorytmem wielorybów w obu przypadkach zachowane są cechy algorytmów wykonywanych po metodzie pól potencjałowych.

Najważniejsze statystyki czasów wykonywania algorytmu Q–learningu łącznie z inicjalizacją zaprezentowane są w tabeli 4.1, a statystyki czasów inicjalizacji poszczególnych algorytmów widnieją w tabeli 4.2. Porównując czasy wykonywania algorytmu w sekundach, można zauważyć, że metoda sztucznych pól potencjałowych jest znacząco szybsza niż inne algorytmy. Wynikiem tego jest prawdopodobnie prostota mapy, która nie posiada dużej ilości wąskich zakrętów ani przeszkód, dzięki czemu agent algorytmu może przez długi czas podążać wzdłuż ścieżki wyznaczonej przez inicjalizację początkową. Sama inicjalizacja jest wyjątkowo krótka i dla wszystkich map jej czas będzie wynosił 10 milisekund. Jak widać obecność przeszkód nie wpływa na algorytm inicjalizacji. Inicjalizacja zerowa

Tabela 4.1: Czasy wykonywania algorytmu Q–learningu dla mapy z okrągłą przeszkodą wyrażone w sekundach

Mapa	Min	Max	Średnia	Mediana
Zero	22.71	45.25	29.76	28.39
APF	0.71	1.23	0.93	0.96
FPA	22.49	38.72	30.97	31.08
WOA	5.19	43.67	17.79	10.39
WrapFPA	6.31	7.40	6.91	6.86
WrapWOA	4.38	5.27	4.78	4.66

Tabela 4.2: Czasy inicjalizacji otrzymane dla mapy z okrągłą przeszkodą wyrażone w sekundach

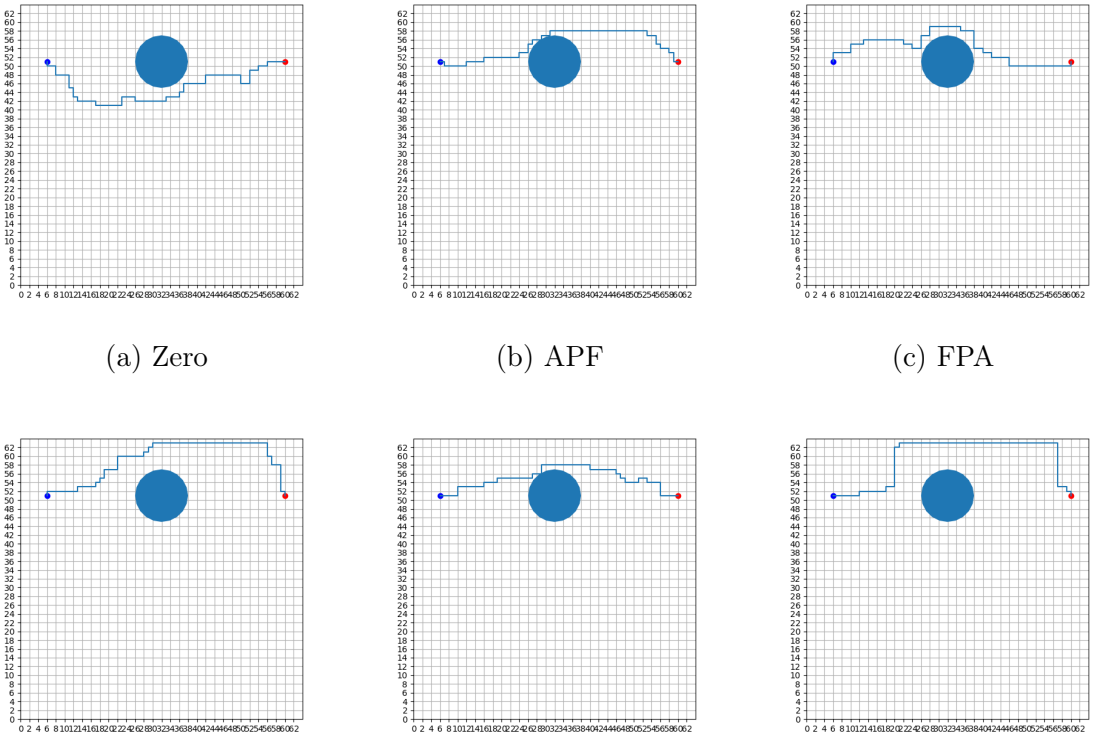
Algorytm	Min	Max	Średnia	Mediana
Zero	0.00	0.00	0.00	0.00
APF	0.01	0.01	0.01	0.01
FPA	5.82	6.40	6.11	6.12
WOA	4.28	5.10	4.76	4.75
WrapFPA	5.73	6.48	6.06	5.95
WrapWOA	4.24	4.97	4.61	4.53

oraz algorytm inicjalizacji kwiatowej posiadają bardzo podobne wyniki jeżeli chodzi o całkowity czas wykonywania algorytmu. Warto jednak zauważyć, że w wypadku algorytmu zapylania kwiatów sam Q-learning był krótszy, gdyż około 6 sekund było poświęcane na inicjalizację. Mała różnica w czasach całkowitych najprawdopodobniej wynika ze słabego przystosowania algorytmu zapylania do środowisk z małą ilością przeszkód. Trzecim najwolniejszym algorytmem jeżeli chodzi o całkowity czas wykonywania algorytmu był algorytm wielorybów. Średnia czasów tego algorytmu jednak jest wciąż zdecydowanie niższa niż inicjalizacja zerowa. Sama inicjalizacja zajmowała około 4,76 sekund. Dzięki wykorzystaniu obramowania mapy, algorytm ten osiągał przewagę nad inicjalizacją zerową, mimo niskiej liczbie przeszkód występujących na mapie. Wykorzystanie połączenia metod doprowadziło do lepszego wyniku niż zastosowanie samego algorytmu wielorybów czy też zapylania. W obu przypadkach zmniejszony został ogólny czas wykonywania algorytmu, przy czym zdecydowana większość została wykorzystana na inicjalizację.

Tabela 4.3: Średnia liczba kroków w Q–learningu dla mapy z okrągłą przeszkodą

Algorytm	Min	Max	Średnia	Mediana
Zero	4350.36	5930.56	5084.16	5069.12
APF	357.30	601.47	479.21	509.29
FPA	3720.08	5353.79	4469.30	4426.58
WOA	372.98	5527.56	2863.90	2406.93
WrapFPA	284.51	562.02	446.51	466.94
WrapWOA	76.08	164.39	94.00	86.89

Kolejną metodą oceny jakości metody może być średnia liczba kroków wykonanych



(a) Zero (b) APF (c) FPA

(d) WOA (e) APF-FPA (f) APF-WOA

Rysunek 4.1: Przykładowe ścieżki otrzymane dla mapy z okrągłą przeszkodą

przez algorytm Q–learningu przy danej inicjalizacji. Zebranie tych wyników widnieje w tabeli 4.3. Jak widać inicjalizacja zerowa, do której porównywane są inne metody inicjalizacji skutkuje w największej liczbie kroków wynoszącej około 5000. Następna w kolejności metoda to metoda zapylania. Nieznacząca poprawa jeśli chodzi o liczbę kroków podobnie jak w poprzednich przypadkach wynika z braku wystarczającej liczby przeszkód, by algorytm mógł działać poprawnie. Następnie ponownie plasuje się algorytm wielorybów, który dzięki wykorzystaniu brzegu mapy otrzymał średnio połowę liczby kroków wykonanych przez algorytm przy inicjalizacji zerowej. Następnie podobne wyniki zostały otrzymane zarówno przez inicjalizację metodą sztucznych pól potencjałowych, jak i jej kombinacją z algorytmem zapylania kwiatów. Najlepsza redukcja w ilości kroków wynika jednak z połączenia wcześniej wspomnianej metody sztucznych pól potencjałowych i algorytmu wielorybów. Przestrzenna natura algorytmu pozwoliła dokładnie wypełnić macierz Q w otoczeniu celu, przy odpowiednim pominięciu reszty mapy.

Tabela 4.4: Uzyskana długość ścieżki dla mapy z okrągłą przeszkodą

Algorytm	Min	Max	Średnia	Mediana
Zero	69.00	87.00	78.40	78.00
APF	69.00	79.00	72.80	72.00
FPA	69.00	87.00	78.60	79.00
WOA	73.00	99.00	83.20	85.00
WrapFPA	69.00	73.00	70.20	70.00
WrapWOA	75.00	99.00	85.00	86.00

Jeśli chodzi o długość uzyskanej ścieżki to najgorzej plasowały się metody wykorzystujące algorytm wielorybów. Po raz kolejny powodem jest tendencja do podążania brzegiem mapy, co znacznie zwiększa długość trasy. Następnie po raz kolejny istnieje mała różnica pomiędzy wynikami dla inicjalizacji algorytmem zerowym, a algorytmem kwiatowym. Kolejne w rankingu długości są inicjalizacja metodą sztucznych pól potencjałowych oraz jej połączenie z algorytmem zapylania kwiatów. Warto tutaj również zwrócić uwagę, że wszystkie metody oprócz tych opartych na algorytmie wielorybów, przynajmniej raz osiągnęły ścieżkę optymalną z długością wynoszącą 69 jednostek.

Ostatnim porównywanym aspektem jest suma kątów obrotu liczona wzdłuż ścieżki. Aspekt ten jest zaprezentowany na tabeli 4.5, i jest on liczony według wzoru (4.1). Największe wyniki otrzymywały metody oparte na algorytmach wielorybów, następnie inicjalizacja zerowa oraz metoda zapylania kwiatów, a najmniejsze wartości były otrzymane przez inicjalizację sztucznymi polami potencjałowymi oraz ich połączeniem z inicjalizacją kwiatową. Łatwo jest zauważyć, że suma kątów obrotu jest znacząco związana z samą długością uzyskanej ścieżki. Intuicyjnie ma to sens, gdyż dłuższa ścieżka wymaga często wymaga większej liczby obrotów. Można więc w tym przypadku powiedzieć, że największa suma oznacza również najdłuższą ścieżkę.

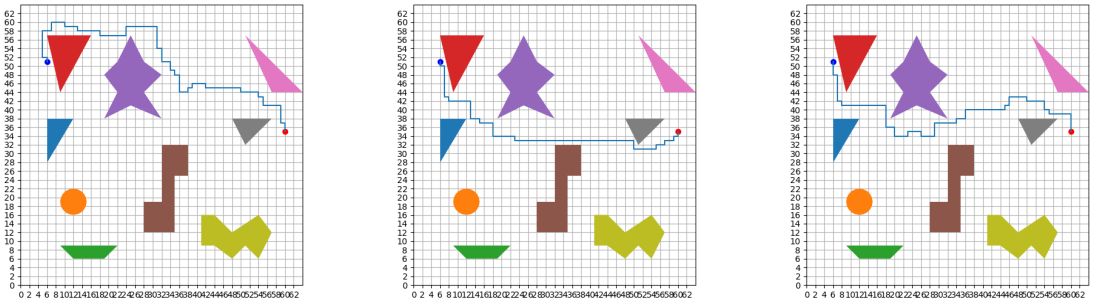
Tabela 4.5: Suma kątów obrotu liczona wzdłuż ścieżki dla mapy z okrągłą przeszkodą

Algorytm	Min	Max	Średnia	Mediana
Zero	21.99	48.69	35.58	34.56
APF	21.21	36.91	27.33	25.92
FPA	21.21	49.48	36.36	36.91
WOA	27.49	69.12	42.88	44.77
WrapFPA	21.21	27.49	23.56	23.17
WrapWOA	30.63	67.54	45.40	46.73

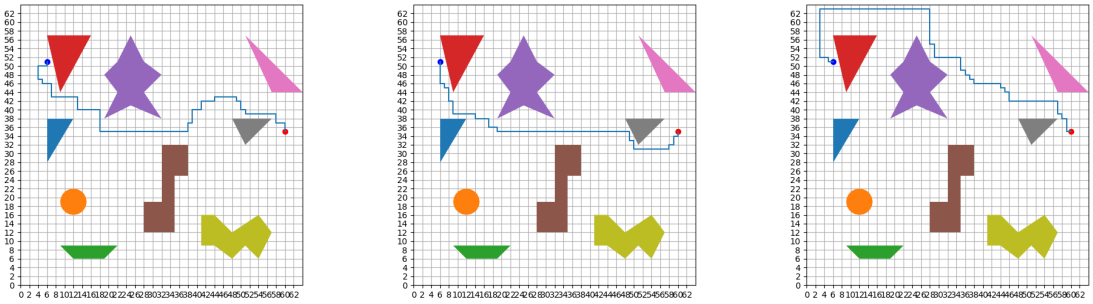
4.2 Mapa z wieloma przeszkodami w formie wieloboków

Następna mapa jaka została wykorzystana do porównania algorytmów to nietrywialna mapa zawierająca 9 przeszkód. Przeszkodami występującymi na mapie są zarówno proste kształty takie jak trójkąt, czy też koło oraz bardziej skomplikowane wielokąty. Przeszkody występujące na mapie są dość duże oraz posiadają duże odległości między sobą. Punkt startowy znajdował się w lewym górnym rogu mapy, a punkt końcowy pośrodku prawej jej krawędzi. Przykładowe ścieżki otrzymane przez poszczególne algorytmy znajdują się na rysunku 4.2.

Kształty otrzymanych ścieżek zachowują podobne prawidłowości jak kształty otrzymane dla mapy nr 1, jednak istnieje kilka różnic. Losowość oraz duża ilość zakrętów występująca w ścieżce otrzymanej poprzez inicjalizacje zerową została zachowana, podobnie jak preferencja do prosty ścieżek i ścisłego omijania przeszkód w inicjalizacji metodą sztucznych pól potencjałowych oraz jej kombinacji z algorytmem zapylania kwiatów. Również połączenie z algorytmem wielorybów zachowało swoją preferencję do obliczania ścieżki podążającej po krawędzi mapy. Nastąpił jednak zanik tej cechy w algorytmie wielorybów. Warto również zauważyć poprawę w jakości ścieżki otrzymanej przez metodę zapylania



(a) Zero (b) APF (c) FPA



(d) WOA (e) APF-FPA (f) APF-WOA

Rysunek 4.2: Przykładowe ścieżki otrzymane dla mapy z wieloma przeszkodami w formie wieloboków

kwiatów. Najprawdopodobniej wynika to ze zwiększenia ilości przeszkód, co pozytywnie wpłynęło na działanie obu wspomnianych algorytmów.

Tabela 4.6: Czasy wykonywania algorytmu Q–learningu dla mapy z wieloma przeszkodami w formie wieloboków w sekundach

Algorytm	Min	Max	Średnia	Mediana
Zero	21.09	49.59	36.82	36.54
APF	1.72	176.87	19.82	2.40
FPA	30.36	39.01	34.49	34.13
WOA	10.84	43.36	29.08	32.44
WrapFPA	6.94	195.05	44.43	8.09
WrapWOA	4.36	48.02	24.88	26.33

Ogólne czasy wykonywania algorytmu Q-learnigu oraz czasy wykonywania algorytmu inicjalizacji widnieją w tabelach 4.6, oraz 4.7. Czasy inicjalizacji nie są znacząco różne od czasów uzyskanych dla mapy nr 1, jednak wystąpiła zdecydowana różnica jeżeli chodzi o ogólne czasy wykonywania algorytmu. Algorytmem z najgorszym czasem okazało się połączenie algorytmu zapylania kwiatów z metodą sztucznych pól potencjałowych. Powodem tej zmiany jest to, że poprzez skomplikowanie mapy, algorytm w dwóch przypadkach z dziesięciu nie był w stanie znaleźć ścieżki w wyznaczonym limicie 10 milionów kroków w jednej iteracji. Podobny problem miała sama metoda pól potencjałowych, która

Tabela 4.7: Czasy inicjalizacji otrzymane dla mapy z wieloma przeszkodami w formie wieloboków w sekundach

Algorytm	Min	Max	Średnia	Mediana
Zero	0.00	0.00	0.00	0.00
APF	0.01	0.01	0.01	0.01
FPA	5.79	6.32	5.94	5.83
WOA	4.26	4.80	4.41	4.39
WrapFPA	5.59	6.35	5.80	5.74
WrapWOA	4.20	4.43	4.27	4.24

w jednym przypadku nie doprowadziła do znalezienia ścieżki, jednak krótki czas w pozostałych przypadkach testowych, dalej za skutkował najniższą średnią wśród algorytmów inicjalizacji. Wynikiem tego jest prostota wykorzystanej metody pól potencjałowych, która doprowadziła do częstych prób interakcji z przeszkodami przez agenta Q–learningu. Zawyżona średnia w tych przypadkach, jest jednak równoważona medianą znacząco mniejszą od innych metod inicjalizacji. Drugą najwyższą średnią jest inicjalizacja zerowa, a następnie, podobnie jak w poprzednim przypadku, algorytm zapylania kwiatów, algorytm wielorybów oraz jego połączenie z metodą sztucznych pól potencjałowych, osiągające drugą najniższą średnią.

Tabela 4.8: Średnia liczba kroków w Q–learningu dla mapy z wieloma przeszkodami w formie wieloboków

Algorytm	Min	Max	Średnia	Mediana
Zero	5047.57	6567.08	6048.55	6152.01
APF	919.89	122032.43	13235.61	1173.96
FPA	4310.45	6137.49	5050.76	4991.63
WOA	2386.96	6465.86	4464.82	4296.34
WrapFPA	605.05	131200.19	26137.87	1142.24
WrapWOA	84.02	25058.08	11637.02	12681.03

W wypadku średniej liczby wykonanych kroków w trakcie algorytmu Q–learningu zaprezentowanej w tabeli 4.8 można zauważyć kolejne ciekawe zależności. Trzy najwyższe średnie należą kolejno do kombinacji algorytmu zapylania kwiatów z metodą sztucznych pól potencjałowych, samej metody oraz do kombinacji z algorytmem wielorybów. Wynika to ze wspomnianych wcześniej efektów metody sztucznych pól potencjałowych, czyli częstych zderzeń agenta z przeszkodami. Interesujący jest fakt, że w wypadku połączenia z metodą wielorybów, zwiększona średnia liczba kroków, nie odpowiadały zwiększeniu średniego czasu.

Tabela 4.9: Uzyskana długość ścieżki dla mapy z wieloma przeszkodami w formie wieloboków

Algorytm	Min	Max	Średnia	Mediana
Zero	85.00	103.00	92.40	92.00
APF	79.00	200.00	94.30	81.00
FPA	81.00	99.00	91.60	91.00
WOA	85.00	147.00	96.40	92.00
WrapFPA	79.00	200.00	96.40	84.00
WrapWOA	83.00	109.00	100.00	101.00

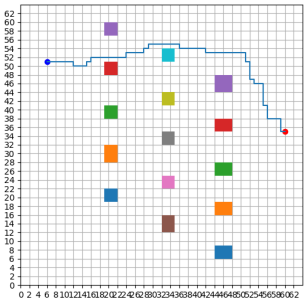
Średnie długości ścieżki dla mapy z wieloma przeszkodami w formie wieloboków oraz suma kątów obrotu są zaprezentowane w tabelach 4.9, oraz 4.10. Maksymalne wartości dla metody sztucznych pól potencjałowych oraz jej połączeniu z algorytmem zapylania kwiatów są skutkiem nieznalezienia ścieżki. W takim wypadku zarówno długość ścieżki, jak i suma kątów była zapisywana jako 200 jednostek. Warto zauważyć, że pomimo nieznalezienia ścieżki obie metody były w stanie znaleźć najkrótszą ścieżkę w przynajmniej jednym przypadku. Z tego powodu średnia długość jest podobna dla wszystkich algorytmów. Wszystkie algorytmy otrzymały podobną długość ścieżki, jednak połączenie sztucznych pól potencjałowych z algorytmem wielorybów wciąż skutkowało największą sumą kątów obrotu.

Tabela 4.10: Suma kątów obrotu liczona wzdłuż ścieżki dla mapy z wieloma przeszkodami w formie wieloboków

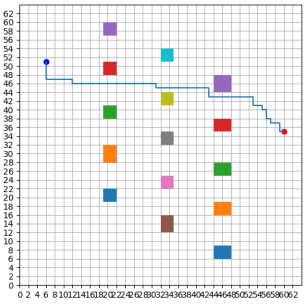
Algorytm	Min	Max	Średnia	Mediana
Zero	45.55	73.83	57.18	56.94
APF	36.13	200.00	57.78	39.27
FPA	40.06	67.54	55.92	54.98
WOA	46.34	142.16	63.77	56.94
WrapFPA	35.34	200.00	64.01	43.98
WrapWOA	41.63	83.25	69.12	70.69

4.3 Mapa z równomiernie rozmieszczonymi przeszkodami

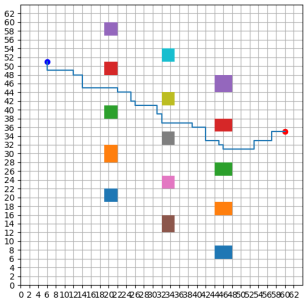
Następną przebadaną mapą była mapa z symetrycznie rozmieszczonymi kwadratami oraz prostokątami. Przykładowe otrzymane mapy są zaprezentowane na rysunku 4.3. Zachowania otrzymanych ścieżek są podobne do poprzednich przypadków, z wyraźnie zarysowanymi krzywymi wynikającymi z operacji okrążania w algorytmie wielorybów oraz ekstremalnym przypadkiem podążania po obrębie mapy w połączeniu z metodą sztucznych pól potencjałowych. Przypadek ten prawdopodobnie jest spowodowany wykluczeniem środkowych pozycji mapy z powodu zagęszczenia przeszkód przez algorytm wielorybów, co skutkowało większymi wartościami macierzy Q na obrębie mapy, gdyż były one wcześniej zainicjowane przez sztuczne pola potencjałowe.



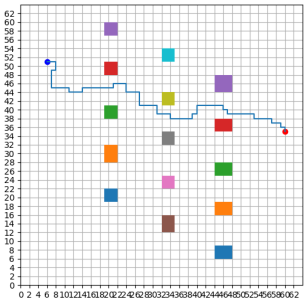
(a) Zero



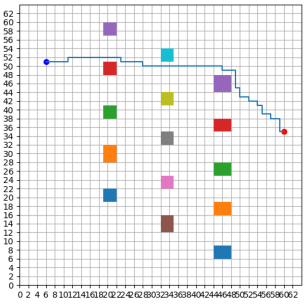
(b) APF



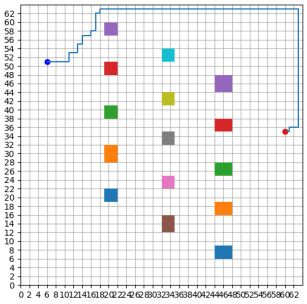
(c) FPA



(d) WOA



(e) APF-FPA



(f) APF-WOA

Rysunek 4.3: Przykładowe ścieżki otrzymane dla mapy z równomiernie rozmieszczonymi przeszkodami

Tabela 4.11: Czasy wykonywania algorytmu Q–learningu dla z równomiernie rozmieszczonymi przeszkodami w sekundach

Algorytm	Min	Max	Średnia	Mediana
Zero	21.15	33.16	29.32	29.59
APF	0.83	1.90	1.31	1.32
FPA	21.11	33.08	28.13	28.87
WOA	5.84	29.59	15.44	13.20
WrapFPA	6.65	7.58	7.04	7.12
WrapWOA	4.43	5.03	4.65	4.53

Tabela 4.12: Czasy inicjalizacji otrzymane dla mapy z równomiernie rozmieszczonymi przeszkodami w sekundach

Algorytm	Min	Max	Średnia	Mediana
Zero	0.00	0.00	0.00	0.00
APF	0.01	0.01	0.01	0.01
FPA	5.69	6.35	5.93	5.87
WOA	4.41	4.98	4.58	4.53
WrapFPA	5.66	6.45	5.94	5.83
WrapWOA	4.25	4.82	4.46	4.36

Ogólne czasy wykonywania algorytmu oraz czasy inicjalizacji w sekundach są zaprezentowane w tabelach 4.11, oraz 4.12. Po raz kolejny czasy inicjalizacji są bardzo bliskie poprzednim przypadkom, co sugeruje, że nie są one zależne od mapy. Jeżeli chodzi o średnie czasy wykonywania algorytmu Q–learningu otrzymane czasy są zbliżone do czasów otrzymanych przy mapie nr 1. Mimo zwiększenia ilości przeszkód tym razem metoda sztucznych pól potencjałowych nie miała problemu z wytworzeniem ścieżki. Oznacza to, że w wypadku dużej ilości małych przeszkód metoda może być wykorzystywana z dużym sukcesem. W przypadku średniej liczby kroków, uzyskanej długości ścieżki oraz sumy kątów obrotu, występują dokładnie takie same zależności jak w przypadku mapy z okrągłą przeszkodą.

Tabela 4.13: Średnia liczba kroków w Q–learningu dla mapy z równomiernie rozmieszczonymi przeszkodami

Algorytm	Min	Max	Średnia	Mediana
Zero	4087.69	5172.90	4616.64	4651.54
APF	453.27	858.79	669.81	703.63
FPA	3518.07	4420.16	3972.81	3963.08
WOA	531.51	5023.78	2563.79	2244.94
WrapFPA	516.61	717.67	583.66	558.27
WrapWOA	75.39	156.03	107.71	103.86

Tabela 4.14: Uzyskana długość ścieżki dla mapy z równomiernie rozmieszczonymi przeszkodami

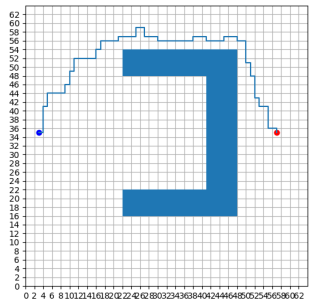
Algorytm	Min	Max	Średnia	Mediana
Zero	79.00	89.00	83.20	83.00
APF	71.00	77.00	72.80	73.00
FPA	75.00	89.00	79.20	79.00
WOA	73.00	105.00	83.00	78.00
WrapFPA	71.00	77.00	72.60	73.00
WrapWOA	71.00	107.00	94.20	101.00

Tabela 4.15: Suma kątów obrotu liczona wzdłuż ścieżki dla mapy z równomiernie rozmieszczonymi przeszkodami

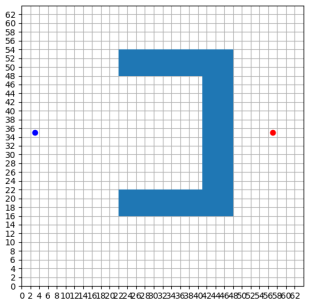
Algorytm	Min	Max	Średnia	Mediana
Zero	37.70	51.84	43.51	42.80
APF	23.56	34.56	27.49	27.49
FPA	31.42	52.62	37.70	36.91
WOA	28.27	77.75	43.28	35.34
WrapFPA	23.56	33.77	27.02	27.10
WrapWOA	25.13	78.54	59.85	70.29

4.4 Mapa z minimum lokalnym

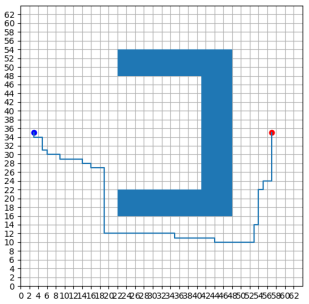
Mapa z minimum lokalnym zawierała jedną przeszkodę w kształcie podkowy zwróconej otworem w stronę punktu startowego. Przykładowe otrzymane ścieżki są zaprezentowane na rysunku 4.4. Można na nim zauważyć, że wszystkie metody inicjalizacji zawierające w sobie metodę sztucznych pól potencjałowych nie były w stanie uzyskać ścieżki przy spełnieniu ograniczenia 10 milionów kroków w jednej iteracji algorytmu. Prezentuje to ekstremalny przypadek, w którym agent Q–learningu nie jest w stanie opuścić podkowy z powodu inicjalizacji. Wartości macierzy Q sugerują agentowi prostą ścieżkę, jednak skutkuje to tylko częstą próbą zderzenia się z przeszkodą. Pozostałe trzy algorytmy cały czas prezentują takie same właściwości, z algorytmem wielorybów po raz kolejny preferującym ścieżki na krawędzi mapy.



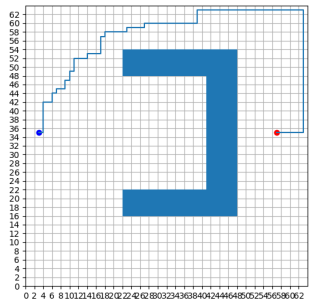
(a) Zero



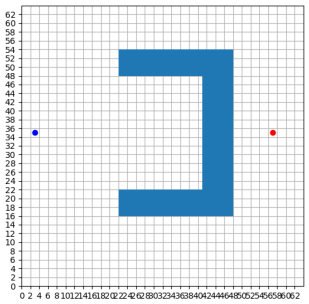
(b) APF



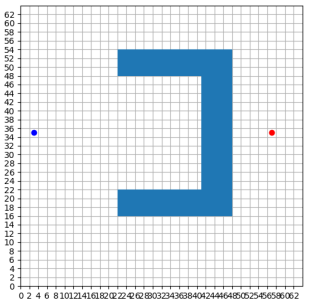
(c) FPA



(d) WOA



(e) APF-FPA



(f) APF-WOA

Rysunek 4.4: Przykładowe ścieżki otrzymane dla mapy z minimum lokalnym

Tabela 4.16: Czasy wykonywania algorytmu Q–learningu dla mapy z minimum lokalnym w sekundach

Algorytm	Min	Max	Średnia	Mediana
Zero	71.44	89.62	80.64	81.70
APF	174.80	377.24	203.95	187.65
FPA	57.22	79.91	66.31	63.05
WOA	11.05	80.83	35.86	21.92
WrapFPA	181.20	195.56	189.80	193.89
WrapWOA	182.62	193.89	190.71	192.90

Tabela 4.17: Czasy inicjalizacji otrzymane dla mapy z minimum lokalnym w sekundach

Algorytm	Min	Max	Średnia	Mediana
Zero	0.00	0.00	0.00	0.00
APF	0.01	0.01	0.01	0.01
FPA	5.68	6.66	6.19	6.32
WOA	4.32	5.00	4.73	4.80
WrapFPA	5.75	6.46	6.17	6.35
WrapWOA	4.28	4.87	4.67	4.81

Ogólne czasy wykonywania algorytmu Q–learningu oraz czasy wykonywania algorytmu inicjalizacji widnieją w tabelach 4.16, oraz 4.17. Można zauważyć na nich, że wykonanie ponad 10 milionów kroków w jednej iteracji zajmowało około 180 sekund oraz niezmienność czasów inicjalizacji. W jednym przypadku metoda pól potencjałowych była w stanie skończyć jedną iterację Q–learningu dzięki zastosowaniu prawdopodobieństwa losowych ruchów w algorytmie. Trzy najniższe czasy wykonywania to kolejno algorytm wielorybów, algorytm zapylania kwiatów oraz inicjalizacja zerowa. Algorytm wielorybów osiągnął najlepszy czas wykonywania z powodu preferowania krawędzi mapy. Średnia liczba kroków została zaprezentowana w tabeli 4.18 i odpowiada ona otrzymanym czasom.

Tabela 4.18: Średnia liczba kroków w Q–learningu dla mapy z minimum lokalnym

Algorytm	Min	Max	Średnia	Mediana
Zero	7880.53	11023.20	9719.88	9771.23
APF	6678584.00	10000001.00	9667859.30	10000001.00
FPA	6448.18	10435.28	8222.27	8223.54
WOA	2432.54	10619.15	5460.80	4252.74
WrapFPA	5001789.50	10000001.00	9500179.85	10000001.00
WrapWOA	10000001.00	10000001.00	10000001.00	10000001.00

Tabele prezentujące długość uzyskanej ścieżki oraz sumę kątów obrotu to kolejno 4.19 oraz 4.20. Po raz kolejny istnieje zależność pomiędzy długością ścieżki, a sumą kątów obrotu. Ponieważ metody inicjalizacji wykorzystujące sztuczne pola potencjałowe nie były w stanie wytworzyć odpowiedniej ścieżki ich wyniki zostały zastąpione liczbą 200. Pomimo otrzymania najniższej długości zarówno algorytm zapylania kwiatów, jak i algorytm wielorybów, posiadają większą średnią długość niż inicjalizacja zerowa.

Tabela 4.19: Uzyskana długość ścieżki dla mapy z minimum lokalnym

Algorytm	Min	Max	Średnia	Mediana
Zero	103.00	117.00	109.80	109.00
APF	200.00	200.00	200.00	200.00
FPA	99.00	133.00	120.80	124.00
WOA	99.00	127.00	117.80	123.00
WrapFPA	200.00	200.00	200.00	200.00
WrapWOA	200.00	200.00	200.00	200.00

Tabela 4.20: Suma kątów obrotu liczona wzdłuż ścieżki dla mapy z minimum lokalnym

Algorytm	Min	Max	Średnia	Mediana
Zero	73.83	96.60	84.98	83.64
APF	200.00	200.00	200.00	200.00
FPA	68.33	120.17	101.79	107.21
WOA	68.33	111.53	97.00	104.46
WrapFPA	200.00	200.00	200.00	200.00
WrapWOA	200.00	200.00	200.00	200.00

4.5 Mapy losowe

Ostatnimi badanymi mapami było pięć map losowych. Na mapach występowało pomiędzy 15, a 35 nieprzecinających się nieregularnych przeszkód, mapy były więc gęsto wypełnione przeszkodami. Istnienie ścieżki pomiędzy punktem początkowym, a końcowym było gwarantowane za pomocą wykorzystania algorytmu propagacji fali. Każda z wygenerowanych przeszkód spełniała kilka warunków:

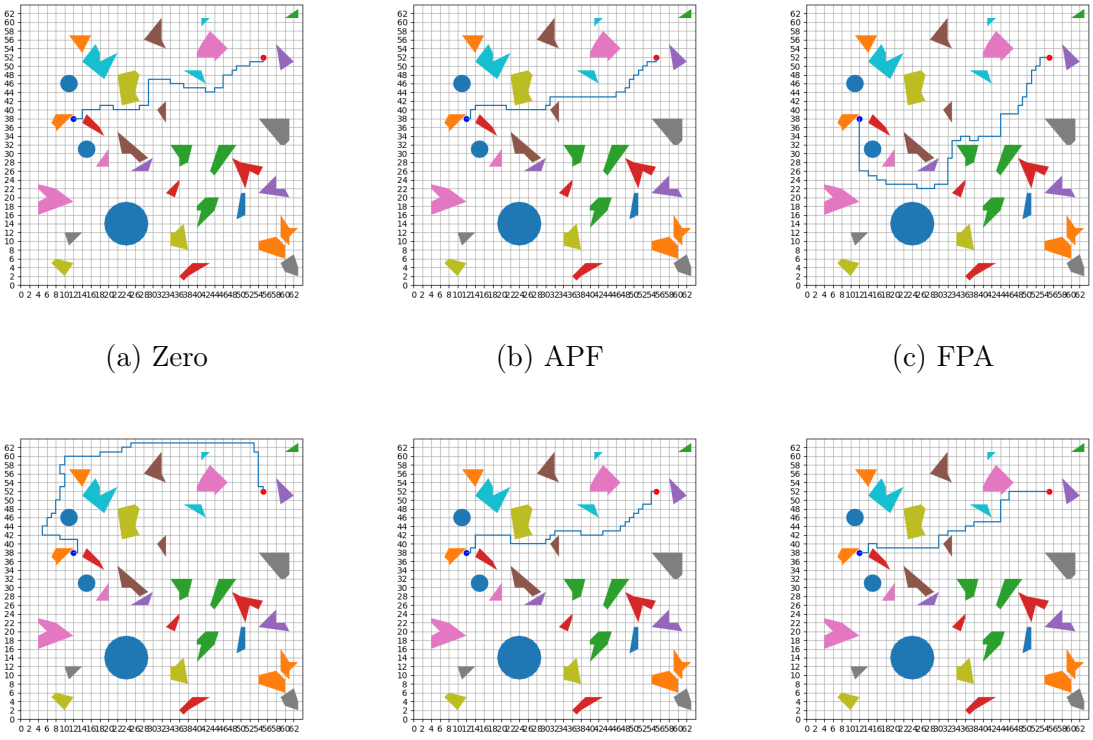
- była okręgiem z prawdopodobieństwem $p = 0.1$,
- była trójkątem z prawdopodobieństwem $p = 0.2$,
- była wielokątem z prawdopodobieństwem $p = 0.7$,
- każdy wielokąt posiadał pomiędzy 4, a 8 wierzchołków.
- odległość pomiędzy wszystkimi wierzchołkami wielokąta wynosiła pomiędzy 5, a 10 jednostek.

Przykładowa mapa oraz uzyskane ścieżki zaprezentowane zostały na rysunku 4.5. Mimo dużej gęstości przeszkód, w 50 testach, tylko raz nie została wygenerowana ścieżka.

Tabela 4.21: Czasy otrzymane dla map losowych w sekundach

Algorytm	Min	Max	Średnia	Mediana
Zero	15.08	45.91	28.38	28.63
APF	0.13	178.26	7.28	2.87
FPA	13.22	44.06	25.42	25.32
WOA	4.91	43.90	19.46	17.08
WrapFPA	5.78	22.21	9.45	8.73
WrapWOA	4.27	5.02	4.63	4.49

Czasy wykonywania algorytmu oraz czasy inicjalizacji zaprezentowane są w tabelach 4.21, oraz 4.22. Czasy inicjalizacji po raz kolejny są niezmienione, co potwierdza, że nie są one zależne od mapy. Czasy wykonywania poszczególnych algorytmów są podobne do poprzednich przykładów, z najdłuższym czasem uzyskanym przez inicjalizacje zerową, a następnie algorytm zapylania kwiatów, algorytm wielorybów oraz połączenie algorytmu zapylania i metody sztucznych pól potencjałowych. Najmniejszy średni czas uzyskało połączenie algorytmu wielorybów ze sztucznymi polami potencjałowymi, a drugim



(a) Zero (b) APF (c) FPA

(d) WOA (e) APF-FPA (f) APF-WOA

Rysunek 4.5: Przykładowe ścieżki otrzymane dla jednej z losowych map

Tabela 4.22: Czasy inicjalizacji otrzymane dla map losowych

Algorytm	Min	Max	Średnia	Mediana
Zero	0.00	0.00	0.00	0.00
APF	0.01	0.01	0.01	0.01
FPA	5.33	6.56	5.94	5.78
WOA	4.15	4.87	4.50	4.35
WrapFPA	5.45	6.48	6.02	5.88
WrapWOA	4.14	4.89	4.49	4.34

najszybszym była metoda sztucznych pól potencjałowych. Zamiana ta jest jednak spowodowana przypadkiem, w którym pola potencjałowe nie były w stanie uzyskać ścieżki. Widać to patrząc na mediany czasów, gdzie kolejność tych algorytmów jest zamieniona.

Średnia liczby kroków dla map losowych jest zaprezentowana w tabeli 4.23. Po raz kolejny można zauważyć zawyżoną średnią metody sztucznych pól potencjałowych. Największa średnia należy po raz kolejny do inicjalizacji zerowej, a następnie algorytmu kwiatowego, metody sztucznych pól potencjałowych, algorytmu wielorybów. Dwie metody z najmniejszą ilością kroków to połączenia algorytmów ze sztucznymi polami potencjałowymi, gdzie wykorzystanie algorytmu wielorybów uzyskało najmniejszą średnią liczbę kroków. Sugeruje to bardzo wysoką efektywność tej metody inicjalizacji w przypadku map z gęsto rozmieszczonymi przeszkodami.

Tabela 4.23: Średnia liczba kroków w Q–learningu dla map losowych

Algorytm	Min	Max	Średnia	Mediana
Zero	2989.26	10609.48	6362.51	6657.75
APF	64.03	120310.82	4383.21	1438.31
FPA	2311.60	7362.31	4719.74	4938.32
WOA	281.69	8098.35	3706.55	3715.64
WrapFPA	53.77	8154.55	1825.55	1409.29
WrapWOA	56.07	104.16	71.28	69.57

Średnia uzyskana długość ścieżki oraz suma kątów obrotu zaprezentowane są w tabelach 4.24, oraz 4.25. Mediany średnich wskazują na podobne wyniki inicjalizacji zerowej, algorytmu zapylania kwiatów oraz algorytmu wielorybów. Połączenie algorytmu wielorybów ze sztucznymi polami potencjałowymi zaskutkowało mniejszą medianą średniej długości, jednak najlepsze wyniki otrzymały metoda sztucznych potencjałów oraz jej połączenie z algorytmem kwiatowym.

Tabela 4.24: Uzyskana długość ścieżki dla map losowych

Algorytm	Min	Max	Średnia	Mediana
Zero	58.00	82.00	71.52	73.50
APF	50.00	200.00	77.00	64.00
FPA	54.00	98.00	72.76	73.00
WOA	56.00	98.00	73.64	73.50
WrapFPA	52.00	81.00	66.08	64.00
WrapWOA	56.00	86.00	68.96	69.00

Tabela 4.25: Suma kątów obrotu liczona wzdłuż ścieżki dla map losowych

Algorytm	Min	Max	Średnia	Mediana
Zero	19.63	191.64	76.36	56.16
APF	7.07	196.35	69.18	51.05
FPA	13.35	189.28	78.30	64.40
WOA	16.49	198.71	79.73	58.90
WrapFPA	10.21	193.21	67.87	51.05
WrapWOA	15.71	193.21	72.40	51.84

Rozdział 5

Podsumowanie

Wykorzystanie różnych metod inicjalizacji macierzy Q w wykorzystaniu algorytmu Q–learningu w zadaniu planowania ścieżki pozwala na zmniejszenie czasu inicjalizacji. Zaimplementowano algorytm Q–learningu w języku Python, z wykorzystaniem popularnych bibliotek Numpy oraz Matplotlib. Algorytm wykorzystywał podejmowanie losowych decyzji przez agenta z pewnym prawdopodobieństwem, które było zmniejszane wraz z rosnącą ilością iteracji i za które odpowiadał parametr ε. Przygotowano zarówno zestaw map służących do porównania metod inicjalizacji, jak i algorytm generujący mapy losowe. Przygotowane mapy reprezentowały trywialny przypadek z jedną przeszkodą, nierównomiernie rozmieszczone wieloboki, regularnie rozmieszczone prostokąty oraz wielobok w kształcie podkowy. Algorytm losowej generacji przeszkód skutkował dużą gęstością przeszkód oraz gwarantował on, że istnieje przynajmniej jedna ścieżka łącząca punkt początkowy oraz punkt końcowy za pomocą algorytmu propagacji fali.

Zaimplementowany został szereg metod inicjalizacji macierzy Q takich jak metoda sztucznych pól potencjałowych, algorytm zapylania kwiatów oraz algorytm wielorybów. Wykorzystana metoda sztucznych pól potencjałowych do inicjalizacji macierzy Q brała pod uwagę tylko odległość pola od punktu docelowego, ignorując przy tym przeszkody. Algorytm wielorybów oraz algorytm zapylania kwiatowego wykorzystywały zagadnienie populacji reprezentującej odpowiednio kwiaty oraz wieloryby. W obu przypadkach, do oceny danego osobnika zostało wykorzystane równanie Bellmana. Zaproponowane oraz zaimplementowane zostało również połączenie wybranych algorytmów z metodą sztucznych pól potencjałowych. Połączenie to polegało na wstępnej inicjalizacji macierzy poprzez metodę sztucznych pól potencjałowych, która następnie była wykorzystywana przez algorytm zapylania kwiatów lub też algorytm wielorybów.

Podjęta została próba wykorzystania głębokich sieci neuronowych oraz konwolucyjnych sieci neuronowych do inicjalizacji macierzy Q, jednak nie zostały otrzymane satysfakcjonujące wyniki. Głęboka sieć neuronowa została przebadana w początkowych stadiach pracy, gdy mapa posiadała zaledwie wymiary 20×20, zamiast 64×64, które były stosowane w późniejszych badaniach. Sieć ta mimo małych rozmiarów mapy posiadała dużą liczbę parametrów, przez co uczenie na zbiorze testowym nie było efektywne. Spowodowane to było tym, że każdy neuron każdej warstwy był połączony z wszystkimi neuronami warstwy następnej. Taka architektura skutkowała również brakiem zachowania informacji przestrzennej o mapie.

W celu rozwiązania tych problemów zostały wykorzystane sieci konwolucyjne. Architektura badanej sieci była oparta na sieci wykorzystywanej jako generator w sieci *Pix2Pix* [6]. Mimo zwiększenia wymiarów mapy, sieć ta posiadała znacząco mniejszą liczbę parametrów oraz zachowywała informację przestrzenną o mapie. Dodatkowo zbiór uczący zawierał

zarówno macierze uzyskane przez inicjalizację zerową, podobnie jak w poprzedniej wersji, ale również macierze uzyskane przez zaimplementowane połączenie metod. Podjęte działania nie przyniosły jednak oczekiwanych efektów, gdyż generowane macierze ponownie nie prowadziły do otrzymania ścieżki w wyznaczonym limicie 20 milionów kroków w iteracji algorytmu. Wynik taki został prawdopodobnie otrzymany poprzez wykorzystaną funkcję strat w procesie uczenia sieci neuronowej. Ponieważ zdecydowana większość wartości w macierzy Q uzyskanej w trakcie algorytmu Q–learningu jest bardzo bliska zeru, sieć generowała macierze zawierające pola o małej wartości niebędącej zerem. Generacja taka nie prowadziła do zachowania relacji pomiędzy poszczególnymi komórkami macierzy treningowej w wytworzonej macierzy.

Istnieje wiele kierunków dalszych badań na temat wykorzystania sieci neuronowych do inicjalizacji macierzy Q. Możliwe jest zastosowanie funkcji strat, która nie ocenia różnicy w wartości pomiędzy poszczególnymi komórkami macierzy, ale sprawdza, czy relacja między sąsiednimi komórkami jest zachowana w macierzy wyjściowej. Inną możliwością poprawy otrzymanych wyników mogłoby być wykorzystanie większej ilości danych w zbiorach uczących. Mogłoby to być osiągnięte poprzez zastosowanie metod rozszerzania danych wejściowych, powszechnie stosowanych w sieciach przetwarzających obrazy, takich jak obrót, czy też odbicie lustrzane.

Wszystkie zaimplementowane metody zostały porównane z inicjalizacją zerową macierzy Q. W celach porównawczych wykorzystane zostały ogólny czas wykonywania algorytmu, czas inicjalizacji macierzy Q, średnia ilość kroków podczas algorytmu Q–learningu, długość otrzymanej ścieżki oraz gładkość ścieżki. Algorytmy ogólnie prezentowały poprawę czasu wykonywania algorytmu, chociaż metody oparte na sztucznych polach potencjałowych wykazywały brak możliwości znalezienia ścieżki w obecności dużych przeszkód. Zostało to zauważone podczas badań z wykorzystaniem mapy o kształcie podkowy, gdzie żadna metoda wykorzystująca pola potencjałowe nie była w stanie wyznaczyć ścieżki. Stwierdzono również że długość czasu inicjalizacji nie jest zależna od mapy, a długość ścieżki koreluje z sumą kątów obrotu liczoną wzdłuż ścieżki. Wykonane badania mogłyby zostać poszerzone o kolejne algorytmy inicjalizacji takie jak np. metody oparte na sztucznych polach potencjałowcyh wykorzystujące również odpychające pola przeszkód. Innym możliwym kierunkiem rozwoju mogłoby być zwiększenie rozmiaru mapy, czy też sprawdzenie skuteczności algorytmów w przypadku środowisk trójwymiarowych. Można by również zbadać inne możliwe połączenia algorytmów jak np., połączenie algorytmu zapylania kwiatów z algorytmem wielorybów, czy też połączenie wszystkich trzech algorytmów. W związku z takimi badaniami można by również porównać wpływ kolejności wykonywanych algorytmów na inicjalizację.

Załącznik A

Do pracy załączono płytę DVD zawierającą w poszczególnych katalogach:

`/Praca_magisterska.pdf` — wersja cyfrowa pracy,

`/Kod_zrodlowy` — kod źródłowy zaimplementowanych algorytmów inicjalizacji, algorytmu Q–learningu, wykorzystanej reprezentacji mapy, skrypt przeprowadzający symulację dla wszystkich algorytmów oraz skrypt generujący dane statystyczne i rysunki ścieżek z otrzymanych wyników

`/Wyniki_symulacji` – otrzymane wyniki dla zbadanych algorytmów podzielone na podfoldery oznaczające daną mapę.

`/Sciezki_oraz_dane_statystyczne` – dane statystyczne oraz ścieżki otrzymane w wyniku symulacji.

Literatura

[1] M. de Sales Guerra Tsuzuki, T. de Castro Martins, F. K. Takase. Robot path planning using simulated annealing. strony 175–180, 2006. 12th IFAC Symposium on Information Control Problems in Manufacturing, St-Etienne, France.

[2] H. Du, B. Hao, J. Zhao, J. Zhang, Q. Wang, Q. Yuan. A path planning approach for mobile robots using short and safe q-learning. *PLOS ONE*, strony 1–20, 2022.

[3] S. Halder, K. Afsari. Robots in inspection and monitoring of buildings and infrastructure: A systematic review. *Applied Sciences*, 2023.

[4] B. Hao, H. Du, Z. Yan. A path planning approach for unmanned surface vehicles based on dynamic and fast q-learning. *Ocean Engineering*, 2023.

[5] P. E. Hart, N. J. Nilsson, B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, strony 100–107, 1968.

[6] P. Isola, J.-Y. Zhu, T. Zhou, A. Efros. Image-to-image translation with conditional adversarial networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, strony 5967–5976, 2017.

[7] B. Jang, M. Kim, G. Harerimana, J. W. Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, strony 133653–133667, 2019.

[8] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Proceedings. 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA*, strony 500–505, 1985.

[9] S. Kirkpatrick, C. Gelatt, M. Vecchi. Optimization by simulated annealing. *Science*, strony 671–680, 1983.

[10] Y. Li, H. Wang, J. Fan, Y. Geng. A novel q-learning algorithm based on improved whale optimization algorithm for path planning. *PLOS ONE*, strony 1–30, 2022.

[11] E. S. Low, P. Ong, K. C. Cheah. Solving the optimal path planning of a mobile robot using improved q-learning. *Robotics and Autonomous Systems*, strony 143–161, 2019.

[12] T. Luo. Improved reinforcement learning algorithm for mobile robot path planning. *2nd International Conference on Computer, Communication, Control, Automation and Robotics (CCCAR2022),Shanghai, China*, strony 20–30, 2022.

[13] S. Mirjalili, A. Lewis. The whale optimization algorithm. *Advances in Engineering Software*, strony 51–67, 2016.

[14] B. Patle, A. Pandey, A. Jagadeesh, D. Parhi. Path planning in uncertain environment by using firefly algorithm. *Defence Technology*, strony 691–701, 2018.

[15] N. Sariff, N. Buniyamin. An overview of autonomous mobile robot path planning algorithms. *4th Student Conference on Research and Development, Shah Alam, Malaysia*, strony 183–188, 2006.

[16] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 1998.

[17] X.-S. Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-inspired Computation*, 2010.

[18] X.-S. Yang. Flower pollination algorithm for global optimization. *Unconventional Computation and Natural Computation: 11th International Conference, Orléan, France*, strony 240–249. Springer, 2012.

[19] Y. Zhao, Y. Zhang, S. Wang. A review of mobile robot path planning based on deep reinforcement learning algorithm. *International Conference on Artificial Intelligence and Big Data Applications, Hubei, China*, strony 12–11. IOP Publishing, 2021.