Szymon Bałdyga, Krzysztof Lichy
Instytut Informatyki Politechnika Łódzka
Wólczańska 215, 90-924 Łódź
email: krzysztof.lichy@p.lodz.pl

WYKORZYSTANIE ALGORYTMU A-STAR W WYZNACZANIU TRAS NA AKWENACH WODNYCH

Streszczenie – Celem niniejszej pracy jest stworzenie rozwiązania pozwalającego żeglarzom na wyznaczanie tras na akwenach wodnych. Praca skupia się na praktycznym aspekcie wykorzystania algorytmu przeszukiwania grafów w zastosowaniu nawigacji żeglarskiej. Opisano niezbędne szczegóły techniczne oraz zaprezentowano stworzoną aplikację, jej możliwości, funkcjonalności oraz możliwe ograniczenia aplikacji. Przeprowadzono testy przygotowanego programu wraz z analizą oraz poprawianiem wydajności. Na koniec przedstawiono wnioski z przeprowadzonego badania, ocenę przydatności do zastosowania komercyjnego oraz zaproponowano potencjalne sposoby podniesienia efektywności procesu wyznaczania tras.

Słowa kluczowe: nawigacja śródlądowa, A-star, żeglarstwo

1 Wstęp

W dzisiejszych czasach jednym z aspektów funkcjonowania człowieka jest optymalne zarządzanie czasem i kosztami. Ludzie starają się optymalizować czas i środki wykorzystywane na różne aspekty działalności. Jednym z nich jest transport. Od kilku lat obserwowana jest tendencja do wykorzystywania nawigacji satelitarnych w celu skrócenia czasu niezbędnego na przemieszczanie się i odnalezieniu optymalnej drogi. O ile taki problem w zakresie przemieszczania się po lądzie został już zaadresowany i jest przedmiotem wielu komercyjnych produktów na rynku, o tyle zadanie wyznaczania optymalnej trasy na wodzie nie zostało rozwiązane w zadawalający sposób

Zakres niniejszej pracy obejmuje problem wyznaczania tras na śródlądowych akwenach wodnych. Rozpatrywane przypadki tras dobrano w taki sposób, aby ich przepłynięcie nie zajmowało dłużej niż od świtu do zmierzchu. W praktyce żeglarskiej przyjęto, że trasy planowanie są z wyprzedzeniem przy uwzględnieniu wielu czynników jak pogoda czy możliwości załogi. Z tego względu wyznaczanie trasy dla żeglarzy może trwać stanowczo dłużej, inaczej niż w przypadku nawigacji lądowej, gdzie użytkownicy przyzwyczajeni są do

kilkusekundowych procesów wyznaczania tras. Mając na uwadze powyższe, zakłada się, że akceptowalny czas wyznaczania trasy, którą będzie się podążać najbliższe kilka godzin, powinien wynosić nie więcej niż 10 minut. W hipotetycznym scenariuszu zastosowania, przeliczanie takiej trasy można zlecić rano, a następnie rozpocząć całodniową żeglugę.

Zakres pracy obejmuje korzystanie z takiego rozwiązania przez dwa typy statków wodnych: jachty motorowe i jachty żaglowe.

Jachty motorowe mogą poruszać się we wszystkich kierunkach, zatem nie istnieją ograniczenia co do kierunku poruszania się w przypadku podążania wyznaczoną trasą. Jachty żaglowe ze względu na naturę swojego napędu posiadają pewne ograniczenia, które powinny zostać uwzględnione przez rozwiązanie dostarczające trasę.

2 Proponowane autorskie rozwiązanie

Przeprowadzone badanie wykazało, że na rynku nie ma dostępnych odpowiednich narzędzi do nawigacji dla żeglarzy i motorowodniaków, które pozwalałoby w czasie rzeczywistym, bez dostępu do Internetu na wyznaczanie drogi. Przegląd rozwiązań zdecydowanie wykracza poza ramy tej pracy i można go znaleźć w [9].

Jako rozwiązanie tego problemu zaproponowano wykonanie aplikacji desktopowej, realizującej zadanie wyznaczania trasy wraz z uwzględnianiem kierunku wiatru oraz przeszkód wodnych takich jak wyspy czy mielizny. Do wykonania zadania zaproponowano wykorzystanie znanego z gier komputerowych algorytmu A*, którego rolą byłaby analiza węzłów na siatce mapy wyznaczonej na podstawie dostępnej mapy online z rozróżnieniem terenu wody od lądu. Stworzony w 1968 algorytm A* to jeden z najpopularniejszych algorytmów przeszukiwania grafów. Algorytm działa na zasadzie obliczania przybliżonego kosztu ruchu sąsiednich węzłów i wybieraniu najbardziej obiecującego. Do szacowania, algorytm wykorzystuje funkcję heurystyczną. To właśnie od tej funkcji zależy w głównej mierze zachowanie algorytmu, a co za tym idzie polityka wybierania kolejnych węzłów [11].

3 Opis działania i heurystyka

Algorytm A* w trakcie swojego działania przechowuje i zarządza dwiema listami: Open i Closed, czyli otwarte i zamknięte węzły. Zawierają one odpowiednio węzły możliwe do przeliczenia oraz te, które zostały już przeliczone i nie powinny być więcej rozpatrywane. Na początku lista open zawiera jedynie węzeł początkowy, natomiast lista Closed jest pusta. Sąsiednie (możliwe do przejścia węzły pochodne od

węzła startowego) są wyznaczane, a następnie ich koszt jest obliczany na podstawie sumy rzeczywistego kosztu przejścia, który jest znany oraz estymowanego na podstawie funkcji heurystycznej kosztu przejścia do punktu końcowego. Następnie wybierany jest nowy węzeł, którego koszt jest najmniejszy. Reszta punktów sąsiednich jest dodawana na listę Open. Punkt poprzedni jest dodawany do listy Closed, a najbardziej obiecujący punkt staje się punktem bieżącym. Następnie iteracja algorytmu powtarza się. Warunkami końcowymi algorytmu jest osiągnięcie punktu końcowego, czyli taka sytuacja, w której punkt bieżący jest jednocześnie punktem końcowym. W przypadku, gdy algorytm rozpatrzy wszystkie możliwe węzły z listy Open, wtedy algorytm kończy pracę bez wyznaczenia trasy – możliwa trasa nie istnieje [10].

Mając na uwadze zastosowanie i warunki w jakich będzie pracował algorytm zdecydowano się na zastosowanie heurystyki dystansu, która estymuje koszt trasy jako bezpośrednią odległość w linii prostej od punktu bieżącego do punktu końcowego.

4 Wyznaczenie węzłów siatki dla algorytmu na podstawie mapy

Głównym parametrem wejściowym algorytmu A* jest mapa podzielona na siatkę składającą się z kwadratów, których wierzchołkami są węzły. Przyjęto, że ilość wyznaczonych węzłów sąsiadujących (takich do których można przejść z węzła bieżącego) będzie mogła się zmieniać w zależności od ustawień algorytmu. Przykładowo w podstawowej wersji dany punkt generuje 8 punktów sąsiadujących w następujących kierunkach świata: (N,S,W,E,NW,NE,SW,SE).

To zachowanie można jednak zmienić poprzez zmianę liczby rozpatrywanych węzłów, przykładowo powyższą sytuację opisuje parametr o wartości 3, ponieważ dany punkt generuje sąsiadów w polu 3x3, gdzie punktem środkowym jest węzeł rodzic. Założono, że użytkownik może podać dowolną liczbę nieparzystą większą od 3, tak aby generowane pole zawierające sąsiadów i punkt rodzica zawierało się na planie kwadratu o polu X na X.

Przykładowo, dla wartości parametru 5 możliwe jest przejście w 24 kombinacjach - rozpatrując tylko daną iterację algorytmu. Warto zauważyć, że im większa wartość tego parametru tym większa ilość węzłów sąsiadujących do rozpatrzenia w każdej iteracji algorytmu. Zwiększenie tego parametru pozwala jednak na “pokonywanie” większych odległości w każdym ruchu. Na przykład: maksymalna odległość dla parametru 3 wynosi $\sqrt{2}$, dla wartości 5 jest to już $2\sqrt{2}$.

Kolejną przewidywaną zaletą zwiększenia liczby rozpatrywanych węzłów może być większa zdolność algorytmu do wyznaczania tras z mniejszą liczbą ostrych zakrętów. Warto zauważyć, że dla wartości

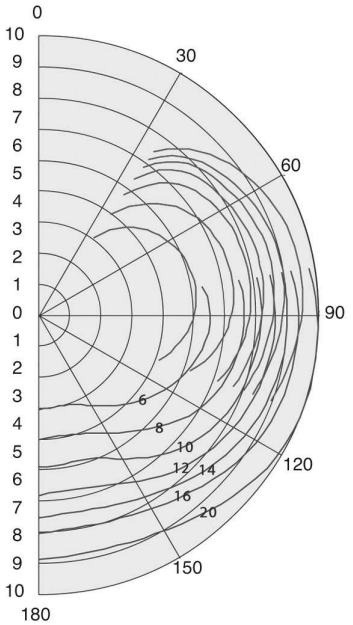
algorytmu 3 rozpatrując możliwe kierunki, mamy do dyspozycji 8 kombinacji z 360 stopniowego kąta pełnego. Daje to możliwość wyznaczenia trasy z dokładnością do 45 stopni. Przy zwiększeniu wartości parametru na 5, dostajemy znacznie większy wachlarz możliwości wyznaczonych kierunków trasy, ponieważ możliwe azymuty są rozmieszczone co 15 stopni. Można jednak wywnioskować, że zdolność algorytmu do skakania poprzez kilka węzłów naraz, może prowadzić do “przeskakiwania” wyznaczonej trasy również przez obszary, gdzie nie powinno mieć to miejsca - przykładowo przez teren lądowy.

Ze względu na założenia zakresu pracy zdecydowano się na uproszczony model mapowania modelu kulistej ziemi na model dwuwymiarowej przestrzeni. Algorytm generowania dwuwymiarowej mapy oblicza rozpiętość pionową i poziomą. Następnie według podanego parametru rozdzielczości mapy generuje odpowiednią liczbę węzłów mapując każdy węzeł o współrzędnych X i Y na dane szerokości i długości geograficznej. Podejście to generuje pewien błąd obliczeniowy, jednak nawet dla całodziennych tras żeglarskich nie przekracza on kilku metrów. Taką niedokładność należy potraktować jako akceptowalną mając na uwadze niedoskonałość użytej mapy oraz niedokładność potencjalnie wykorzystanego systemu pozycjonowania.

5 Teoria żeglowania względem wiatru

Niezbędnym elementem do analizy pracy algorytmu przy uwzględnieniu składowej wiatrowej jest zrozumienie teorii żeglowania i możliwości żeglugi jachtu względem kierunków wiatru.

Jacht żaglowy napędzany jest siłą wiatru. Jednak nie tylko siła, ale i kierunek wiatru mają znaczenie dla prędkości jachtu. Kąt między kierunkiem, z którego wieje wiatr a osią jachtu biegnącą od dziobu do rufy nazywamy kierunkiem względem wiatru. Kąt ten obliczany jest relatywnie od kierunku 0, który definiuje się jako dziób jachtu. Dla stałej siły wiatru i zmiennego kierunku jachtu względem wiatru przykładowy wykres prędkości jachtu przedstawiono na rysunku 1. Wykres przedstawia prędkość jachtu w zależności od kierunku, z którego wieje wiatr. Rysunek pokazuje działanie wiatru wiejącego tylko z jednej strony jachtu. Sytuacja na drugim halsie, czyli gdy wiatr wieje symetrycznie z drugiej strony jachtu jest dokładnie taka sama. Jacht żaglowy ze standardowym ożaglowaniem bermudzkim ma możliwość płynięcia w stosunku do wiatru pod różnym kątem z wyłączeniem kąta martwego . Jak widać na powyższym wykresie kąt ten obejmuje zakres 0 do około 45 stopni. Jest to dość standardowy kąt martwy dla turystycznych jachtów żaglowych. Jachty o sportowej charakterystyce posiadają znacznie mniejszy kąt martwy o wartościach nawet poniżej 35 stopni.

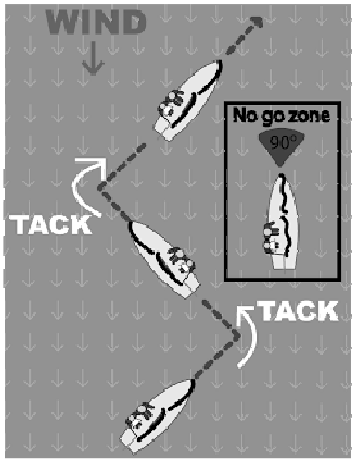


Rys. 1. Wykres prędkości jachtu w zależności od jego kursu względem wiatru [1]

Zjawisko kąta martwego definiuje zakres kierunków, w których jacht może płynąć napędzany siłą żagli. Standardowo zakres możliwych kursów jachtu względem wiatru zaczyna się więc od 45 i kończy na 315 stopniach, gdzie kierunek z którego wieje wiatr traktujemy jako kąt 0. Zatem jacht posiada dość szeroki zakres możliwych kursów z wyłączeniem 90 stopniowego wycinka, gdy wiatr wieje od strony dziobu.

Podróżowanie jachtu pod wiatr jest możliwe, jako wypadkowa maksymalnych kursów pod wiatr. Manewr ten przeprowadza się w następujący sposób, płynąc maksymalnym kursem pod wiatr (przy założeniu kąta martwego 90 stopni) czyli takim, gdzie kurs jachtu dąży do 45 stopni do wiatru wykonuje się dodatkowy manewr zwany zwrotem. Zwrot przez dziób polega na energicznym minięciu dziobem jachtu linii wiatru korzystając z rozpędu jachtu. Wykonanie tego manewru w krótkim czasie jest niezbędne, ze względu na fakt, że jacht w trakcie manewru nie jest napędzany siłą wiatru i musi wykonać cały manewr korzystając z wcześniej nabranej prędkości. Następnie po minięciu linii wiatru kontynuuje się manewr skrętu, tak aby zakończyć manewr około 45 stopni do wiatru, ale po drugiej stronie linii wiatru , czyli na drugim halsie.

Wykonywanie tego manewru w taki sposób, aby wypadkowy kurs kierował łodzią pod wiatr nazywany jest halsowaniem.



Rys. 2. Grafika przedstawiająca schemat manewrów halsowania [2]

W praktyce, biorąc pod uwagę dryfowanie jachtu oraz inne czynniki, halsowanie wygląda jak na poniższym rysunku.

Rys. 3. Zrzut ekranowy ze strony Endomondo z zapisaną trasą jachtu, który halsował pod wiatr [zbiory własne]

6 Projekt.

Aplikacja zawiera autorską implementację algorytmu A*, pozwalającą wyznaczać trasę na obszarach wodnych. Aplikacja korzysta z ogólnodostępnych map OpenStreetMap [6].

Przed implementacją zdefiniowano następujące wymagania:

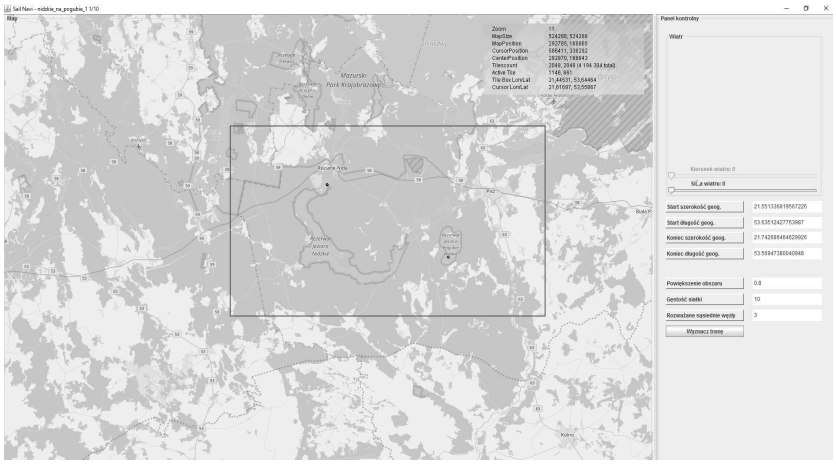
- aplikacja z interfejsem graficznym pokazującym trasę na mapie,
- panel boczny z potrzebnymi przyciskami i polami do kontroli prac programu,
- aplikacja powinna zawierać własną szybką implementację algorytmu A*,
- aplikacja powinna mieć możliwość dodania do obliczeń zmiennej wiatrowej,
- stworzenie ukrytego menu pobocznego do celów zebrania wyników dla analizy niniejszej pracy,
- możliwość ładowania scenariuszy testowych z pliku .txt,
- możliwość włączenia rysowania pracy algorytmu w czasie rzeczywistym.

Do napisania aplikacji wykorzystano:

- platformę Java 1.8 - przy budowaniu projektu wykorzystano JDK w wersji 1.8u72 dla systemu 64 bitowego [7],
- edytor kodu Netbeans 8.0.2 dla systemu 64 bitowego wraz z menadżerem projektu Maven,
- bibliotekę logującą log4j w wersji 1.2.17,
- bibliotekę automatyzującą testy jednostkowe JUnit w wersji 4.10,
- gotowy komponent do obsługi OpenStreetMap o nazwie MapPanel.

Interfejs graficzny został oparty o biblioteki Swing [8] dla Javy z głównym wykorzystaniem gotowego komponentu Mappanel. Komponent Mappanel posiada bardzo wiele przydatnych funkcji oraz gotowych funkcjonalności i doskonale nadaje się do zastosowania w omawianym projekcie. Komponent jest jednak dystrybuowany w postaci jednej klasy Java o ogromnej objętości. Dla czytelności kodu aplikacji zdecydowano się na refactoring kodu komponentu wraz z podzieleniem na moduły.

Interfejs graficzny aplikacji został podzielony na dwie części. Pierwsza część odpowiada za prezentację mapy oraz umożliwia jej przewijanie, podgląd pracy algorytmu oraz wyznaczonej trasy. Druga część to część odpowiadająca za kontrolę aplikacji. Do dyspozycji użytkownika są pola do wpisania koordynatów punktu startowego oraz końcowego, zdefiniowanie gęstości siatki oraz liczbę rozpatrywanych sąsiadujących węzłów. Aplikacja umożliwia również włączenie funkcjonalności, która dodaje do obliczeń zmienną wiatrową.



Rys. 4. Zrzut ekranu przygotowanej aplikacji „SailNavi”

Rys. 5. Zrzut ekranowy z wyznaczoną trasą

Interfejs graficzny prezentuje wyznaczoną trasą identyfikując elementy w następujący sposób:

- kolorem żółtym oznaczono węzły przetworzone – lista węzłów zamkniętych (closed list),
- kolorem białym oznaczono węzły do przetworzenia – na liście otwartych węzłów (open list),
- kolorem czarnym oznaczono wyznaczoną trasę,
- fioletowym kwadratem oznaczono rozważany obszar mapy.

7 Integracja z OpenStreetMap

Twórcy OpenStreetMap umożliwiają łatwy dostęp do map w postaci graficznej reprezentacji fragmentów mapy - kafli (ang. tiles). Pobranie

fragmentu odbywa się poprzez pobranie pliku graficznego z serwerów OpenStreetMap przy pomocy protokołu HTTP. Rozwiązanie przewiduje możliwość pobrania fragmentów mapy na 16-18 (w zależności od serwera hostującego) poziomach przybliżenia mapy. Poziom przybliżenia nie wpływa tylko na powiększenie prezentowanego obszaru, ale również zwiększa poziom szczegółowości mapy. Dzięki takiemu podejściu większy poziom przybliżenia daje nam podobny efekt jak przybliżanie map wektorowych, podobnie jak w popularnej mapie Google Maps. Warto zauważyć, że pobierane fragmenty są w rastrowym formacie PNG (ang. portable network graphics).

Usługa udostępniająca fragmenty map dla deweloperów posiada pewne ograniczenia. Zgodnie z polityką użycia deweloper nie ma prawa:

- ściągać większej ilości fragmentów mapy niż jest to wymagane przez działanie aplikacji
- pobrać jednorazowo większego obszaru mapy na dużym zbliżeniu mapy
- udostępniać zaimplementowanego rozwiązania w celach komercyjnych bez wcześniejszego ustalenia z administratorami usługi

Biorąc pod uwagę powyższe ograniczenia przy implementacji aplikacji zrezygnowano z buforowania pobieranych fragmentów map używając pracującego w tle niezależnego wątku. Zrezygnowano również z pobrania testowego obszaru mapy lokalnie, aby można go było wykorzystać niezależnie od dostępu połączenia internetowego.

W trakcie implementacji wykorzystano natomiast mechanizm lokalnego buforowania potrzebnych do pracy algorytmu fragmentów. Dzięki temu pomimo kilkukrotnego żądania pobrania danego obszaru mapy, obszar ten pobierany jest tylko jednorazowo w trakcie działania programu, a następnie jest wykorzystywany w różnych etapach aplikacji w zależności od przebiegu pracy algorytmu. Takie podejście ogranicza wykorzystanie dostępnych zasobów sieciowych, obciążenie serwerów hostujących usługę jednocześnie tylko nieznacznie zwiększa zapotrzebowanie na pamięć całej aplikacji.

8 Interfejs rozpoznawania obszarów lądowych oraz wodnych

Podstawową informacją niezbędną do działania algorytmu A* jest możliwość określenia, czy obiekt poruszający się po mapie może wykonać ruch na nową pozycję i jaki jest względny koszt tego ruchu. Aby umożliwić poprawną pracę algorytmu zaimplementowano interfejs wraz z klasą implementującą, która na podstawie dwóch współrzędnych jest w stanie określić, czy danych ruch jest możliwy. Dzięki kafelkowi mapy na podstawie definicji kolorystycznej reprezentacji wody na mapie jest w stanie rozpoznać, czy dany punkt jest obszarem wodnym.

Rozpoznawanie obszaru wodnego na podstawie koloru RGB na mapie to jedyny sposób uzyskania tej informacji z map OpenStreetMap. Takie podejście może jednak generować błąd, szczególnie w miejscach napisów, gdzie tekst na mapie napisano innym kolorem.

9 Generowanie węzłów na podstawie mapy

Do realizacji funkcjonalności mapy do celów obróbki przez algorytm A* został przygotowany interfejs AStarMap oraz jego podstawowa implementacja AStarBetterMapImpl. Skorzystano tutaj z interfejsu dla uniwersalności i łatwości zmiany implementacji. To właśnie od tej implementacji zależy mechanizm wyznaczania węzłów oraz sposób określania, czy dany obszar jest wodą czy nie. Mając na uwadze nie idealną dotychczasową implementację tego zagadnienia, na potrzeby testów przygotowano również poboczną implementację AstarFasterMapImpl. Jednak jest ona znacznie mniej precyzyjna i nie została ujęta w dalszych badaniach nad projektem.

10 Testowanie

Do analizy zaimplementowanego rozwiązania przygotowano przypadki testowe, które obejmują różne sytuacje i okoliczności wyznaczania trasie w potencjalnym praktycznym ujęciu. Rozważane punkty startowe i końcowe zostały dobrane celowo tak, aby testowały specyficzne sytuacje oraz nietypowe topograficznie lokalizacje, które mogą potencjalnie stanowić trudność w podjętym podejściu. Każda z tras została wyznaczona dla 7 różnych konfiguracji następujących parametrów wejściowych:

- gęstości siatki mapy,
- ilości rozważanych sąsiednich węzłów.

Dla wyznaczonych tras zebrano również czasy przetwarzania zadania. Przyjęto następujące zasady oceny tras:

- wyznaczone trasy zostały ocenione przez eksperta w skali 1-10,
- w przypadku braku wyznaczonej trasy, jeśli takowa jest możliwa przyznano ocenę 0,
- w przypadku gdy trasa została wyznaczona, a nie powinna nadano ocenę 0,
- trasy, w których doszło do wyznaczenia trasy nad terenem lądowym dostały ocenę niższą niż 5,
- test został przerwany, jeśli czas wyznaczania przekraczał 10 minut, a czas został oznaczony jako „10:00:00”.

11 Przykładowe przypadki

Opis wszystkich opracowanych i przetestowanych tras wykracza poza ramy niniejszej pracy. Poniżej zostaną przedstawione najbardziej interesujące przypadki. Więcej przykładów wraz z analizą można znaleźć w [9].

11.1 Z Mikołajek na Seksty

Punkty zostały dobrane w taki sposób, aby przetestować prosty przypadek wyznaczania trasy. Przewidywana trudność dla algorytmu to konieczność ominięcia cypla przy przejściu z jeziora Śniardwy na Seksty. Punkt końcowy został umieszczony celowo w zatoczce na jeziorze Seksty (nazywaną również jeziorem Kaczerajno). W tym miejscu trasa powinna ominąć cypel, a następnie zawrócić za nim w kierunku punktu docelowego (Rys. 6).

Ze względu na ograniczenia algorytmu w zakresie wykrywania terenu wodnego od terenu piaszczystego trudność może stanowić również oznaczony na mapie OpenStreetMap przebieg granicy rezerwatu Pierwos w okolicach miejscowości Stawek na jeziorze Mikołajskim. Granica ta oznaczona jest kolorem zielonym, co może spowodować, że wyznaczone w tym miejscu węzły nie będą odpowiadały definicji wody w programie i zostaną rozpoznane jako ląd. Istnieje zatem podejrzenie, że w tym miejscu program nie będzie mógł przekroczyć tej linii. Przyjęto:

punkt startowy 21.57460657630627, 53.79579810946028,

punkt końcowy 21.720446769981475, 53.701908933799814.

Rys. 6. Przypadek testowy numer 1 z Mikołajek na Seksty

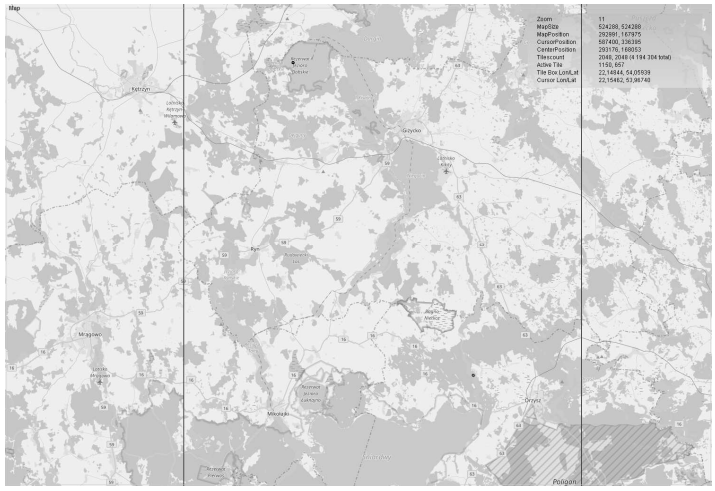
11.2 Ze Śniardw na Roś

Punkty zostały wyznaczone w taki sposób, aby przetestować wyznaczanie trasy z użyciem kanałów wodnych. Jedyną możliwą trasą przejścia jest tutaj kanał Jegliński prowadzący od jeziora Seksty na jezioro Roś. Patrząc na mapę można odnieść wrażenie, że droga wodna istnieje również w postaci rzeki Wyszki prowadzącej przez jeziora Białoławki oraz Kocioł. Ta droga zostałaby jednak odrzucona na podstawie przewodników i dokładniejszych map z batymetrią, ponieważ głębokość wody jest tam za mała dla jachtu, a mosty posiadają zbyt mały prześwit. Przewidywane zachowanie to próba wyznaczenia trasy zarówno przy pomocy rzeki Wyszka oraz kanału Jeglińskiego.

11.3 Z Dobskiego na Tyrkło

Jest to najdłuższy przypadek testowy wybrany do analizy. Jego trudność podnosi odległość między punktami oraz duża ilość kanałów, które należy przebyć, aby dotrzeć do punktu docelowego. Dodatkową trudność stanowią wielokrotne skręty, zatoczki oraz przeszkody które trasa musi ominąć. Fragment trasy między jeziorem Niegocin a jeziorem Tałty będzie musi prowadzić w kierunku dokładnie przeciwnym niż wskazywałby to punkt docelowy. Taka sytuacja powinna znacząco wydłużyć przeliczanie trasy, ponieważ przewiduje się że algorytm zgodnie z heurystyką liczącą odległość w linii prostej będzie sugerował że najlepsze węzły do dalszych obliczeń znajdują się w zupełnie przeciwnym kierunku. Przyjęto:

punkt startowy 21.596974966634683, 54.097853339088665;
punkt końcowy 21.860379726368578, 53.83093468644247.



Rys. 7. Przypadek testowy numer 4 z Dobskiego na Tyrkło

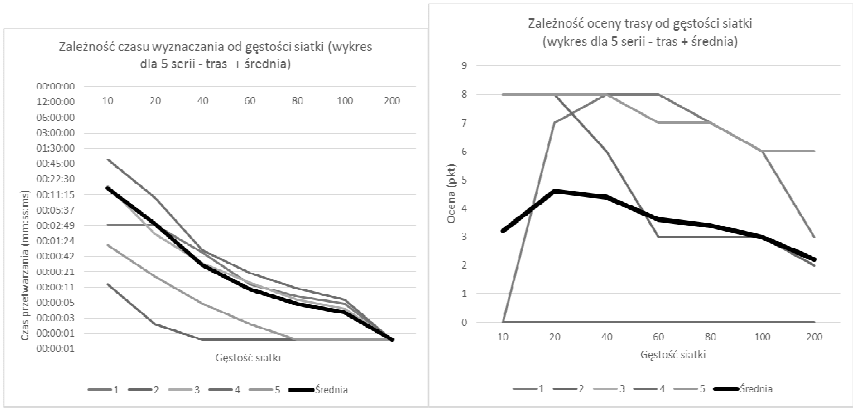
12 Znaczenie gęstość siatki jako parametru

Gęstość siatki, to parametr wyrażony jako liczba rzeczywista większa od 0. Wielkość tego parametru definiuje bok kwadratu siatki. Dla wartości parametru 1 węzły będą wygenerowane w rogach kwadratu o bokach 1m x 1m. Im mniejsza wartość parametru, tym więcej węzłów do rozpatrzenia. Należy się spodziewać, że im mniejsza gęstość siatki, tym algorytm będzie pracował szybciej. Dla mniejszej gęstości należy spodziewać się również mniej dokładnej i bardziej kanciastej trasy.

Przyjęte do pierwszej iteracji parametry wejściowe podano w Tabeli 1.

Tabela. 1. Tabela zawierająca ocenę trasy oraz czas wyznaczania trasy dla 5 przypadków testowych w zależności od gęstości siatki mapy

Przypadki testowe					
Gęstość siatki [m]	**1**	**2**	**3**	**4**	**5**
10	0 00:02:571	8 00:00:127	0 00:16:329	0 00:54:014	7 00:01:101
20	7 00:02:556	8 00:00:024	0 00:01:908	0 00:10:026	7 00:00:178
40	8 00:00:502	6 00:00:006	0 00:00:313	0 00:00:717	7 00:00:054
60	8 00:00:125	3 00:00:004	0 00:00:134	0 00:00:200	7 00:00:024
80	7 00:00:071	3 00:00:002	0 00:00:065	0 00:00:107	7 00:00:013
100	6 00:00:057	3 00:00:002	0 00:00:044	0 00:00:064	7 00:00:006
200	3 00:00:009	2 00:00:002	0 00:00:013	0 00:00:015	7 00:00:002



Rys. 8. Wpływ gęstości siatki na działanie algorytmu

W wyniku przeprowadzonych eksperymentów zaobserwowano następujące właściwości.

- Zgodnie z przewidywaniami zmniejszenie gęstości wpływa korzystnie na czas pracy algorytmu.
- Zgodnie z przewidywaniami zmniejszenie gęstości wpływa negatywnie na ocenę trasy, wyjątkiem jest tutaj różnica między siatką o gęstości 10m i 20m, gdzie mniejsza gęstość wpłynęła pozytywnie na kształt wyznaczonych tras w większości przypadków.
- Trasa numer 4 nie została wyznaczona w żadnym testowanym przypadku.
- Wszystkie iteracje zakończyły się poniżej zakładanych 10 minut.
- Dla trasy numer 1 dla najmniejszej gęstości siatki problem stanowiło przejście przez zaznaczona na mapie kolorem zielonym granicę rezerwatu. W tym miejscu algorytm rozpoznał kolor zielony jako teren inny niż woda i te węzły oznaczył jako niemożliwe do osiągnięcia. W ten sposób na wodzie powstała sztuczna granica, za którą algorytm nie mógł przejść dalej w poszukiwaniu drogi
- Algorytm ma tendencję do prowadzenia tras przy lądzie
- Dla przyjętych parametrów tej iteracji trasy są kanciaste, a skoki między węzłami odbywają się tylko w 8 kierunkach.
- Dla przyjętych iteracji nie udało się wygenerować trasy, która byłaby zdolna nawigować w wąskim kanale – wymagają tego trasy 4 i 5.
- Mając na uwadze zebrane pomiary, najlepszą wartością gęstości siatki jest 20 m.

13 Znaczenie ilości węzłów sąsiadujących jako parametru

Ilość węzłów to parametr definiujący do ilu sąsiadujących węzłów można wykonać ruch z danego punktu na mapie. Im większa liczba

węzłów sąsiadujących tym więcej węzłów do rozpatrzenia z każdą iteracją algorytmu. Wzrost liczby węzłów sąsiadujących powinien wpłynąć również pozytywnie na kształt wyznaczonych tras, umożliwiając jej płynniejsze skręcanie. Zwiększenie tego parametru może wpłynąć negatywnie na czas pracy algorytmu.

Mając na uwadze poprzednią iterację testów, bieżącą partię testów wykonano dla gęstości siatki 20m.

Tabela. 2. Tabela zawierająca ocenę trasy oraz czas wyznaczania trasy dla 5 przypadków testowych w zależności od liczby węzłów sąsiadujących

Przypadki testowe					
Liczba węzłów sąsiadujących	**1**	**2**	**3**	**4**	**5**
3	7 00:02:556	8 00:00:024	0 00:01:908	0 00:10:026	7 00:00:178
5	8 00:07:183	8 00:00:065	0 00:06:623	0 00:16:792	8 00:00:610
7	9 00:15:673	8 00:00:147	0 00:12:828	0 00:25:686	9 00:01:197
11	4 00:55:274	5 00:00:349	0 00:39:567	0 01:27:815	5 00:02:715
15	3 01:12:567	3 00:00:588	0 01:25:933	0 10:00:00	4 00:06:418
19	2 01:49:610	3 00:01:081	4 01:10:189	0 10:00:00	4 00:10:090
31	2 06:27:311	2 00:02:666	3 04:17:017	0 10:00:00	3 00:31:007

W wyniku przeprowadzonych eksperymentów zaobserwowano następujące właściwości.

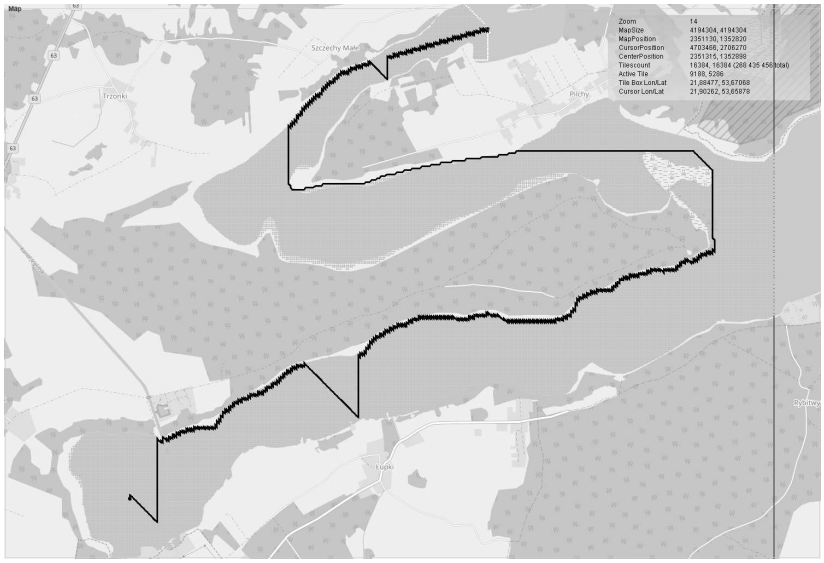
- Zgodnie z przewidywaniami zwiększenie ilości węzłów sąsiadujących wpływa niekorzystnie na czas pracy algorytmu.
- Zgodnie z przewidywaniami zwiększenie ilości węzłów sąsiadujących wpływa pozytywnie na ocenę. Trasy z większą ilością węzłów są mniej kanciaste przy czym różnica między wartościami 11-31 nie jest zauważalna.
- Trasa numer 4 nie została wyznaczona w żadnym testowanym przypadku.

- Iteracje dla trasy numer 4 z ilością węzłów ustawioną na 15,19,31 nie zostały ukończone w przeciągu 10 min i zostały przerwane przez użytkownika zgodnie z procedurą testową.
- Dla trasy numer 3 i wartości parametru 15 zaobserwowano, że część węzłów odwiedzonych przez algorytm znajdował się wewnątrz kanału. Jednak w połowie jego długości algorytm przerwał pracę.
- Dla trasy numer 3 zostały wyznaczone trasy dla wartości parametru 19 oraz 31. Jest to przypadek, w którym algorytm poradził sobie z przejściem przez wąski kanał.
- Porównując czas przebiegu pracy algorytmu dla trasy 3 dla wartości algorytmu 15 i 19 można zauważyć, że pomimo większej wartości parametru, czas pracy algorytmu zmalał. Stało się dlatego, że w przypadkach, gdzie wyznaczanie nie kończy się sukcesem algorytm musi przetworzyć wszystkie możliwe węzły. W przypadku sukcesu algorytm przerywa pracę i algorytm pracuje krócej.
- Dla przyjętych iteracji nie udało się wygenerować trasy, która byłaby zdolna nawigować w wąskim kanale – wymagają tego trasy 4 i 5.
- Mając na uwadze zebrane pomiary, najlepszą wartością ilości rozpatrywanych węzłów sąsiadujących jest liczba 7.

14 Składowa wiatrowa

Zgodnie z wymaganiami, proponowane rozwiązanie powinno mieć możliwość uwzględniania przy wyznaczaniu trasy obecnego kierunku wiatru. Aby sprostać temu wymaganiu, aplikację rozszerzono o możliwość wprowadzenia do obliczeń dodatkowej zmiennej w postaci wiatru.

W celu analizy wyznaczonych tras dla włączonej funkcjonalności wiatru zdecydowano się na przeprowadzenie iteracji testów z różną konfiguracją parametru NO_GO_ANGLE (kąta martwego) oraz różnych kierunków wiatru. Do zebrania danych użyto parametrów gęstości siatki 20m oraz liczbę rozpatrywanych węzłów na liczbę 3 oraz 7.



Rys. 9. Trasa ze składową wiatrową – sąsiednie węzły 3, kąt martwy 90 stopni

W wyniku przeprowadzonych eksperymentów zaobserwowano następujące właściwości.

- Ustawienie wiatru dokładnie przeciwlegle do kierunku trasy powoduje że trasa zostaje wyznaczona zygzakiem.
- Dla ustawienia kąta martwego 90 stopni zygzak jest bardzo gęsty i raczej nie nadaje się do podążania jachtem turystycznym.
- Zmniejszenie kąta martwego do 60 stopni spowodowało znacznie rzadsze zmiany kierunku trasy.
- W przypadku rozpatrywania 3 sąsiadujących węzłów, gdy kąt martwy ustawiono na 90 stopni, przebieg trasy jest bardzo kanciasty.
- Dodanie składowej wiatru nie powoduje zmiany w sposobie wyznaczania trasy dla kątów względem wiatru które nie zawierają się w kącie martwym.

15 Wnioski z przeprowadzonych badań i opracowanego projektu

- Wykonany projekt działa i jest zdolny do wyznaczania tras w zaproponowanym podejściu.
- Wyznaczane trasy są nieoptymalne ze względu na ich przebieg blisko lądów.
- W zależności od dobranych parametrów, czasy wyznaczania całodziennej trasy mogą być akceptowalne.
- Zastosowanie zaproponowanego podejścia w zastosowaniu komercyjnym wymaga dodatkowych optymalizacji oraz zastosowania

dodatkowych mechanizmów wspomagających pracę oryginalnego algorytmu.

- Kluczowym dla powodzenia wyznaczania trasy jest odpowiednie dobranie parametrów przetwarzania.
- Algorytm działa zgodnie z przewidywaniami w kontekście wykorzystania składowej wiatru.
- Wykorzystanie składowej wiatru ma bardzo duże znaczenie, ponieważ obecnie na rynku nie istnieją komercyjne zastosowania, które uwzględniają wiatr w swoich trasach.
- Kluczowym aspektem do poprawnego wyznaczania trasy jest odpowiednie przygotowanie implementacji wejściowej mapy do pracy algorytmu. W trakcie badań zostały wyznaczone trasy, które w realnym świecie nie mają możliwości przejścia, przykładowo ze względu na most ze zbyt małym prześwitem, który uniemożliwia przejścia. W przyjętym rozwiązaniu algorytm nigdy nie uwzględni takich danych. Chcąc je uwzględnić należy przygotować odpowiednią implementację, która wykluczy obszar pod mostem z rozważań jako potencjalny węzeł drogi.

16 Proponowane rozwiązania problemów z wyznaczonymi trasami oraz propozycja rozwiązań pozwalających na komercyjne zastosowanie

- Przyklejanie się tras do granic lądu i wody może zostać rozwiązane poprzez zmianę implementacji mapy przetwarzanej przez algorytm A*. Przykładowym rozwiązaniem może być przykładowo zablokowanie terenu kilka metrów od brzegu. Najprawdopodobniej wyznaczane trasy będą miały kształt zbliżony do pobliskiego lądu, ale będą wyglądały znacznie bardziej realistycznie nich dotychczas. Innym pomysłem może być użycie przy implementacji informacji o głębokości akwenu. Niestety takie podejście wymusza skorzystanie z map udostępniających informacje o głębokości. W obecnej implementacji przy użyciu map OpenStreetMap jest to niemożliwe. Takie podejście pozwoliłoby również na rozszerzenie możliwości mapy i przykładowo dodanie możliwości ograniczenia możliwych węzłów przejścia w zależności od ich głębokości oraz zanurzenia jachtu.
- Mając na uwadze obserwacje i wnioski wyciągnięte z przeprowadzonych testów i badań wynika, że kluczowym dla poprawnej pracy algorytmu jest ustawienie parametrów: liczby rozpatrywanych węzłów sąsiadujących oraz odległości między punktami. Zgodnie z badaniami wynika, że najlepsze trasy oraz akceptowalne czasy zostały wygenerowane dla parametrów gęstości siatki 20m oraz 7 rozpatrywanych węzłów przy użyciu heurystyki

dystansu do celu. W komercyjnym zastosowaniu proponowane byłoby zastosowanie dynamicznego dopasowywania tych parametrów w trakcie pracy algorytmu.

- Do komercyjnego zastosowania, sugerowane jest zastosowanie mechanizmu buforowania problematycznych tras. Tak przykładowo można by zrealizować wyznaczanie tras w kanałach, gdzie standardowy algorytm ze względu na zbyt małą liczbę węzłów nie może sobie poradzić. Problem ten może być również rozwiązany poprzez zastosowanie odpowiednio połączonych węzłów, tak by węzeł na początku kanału był na mapie algorytmu sąsiadującym z węzłem na końcu kanału.
- Do wyznaczania trasy w terenie z dużą ilości skomplikowanych przeszkód sugerowana jest implementacja takiej mapy algorytmu, która dynamicznie dostosowywała by swoją rozdzielczość i liczbę węzłów w zależności od trudności problemu. Przykładowo w wąskich kanałach, mapa posiadała by większą liczbę węzłów niż na środku jeziora.

17 Podsumowanie

Celem pracy było stworzenie rozwiązania informatycznego, które zdolne byłoby wyznaczać trasy na obszarach wodnych dla obiektów pływających. W założeniach stworzone rozwiązanie to miało być uzupełnieniem oferty komercyjnych produktów do nawigacji na wodzie. Ważnym elementem pomijanym przy tworzeniu dotychczasowych rozwiązań jest wiatr, dzięki któremu poruszają się niektóre środki transportu wodnego. Uwzględnienie tego czynnika może mieć duży wpływ na bezpieczeństwo w turystyce oraz możliwości poruszania się po nieznanych akwenach. Poprzez implementację programu, który umożliwia realizację celów niniejszej pracy udało się udowodnić, że takie rozwiązanie jest możliwe do zastosowania i wykorzystywania w codziennej żegludze. Do stworzenia aplikacji wykorzystano algorytm A*, który dzięki nowatorskiemu podejściu udało się wykorzystać w zupełnie nowym zastosowaniu. Przy wyznaczaniu tras program prawidłowo uwzględnia dodatkowy czynnik jakim jest kierunek wiatru.

Największą trudnością w trakcie prac nad projektem był odpowiedni dobór parametrów wejściowych gęstości siatki oraz liczby rozpatrywanych węzłów sąsiadujących. W wyniku przeprowadzonych testów ustalono optymalną wartość powyższych parametrów, dzięki czemu praca algorytmu jest efektywna.

Badana wersja aplikacji posiada jednak ograniczenia, przez które zastosowanie jej w praktyce nie jest możliwe. Wykorzystana mapa OpenStreetMap nie udostępnia niezbędnych informacji o głębokościach

i innych przeszkodach wodnych i z tego powodu wyznaczone trasy mogą być niebezpieczne lub nieżeglowne.

W celu osiągnięcia lepszych rezultatów należałoby zastosować proponowane w treści pracy rozwiązania.

Bibliografia

[1] http://pl.delphiayachts.eu/system/yacht_files/223/D31_test_wiatrowy_pl_original.jpg?1362779835
[2] https://sites.google.com/a/washingtonyachtclub.org/new-member-guide/intro-to-sailing/how-to-sail-upwind
[3] A. Kolaszewski, P. Świdwiński. Żeglarz i sternik jachtowy. Wydanie 20. 2011
[4] M. DeLoura. Best of Game Programming Gems. 2008
[5] P. Wróblewski. Algorytmy. Wydanie V. 2015
[6] B. Iwańczak, Quantum GIS Tworzenie i analiza map, 2013
[7] H. Schildt, Java. Kompendium programisty, Wydanie IX, 2016
[8] J. Żukowski, The Definitive Guide to Java Swing. Wydanie III. 2005
[9] S.Bałdyga “Implementacja algorytmu A-Star odnajdywania drogi w zastosowaniu śródlądowej nawigacji żeglarskiej” FTIMS PŁ 2017

USE OF A-STAR ALGORITHM IN THE DESIGN OF WATER VESSELS

Summary – The aim of this article is to find the solution which will allow sailors to create routes on the water. This thesis focuses on a practical aspect of the application of the algorithm for searching graph data structures in waterway navigation. Article concentrates on the issues connected with the program created. Next, crucial technical details were specified and the designed application was presented alongside its capabilites, functionality and possible limitations. In the last part, the tests of the program have been carried out along with the analysis and performance tunning. At the end of paper, there are the conclusions of the study conducted, the assessment of suitability for commercial use and possible ways to increase the efficiency of the process of path finding.

Keywords: inland navigation, A-star, sailing