PROCES PLANOWANIA PRZEMIESZCZANIA POJAZDÓW NIENORMATYWNYCH Z WYKORZYSTANIEM APLIKACJI WEBOWYCH

PLANNING OF MOVEMENT USING WEB APPLICATIONS FOR OVERWEIGHT/OVERSIZED VEHICLES

Arkadiusz JÓŹWIAK
arkadiusz.jozwiak@wat.edu.pl

Wojskowa Akademia Techniczna
Wydział Logistyki
Instytut Logistyki

Igor BETKIER
igor.betkier@gmail.com

Wojskowa Komenda Transportu Warszawa

STRESZCZENIE

W artykule przedstawiono możliwość optymalizacji procesu planowania tras przemieszczania pojazdów nienormatywnych na terenie odpowiedzialności Wojskowej Komendy Transportu Warszawa wykorzystując algorytm Dijkstry i programowanie w języku JavaScript.

SUMMARY

Abstract: Article contains posibilities to improve planning process of movement for overweight/oversized vehicles crosses over Warsaw Military Transportation Headquarter area using Dijkstra's algorithm and JavaScript programming language.

Słowa kluczowe: transport, przemieszczenie, programowanie, planowanie, javascript

Key words: transportation, movement, programming, planning, javascript

WSTĘP

Aktualnie funkcjonujący podsystem transportu i ruchu wojsk, bazujący na obszarach odpowiedzialności Wojskowych Komend Transportu, wymusza ich współdziałanie w zakresie planowania oraz uzgadniania projektów tras przejazdów przy pomocy Systemu Monitorowania Ruchu Wojsk - SI Konwój oraz dodatkowych modułów m.in. Pakiet Grafiki Operacyjnej. Dynamicznie rozwijająca się sytuacja drogowa wymusza ciągłą aktualizację danych gromadzonych przez komendy transportu w zakresie utrudnień drogowych i stanu obiektów inżynieryjnych. Brak wiedzy na temat infrastruktury drogowej w rejonie odpowiedzialności innej komendy transportu, generuje błędy, które skutecznie obniżają wydajność komendy w zakresie możliwości wydawania zezwoleń na przejazd drogowy. Ponadto brak elektronicznych baz danych w wojskowych komendach transportu oraz brak możliwości wyznaczania tras z punktu widzenia parametrów przy pomocy Google Maps, wydłuża proces walidacji trasy. Transport i proces jego planowania utożsamiany jest

z usługami transportowymi, a więc również z ich podstawowymi cechami m.in. złożoność, probabilizm i dynamiczność (Świderski, Jóźwiak, Jachimowski, 2018).

Współczesne technologie webowe, dają szerokie spektrum możliwości w zakresie zautomatyzowania najbardziej powtarzalnych procesów, z uwzględnieniem aktualizacji danych w czasie rzeczywistym. Podsystem transportu i ruchu wojsk, mimo że częściowo wykorzystuje systemy teleinformatyczne do planowania przemieszczania, nie posiada systemu integrującego dane Generalnej Dyrekcji Dróg Krajowych i Autostrad ze swoimi danymi.

Celem artykułu jest opracowanie modelu aplikacji optymalizującej planowanie przemieszczeń pojazdów nienormatywnych poprzez generowanie trasy uwzględniającej parametry pojazdu. Na potrzeby artykułu przyjęto również cele szczegółowe tj.: charakterystyka potrzeb związanych z planowaniem przemieszczania pojazdów nienormatywnych, określenie kluczowych procesów planowania wymagających wsparcia poprzez systemy IT, integracja działania Sekcji Przewozów i Służby Dyspozytorskiej oraz Sekcji Przeładunków i Sieci Transportowej. Zatem, przyjęto problem badawczy w formie pytania: czy możliwe jest stworzenie aplikacji optymalizującej planowanie przemieszczania pojazdów nienormatywnych przy wykorzystaniu technologii webowych?

W pracy wykorzystano następujące metody badawcze: analiza i synteza, dedukcja, modelowanie i wnioskowanie, techniki badawcze tj.: modelowanie opisowe oraz graficzne, modelowanie matematyczne i wnioskowanie przez porównanie, a także narzędzia badawcze: system monitorowania położenia wojsk SI Konwój, Pakiet Grafiki Operacyjnej, MS Excel, zintegrowane środowisko programistyczne Visual Studio Code, w którym napisano część kodu źródłowego w języku JavaScript (standard ECMAScript 2017) oraz silnik przeglądarki Google Chrome v. 67 do jego renderowania.

1. ZAŁOŻENIA TECHNOLOGICZNE I ARCHITEKTURA

Stworzenie aplikacji webowej pozwalającej generować trasy przemieszczania z punktu widzenia parametrów, dokonującej sprawdzenia jej pod kątem ograniczeń narzuconych przez GDDKiA oraz dającej możliwość gromadzenia i prezentacji danych wymaga przykładowego wachlarza technologicznego:

1. HTML (*Hypertext Markup Language)* – język znaczników, niezbędny do zbudowania struktury aplikacji działającej w przeglądarce internetowej. HTML umożliwia nadać semantykę informacjom zawartym w treści strony internetowej, a także daje możliwość osadzania w tekście dokumentu obiektów plikowych tj. multimedia,

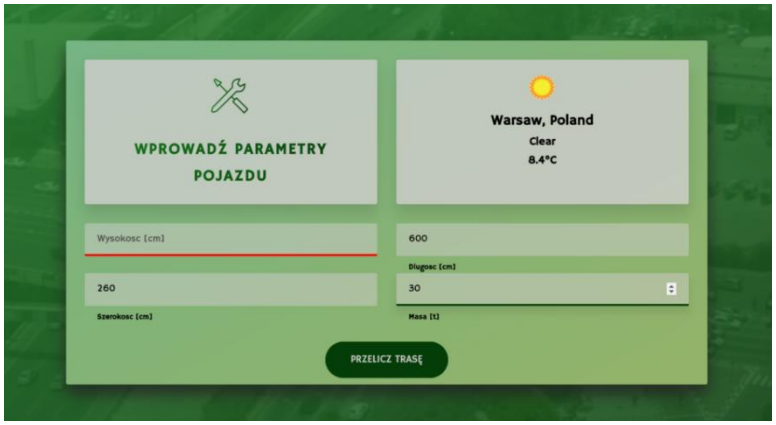
interaktywnych, do których należą formularze danych oraz odnośniki do innych treści znajdujących się w sieci. Aplikacja wykorzystuje HTML do logicznego podzielenia i nadania hierarchii elementom strony, osadzenia plików graficznych, stworzenia formularzy na podstawie których dane wpisane przez użytkownika, będą wykorzystywane przy wyznaczaniu optymalnej trasy, osadzenia skryptów niezbędnych do jej prawidłowego działania.

2. CSS (*Cascading Style Sheets*) – język służący do opisu formy prezentacji stron WWW, lista dyrektyw ustalających w jaki sposób ma zostać wyświetlana przez przeglądarkę internetową zawartość wybranego elementu HTML. CSS oferuje możliwość nadawania właściwości tj. kolor, margines, rodzina czcionek, odstęp międzywierszowy, pozycja elementu względem innych elementów. Z punktu widzenia tworzenia aplikacji webowej, CSS daje możliwość optymalizacji wyglądu interfejsu, tak aby użytkownik miał łatwość w korzystaniu z niej, rozmieszczenie elementów strony było intuicyjne, a wyświetlana treść bazująca na pobranych danych, jak najbardziej przystępna.
3. JavaScript – skryptowy, zorientowany obiektowo język programowania służący do zapewniania interakcji poprzez reagowanie na zdarzenia wywoływane przez użytkownika, walidacji danych wprowadzanych w formularzach i dalsze przetwarzanie tych danych, a także tworzenie złożonych efektów wizualnych (Atencio, 2017). Aplikacja wykorzystuje JavaScript do pobrania danych z sieci tj. aktualna pogoda, zdarzenia drogowe aktualizujące się na stronie GDDKiA, pobranie i przetwarzanie danych z lokalnej bazy danych, obsługi wszelkich zdarzeń, tworzenia animacji, a co najważniejsze – zaimplementowania algorytmu Dijkstry i sposobu wyznaczania trasy na bazie wielu warunków logicznych.

Zaproponowany model aplikacji webowej bazuje na podejściu modułowym, a więc odseparowaniu od siebie elementów wraz ze skryptami odnoszącymi się do nich. Ma to przełożenie na czytelność kodu źródłowego i mniejsze prawdopodobieństwo wystąpienia błędów. Aplikacja została podzielona na 4 podstawowe moduły:

1. Moduł informacyjny – moduł zawierający informacje na temat działania aplikacji, korzyści płynących z jej użytkowania, a także zawierający elementy graficzne, nowoczesny design i przycisk startowy.
2. Moduł formularza i wstępnej walidacji danych – moduł zawierający szereg pól, pobierających parametry pojazdu nienormatywnego, dokonujący wstępnej oceny

poprawności treści wpisanej przez użytkownika. Na rysunku 1 przedstawiono moduł formularza zbudowany w programie Visual Studio Code, wyświetlony w przeglądarce Google Chrome v. 67., gdzie walidacja danych odbywa się dzięki wyrażeniom regularnym. Ponadto moduł posiada dane pozyskane z API serwisu pogodowego *www.wunderground.com*, które wyświetla komunikaty w przypadku wystąpienia ekstremalnych zjawisk pogodowych.



Rys. 1. Moduł formularza i wstępnej walidacji danych w przeglądarce Google Chrome v. 67
Źródło: Opracowanie własne.

3. Moduł obsługi bazy danych – moduł pozwalający dodawać i usuwać obiekty inżynieryjne dla określonych odcinków drogowych do/z lokalnej bazy danych, dając możliwość aktualizacji informacji o infrastrukturze w rejonie odpowiedzialności np.: po wykonanym rekonesansie przez Sekcję Przeładunków i Sieci Transportowej.
4. Moduł mapy interaktywnej – moduł prezentujący optymalną trasę, pomiędzy dwoma miastami zaproponowanymi przez użytkownika z uwzględnieniem parametrów pojazdu. Moduł ma też możliwość wyświetlania kompletnych danych lokalnych, w celu oceny sytuacji transportowej w rejonie odpowiedzialności. Moduł umożliwia skalowanie mapy w formacie SVG, jak również zawiera zdarzenia *mouseover*, dające możliwość prezentacji elementów bazy danych w formie okna *popup* (Duckett, 2018). Na rysunku 2 przedstawiono graficznie elementy bazy danych.

Rys. 2. Moduł mapy interaktywnej w przeglądarce Google Chrome v. 67 – graficzne elementy bazy danych

Źródło: Opracowanie własne.

Moduł mapy interaktywnej zawiera punkty krzyżowania się dróg, a także odcinek drogowy pomiędzy punktem rozpoczęcia *Napierki*, a punktem zakończenia *Glinojeck* wprowadzonymi przez użytkownika. Użytkownik dokonuje wyboru punktu rozpoczęcia/zakończenia poprzez wprowadzenie nazwy miejscowości, bądź poprzez zdarzenie *click* wykonane nad elementem HTML stworzonym na podstawie bazy danych.

2. BUDOWA BAZY DANYCH

W celu zoptymalizowania formatu danych i łatwiejszej pracy z danymi poprzez użycie języka JavaScript, zastosowano format JSON do budowy obiektów przechowujących pary klucz – wartość. Zaproponowane rozwiązanie bazuje na obiektach dwóch typów:

1. Obiekt typu ROAD – obiekt przechowujący dane na temat odcinka drogowego pomiędzy dwoma kluczowymi punktami w sieci, odzwierciedlającymi miejscowości, w których dochodzi do krzyżowania się dróg różnego typu. Obiekt ten zawiera szereg informacji, istotnych z punktu widzenia przemieszczenia, dla których przyjęto następujące klucze:
   - name – nazwa drogi;
   - *id* – numer porządkowy;
   - $x1$, $x2$, $y1$, $y2$ – współrzędne prostej, będącej graficznym odzwierciedleniem drogi, w celu zobrazowania jej na interaktywnej mapie w formacie SVG;
   - number – numer drogi;
   - distance – rzeczywista długość odcinka drogi;
   - km – oznaczenie przedziału kilometrowego drogi;
   - cities – nazwa miejscowości granicznych;

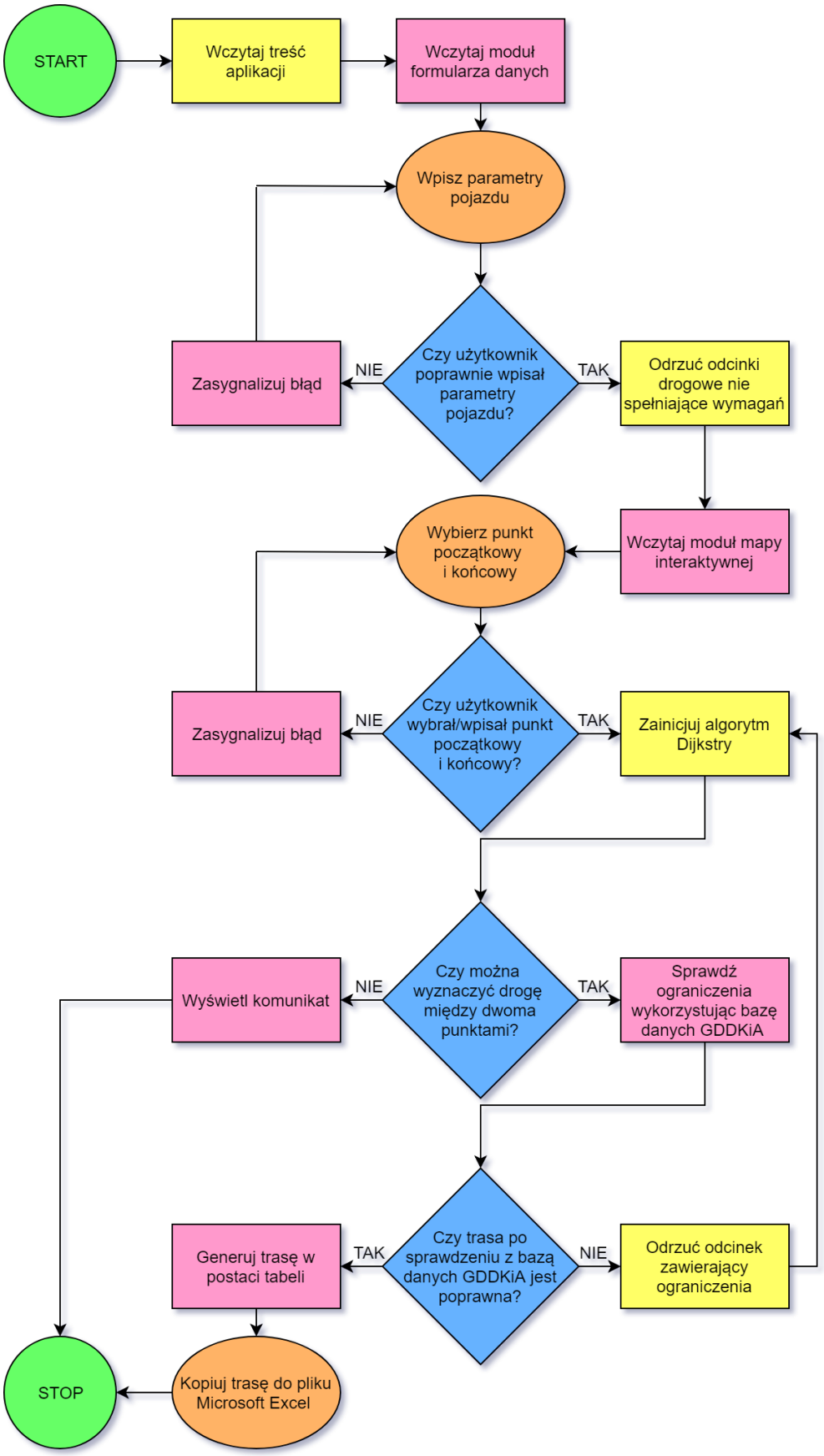
- maxSpeed – maksymalna prędkość, z jaką mogą przemieszczać się pojazdy określonym odcinkiem drogi;
- maxHeight, maxWidth, maxLength – maksymalne dopuszczalne parametry, jakie może posiadać pojazd przemieszczający się odcinkiem drogi;
- maxWeight – maksymalny tonaż/maksymalny nacisk na oś określony przez zarządcę drogi;
- objects – tablica zawierająca obiekty inżynieryjne na odcinku drogowym oraz informacje na ich temat tj. wysokość/nośność, lokalizacja, typ.

2. Obiekt typu CITY – obiekt przechowujący dane na temat kluczowych punktów w sieci, będących skrzyżowaniami dróg bądź miejscowościami granicznymi, zawierający informacje tj.:
   - *id* – numer porządkowy;
   - name – nazwa punktu;
   - *c*x, *cy* – współrzędne punktu, będącego graficznym odzwierciedleniem miejscowości, w celu zobrazowania jej na interaktywnej mapie w formacie SVG;
   - type – typ miejscowości np: border/default;
   - utm – współrzędne prostokątne płaskie punktu w systemie meldunkowym MGRS;
   - wgs84 – system odniesienia wykorzystujący parametry GRS80 oraz satelitarne obserwacje dopplerowskie i laserowe, które wykorzystywany jest zarówno w Pakiecie Grafiki Operacyjnej, jak i w module trasy w programie SI Konwój;
   - streets – tablica zawierająca numery dróg i nazwy ulic występujących na danej drodze w granicach administracyjnych miasta.

Zaproponowana budowa obiektów bazy danych umożliwia budowanie funkcji, które operują na danych, dając możliwości klasyfikacji dróg w zależności od rodzaju klucza, oraz grupowania tych dróg z punktu widzenia parametrów pojazdu, w celu wyznaczania trasy.

3. ALGORYTM DZIAŁANIA

Działanie aplikacji prowadzące do wyznaczenia trasy przejazdu pojazdu nienormatywnego można podzielić na kilka zasadniczych faz. Na rysunku 3 przedstawiono algorytm działania aplikacji wyznaczającej trasę przejazdu, z uwzględnieniem scenariusza braku możliwości wyznaczenia drogi przejazdu bądź wprowadzenia błędnych danych przez użytkownika.



Rys. 3. Algorytm działania aplikacji wyznaczającej trasę przejazdu
Źródło: Opracowanie własne.

Algorytm od punktu START do punktu STOP prowadzi przez działania prowadzące do wyznaczenia trasy przejazdu. Kolorem pomarańczowym oznaczono działanie użytkownika, które po załadowaniu treści strony wymusza wykonanie określonej części kodu źródłowego. Kolorem żółtym oznaczono działanie aplikacji skutkujące modyfikacją dotyczącą obiektów typu ROAD przechowywanych w pamięci przez aplikację. Kolorem różowym wyróżniono pozostałą część skryptu, najczęściej uaktualniającego strukturę HTML bądź wysyłającego zapytania do innych stron internetowych. Miejsce, w którym musi dojść do jednego z dwóch możliwych scenariuszy, mających różne następstwa, oznaczono kolorem niebieskim. W zależności od rodzaju scenariusza zostanie wykonana inna część kodu źródłowego.

W algorytmie wyróżnia się następujące fazy:

1. **Załadowanie treści aplikacji** – zainicjowanie działania aplikacji wymaga wpisania w oknie przeglądarki adresu serwera, na którym znajdują się pliki źródłowe. Po wysłaniu zapytania i uzyskaniu odpowiedzi od serwera, przeglądarka ładuje drzewo DOM, styluje elementy drzewa, ładuje skrypty oraz pobiera z bazy danych wszystkie obiekty typu ROAD i typu CITY zapisując je w tablicach.
2. **Wstępna walidacja danych** – użytkownik poprzez kliknięcie przycisku startowego inicjuje powstanie modułu formularza. Formularz składa się z 4 pól – szerokość, wysokość, długość oraz rzeczywista masa pojazdu. Skrypt dokonuje wstępnej walidacji parametrów, poprzez sygnalizację czerwonym/zielonym podkreśleniem, w zależności od poprawności danych, reagując na wartości wpisywane przez użytkownika. Dla poszczególnych pól formularza można przyjąć odpowiednie przedziały przyjmowanych wartości jak np.: 30 – 110 dla rzeczywistej masy pojazdu, z uwagi na brak ograniczeń na drogach krajowych w województwie mazowieckim dla pojazdów o masie poniżej 30 ton oraz masę zestawu niskopodwoziowego przewożącego czołg M1 Abrams 110 ton (Rychter, Sawicka i Puchała, 2016).
3. **Ostateczna walidacja danych** – użytkownik po kliknięciu przycisku PRZELICZ TRASĘ inicjuje usunięcie z tablicy przechowującej obiekty typu ROAD. Wszystkie te obiekty, które dla kluczy maxHeight, maxWidth, maxLength, maxWeight posiadają wartości mniejsze niż te wpisane przez użytkownika. Oznacza to, że pojazd przekracza maksymalne wartości przyjęte dla danego odcinka drogowego i nie może się tym odcinkiem przemieszczać. W dalszej kolejności aplikacja wysyła zapytanie do bazy danych GDDKiA udostępnionej na oficjalnej stronie i sprawdza, czy dla pozostałych odcinków drogowych nie pojawiło się zdarzenie drogowe, ograniczające możliwość

przejazdu. Ostatecznie aplikacja usuwa z tablicy odcinki drogowe nie mające połączeń z innymi, tworząc w ten sposób graf o nieujemnych wagach krawędzi, dla którego można zastosować algorytm Dijkstry.

4. **Algorytm Dijkstry** – w celu określenia źródła, a więc wyróżnionego wierzchołka w grafie, użytkownik wpisuje w specjalnym polu miasto rozpoczęcia bądź klika na nie na interaktywnej mapie. Następnie w ten sam sposób wybiera miasto docelowe, inicjując iterację po odcinkach drogowych, rozpoczynając od miasta rozpoczęcia, wykonując skrypt określający optymalną drogę oraz jej koszt dla każdego punktu w grafie (Jacyna, 2009). Algorytm zlicza optymalny koszt dotarcia do punktu oznaczonego jako miasto zakończenia oraz zapisuje w postaci tablicy odcinki drogowe pomiędzy dwoma miastami. Takie podejście daje możliwość wygenerowania zarówno bezpiecznej, jak i najkrótszej możliwej trasy. Początkowo użytkownik wprowadza dane (masę, szerokość, wysokość, długość) pojazdu nienormatywnego w formularzu. Akceptowanie wpisanych danych wejściowych inicjuje działanie funkcji iterującej po wszystkich obiektach drogowych, zapisując w tablicy tylko te odcinki, które spełniają określone wymagania (Sanjoy i Christos, 2010). Pseudokod wyżej wspomnianej funkcji, można przedstawić następująco:

$$\boldsymbol{for}(let\ i = 0; i \leq d; i^{++})$$

$$\boldsymbol{if}\begin{pmatrix} road_i.a < vehicle.a \vee road_i.b < vehicle.b \vee \\ road_i.c < vehicle.c \vee road_i.d\ <\ vehicle.d \end{pmatrix}$$

$$V.\boldsymbol{push}(road_i)$$

$$\boldsymbol{end\ if}$$

$$\boldsymbol{end\ for}$$

gdzie:

i – zmienna/licznik,

$++$ – inkrementacja,

d – liczba obiektów w zbiorze obiektów typu ROAD,

V – zbór obiektów ROAD spełniających zdefiniowany warunek,

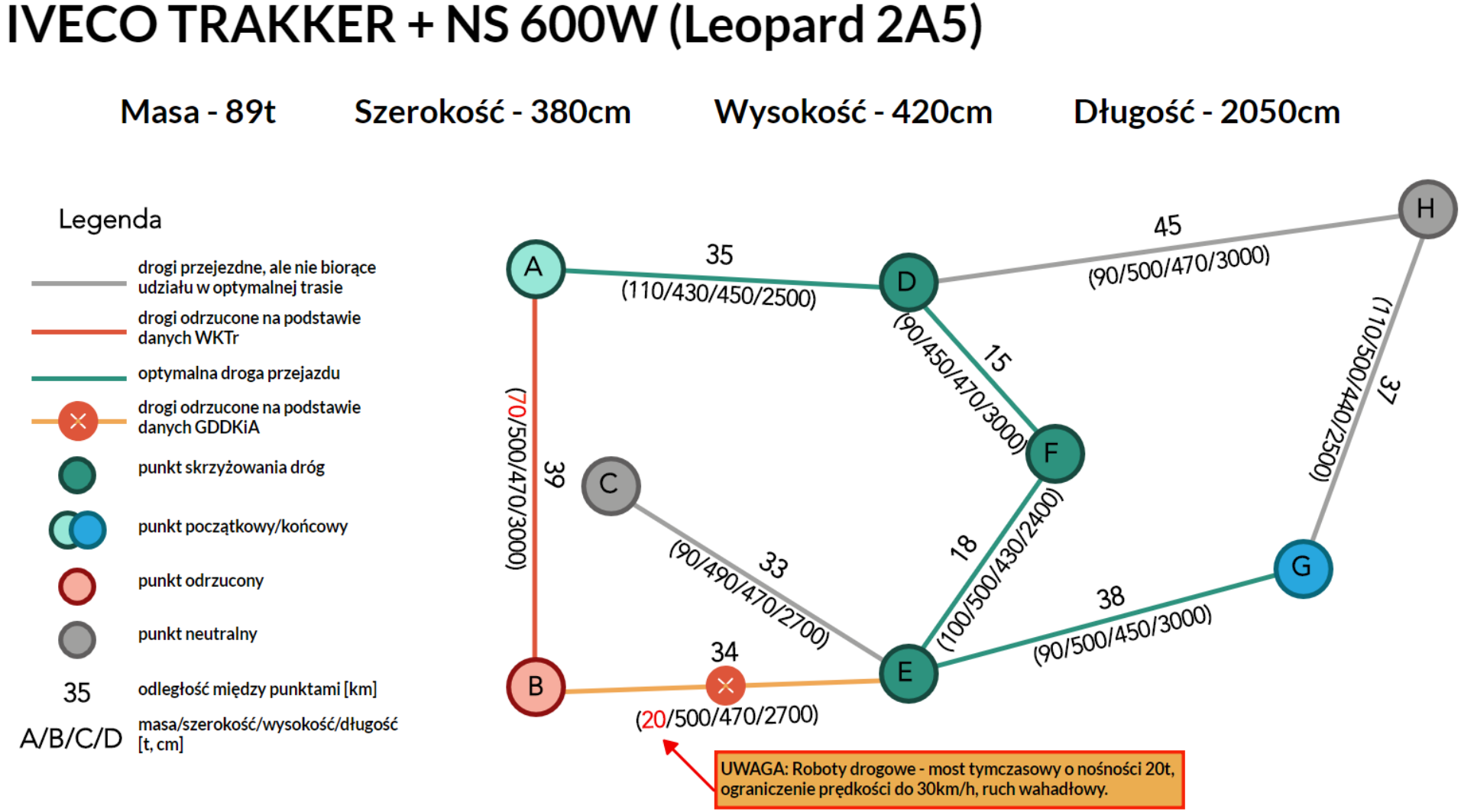
$road_i$ – pojedynczy obiekt typu ROAD,

$vehicle.x$ – określona właściwość pojazdu wprowadzona przez użytkownika,

a, b, c, d – właściwości maxLength/maxWidth/maxWeight/maxHight.

W tak zdefiniowanej funkcji pętla wykona się tyle razy, ile elementów zawiera zbiór obiektów typu ROAD, a obiekt nie zostanie dodany do tablicy zawierającej obiekty po walidacji, jeśli jakikolwiek parametr pojazdu wprowadzony przez użytkownika zostanie

przekroczony (w porównaniu do wartości zawartych w pojedynczych obiektach typu ROAD). Na rysunku 4 przedstawiono graficznie, w jaki sposób zaimplikowano algorytm Dijkstry dla zestawu niskopodwoziowego Iveco Trakker + NS 600W przewożącego czołg Leopard 2A5 w zaproponowanej aplikacji (Rudziński i Kowalczyk, 2012).



Rys. 4. Graficzne przedstawienie obiektów typu ROAD i typu CITY oraz logiki działania skryptu wyznaczającego trasę
Źródło: Opracowanie własne.

Według danych WKTr zarządca drogi AB dopuszcza przejazd tym odcinkiem tylko dla pojazdów o rzeczywistej masie całkowitej do 70 ton, dlatego ten odcinek drogowy został przez funkcję odrzucony. W dalszej kolejności zainicjowana zostaje funkcja wysyłająca zapytanie do serwera GDDKiA. Po odpowiedzi z serwera skrypt usuwa z tablicy odcinek drogowy pomiędzy punktem B i punktem E, zestawiając parametry odcinka z bazy danych WKTr z utrudnieniami z bazy danych GDDKiA, pozyskując je poprzez numer drogi i kilometr wystąpienia zdarzenia – remont mostu, obniżający nośność obiektu do 20 ton. Konsekwencją odrzucenia dróg AB oraz BE jest usunięcie z tablicy przechowującej punkty skrzyżowań dróg punktu B, ponieważ nie ma on połączenia z żadnym odcinkiem drogowym. Użytkownik oznaczając punkt początkowy i końcowy rozpoczyna iterację począwszy od punktu rozpoczęcia. Rezultatem takiego działania może być wyznaczenie optymalnej drogi przejazdu, bądź uzyskanie informacji, że nie istnieje połączenie między punktami dla przyjętych parametrów pojazdu.

Dla przykładowej trasy skrypt buduje tablicę, przypisując do każdego punktu (oznaczonego jako *V*) w grafie koszt dotarcia do tego punktu, co w tabeli 1 oznaczono jako *dv*, a także punkt poprzedni *pv*. Skrypt programu sprawdza, jakie drogi wychodzą z punktu A oraz do jakiego punktu prowadzą, po czym zapisuje jego nazwę oraz koszt dotarcia do tego punktu. Powtarzając procedurę z każdym punktem, pamiętając, że algorytm kontynuuje sprawdzanie dróg z punktu o najmniejszym koszcie dotarcia do niego, osiągnięty zostaje punkt G.

Tabela 1. Koszt dotarcia do określonego punktu w grafie od oznaczonego punktu A.

V	A	D	F	E	C	H	G
d_v	0	35	50	68	101	80	106
p_v	-	A	D	F	E	D	E

Źródło: Opracowanie własne.

Z rysunku 4 wynika, że możliwe było osiągnięcie punktu G z punktu H, ale koszt dotarcia do niego wyniósłby 117 km, dlatego algorytm nadpisał tą wartość kosztem 106 km oraz punktem E. Po zakończeniu iteracji algorytm wraca od punktu G do punktu A zapisując po drodze punkty, przez które prowadzi optymalna trasa, dając możliwość wygenerowania jej w formie tabeli, zawierającej istotne informacje.

5. **Generowanie trasy** – na podstawie odcinków drogowych zapisanych w tablicy w kolejności od miasta rozpoczęcia do miasta zakończenia, skrypt buduje w aplikacji

tabelę za pomocą znaczników HTML przyjmując konwencję znaną z SI Konwój, tak aby możliwe było przeniesienie trasy do pliku z rozszerzeniem wykorzystywanym w Microsoft Excel. Komórki tabeli wypełniane są wartościami zapisanymi dla określonych kluczy obiektu drogowego, ponadto komórka DODATKOWY OPIS wypełniana jest uwagami pobranymi ze strony GDDKiA odnoszącymi się do określonych odcinków drogowych. Resztę danych (m.in. miejsce załadunku/ wyładunku, postoju, noclegu) użytkownik wypełnia we własnym zakresie według zapotrzebowania na przejazd drogowy. Na rysunku 5 przedstawiono moduł trasy w programie SI Konwój zawierający zarówno elementy wygenerowane przez aplikację, jak i te wprowadzone przez użytkownika (Instrukcja operacyjnego wykorzystania teleinformatycznego systemu monitorowania położenia wojsk SI Konwój DU-4.4.4.2, 2015).

L.p.	LP (Powrót)	UTM (meldunkowe) - parametry trasy - Skopiowane z pliku zawierającego punkty łamanej wyeksportowanego z PGO						Parametry trasy - wypełniane ręcznie			Dług. odcinka [km]	Zadana pręd-kość [km/h]	V			Droga sum. do pkt [km]	Czas poko-nania odcinka [min]		
		dług. (WGS84)	szer. (WGS84)	Str efa	P as	Kwa drat	Odl.	Droga	Miejscowość	Zdarzenie Dodatkowy opis	S [km]	V [km/h]	Tam / Powr	ST	Czas Post. [min]	S [km]	T [min]	Data	Czas
1		8.112641583924	4.306968338241	34	U	FF	3745219734		BUDZISKO	GRANICA RP	0	50	>			0	0	19/06	21:00
2		2.933964060174	4.099431037388	34	U	FE	2646196314	8	SUWAŁKI	##############################	27	50	>			27	32,4	19/06	21:32
3		2.894655004944	4.048980913035	34	U	FE	2404190632	8/S61	WĘZEŁ SUWAŁKI POŁUDNIE		6,1	50	>			33,1	7,32	19/06	21:39
4		2.778102281763	3.977439599570	34	U	FE	1661182475	S61/8	WĘZEŁ RACZKI		11,5	50	>			44,6	13,8	19/06	21:53
5		2.960353355983	3.836212997283	34	U	FE	2899767081	8/16	AUGUSTÓW (obwodnica)		20,6	60	>			65,2	20,6	19/06	22:14
6		2.342118225215	3.843955134873	34	U	EE	8830166995	16	EŁK	OBWODNICA	45,6	50	>			110,8	54,72	19/06	23:08
7		1.976790888436	3.790057670446	34	U	EE	6434960606	16	WIERZBINY		27,3	50	>			138,1	32,76	19/06	23:41
8		2.055247097856	3.733672796582	34	U	EE	6961154408	dn	RON / BEMOWO PISKIE (BPTA)	Postój nocny	8,7	50	>	P	668	146,8	10,44	19/06	23:52
9		2.055247097856	3.733672796582	34	U	EE	6961154408		RON / BEMOWO PISKIE (BPTA)	Wznowienie przejazdu	0	50	>			146,8	0	20/06	11:00

Rys. 5. Moduł trasy w programie SI Konwój wraz z wygenerowaną trasą
Źródło: System monitorowania położenia wojsk SI Konwój.

Z rysunku wynika, iż istnieje skuteczna możliwość połączenia ze sobą wyżej wymienionych aplikacji.

4. PODSUMOWANIE

Przeprowadzona symulacja z wykorzystaniem zaproponowanej aplikacji webowej jest możliwa, co tym samym potwierdza rozwiązanie problemu badawczego, ale również i uzasadnia merytorycznie, ponieważ co najmniej kilkukrotnie optymalizuje czas opracowania trasy przejazdu. Ponadto rozwiązanie eliminuje błąd ze strony pracownika. Biorąc pod uwagę dane zgromadzone przez komendę transportu oraz aktualne dane o zdarzeniach drogowych,

których występowanie jest regularne i wymaga ciągłej aktualizacji wiedzy ze strony osoby odpowiedzialnej za wykonanie projektu trasy, metoda usprawniania wyznaczania drogi. Ponadto wygenerowana trasa zawiera uwagi odnośnie utrudnień w ruchu drogowym, takie jak ograniczenia skrajni poziomej, czy występowanie ruchu wahadłowego, uczulając kierowcę na zachowanie szczególnej ostrożności w wymienionych w trasie miejscach (Barcik, 2015). Wykorzystanie API serwisu pogodowego daje możliwość generowania uwag związanych z występowaniem zjawisk mających wpływ na bezpieczeństwo tj. możliwość wystąpienia podmuchów wiatru przekraczających 80-100km/h, wiążących się z potencjalnymi przechyłami bocznymi pojazdów nienormatywnych z punktu widzenia ich wysokości. Przyjęte podejście daje możliwość przeniesienia trasy do modułu trasy w programie SI Konwój, bez wprowadzania zmian. Dodatkowo wygenerowaną trasę można rozbudować poprzez Pakiet Grafiki Operacyjnej, uzupełniając trasę o dojazd do miejsca przeznaczenia drogami niższych klas. Z modyfikacjami zaproponowana aplikacja daje możliwość wykorzystania modułu mapy interaktywnej, do śledzenia zmian w sytuacji drogowej na terenie odpowiedzialności, poprzez ciągłą aktualizację danych w czasie rzeczywistym, z sygnalizacją czasu i miejsca powstania zdarzenia, wspomagając pracę służby dyspozytorskiej. Tym samym, cel artykułu został osiągnięty, a przyjęte rozwiązanie daje możliwości z zakresie optymalizacji bezpieczeństwa przejazdów przez rejon odpowiedzialności i wydajności w zakresie opracowywania projektów zezwoleń na przejazd drogowy, co ma kluczowy wpływ na skuteczność podsystemu transportu podczas ćwiczeń tj. Saber Strike (Zasady wojskowego ruchu drogowego DU-4.4.4(B), 2015) Również poprzez oszczędność czasu, stwarza możliwość skupienia uwagi na innych kluczowych obszarach związanych z transportem wojskowym (Prochowski i Żuchowski, 2009).

LITERATURA

Atencio, L. (2017). *Programowanie funkcyjne z JavaScript. Sposoby na lepszy kod.*, Helion: Gliwice.

Barcik, R. (2015). *Transport drogowy ładunków ponadgabarytowych*. *TTS Technika Transportu Szynowego R. 22, tom (12).*

Duckett, J. (2018). *JavaScript i jQuery. Interaktywne strony WWW dla każdego.,* Helion: Gliwice.

Jacyna, M. (2009). *Modelowanie i ocena systemów transportowych*. Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej.

MON Centrum Doktryn i Szkolenia SZ. (2015). *Instrukcja operacyjnego wykorzystania teleinformatycznego systemu monitorowania położenia wojsk SI Konwój DU-4.4.4.2*. Warszawa: MON Centrum Doktryn i Szkolenia SZ.

MON Centrum Doktryn i Szkolenia SZ. (2015). *Zasady wojskowego ruchu drogowego DU-4.4.4(B)*. Warszawa: MON Centrum Doktryn i Szkolenia SZ.

Prochowski, L., Żuchowski, A. (2009). *Technika transportu ładunków*. Warszawa: Wydawnictwo Komunikacji i Łączności.

Rudziński, R., Kowalczyk, M. (2012). Organizacja przewozu ładunków ponadnormatywnych transportem drogowym w Polsce. *Zeszyty Naukowe Uniwersytetu Przyrodniczo-Humanistycznego w Siedlcach, tom (22).*

Rychter, M., Sawicka, P., Puchała, A. (2016). Bezpieczeństwo w Transporcie Drogowym Ładunków Ponadgabarytowych w Polsce. *Autobusy: technika, eksploatacja, systemy transportowe R. 17, tom (12)*.

Sanjoy, D., Christos, P. (2010). *Algorytmy*. Warszawa: Wydawnictwo Naukowe PWN

Świderski, A., Jóźwiak, A, Jachimowski, R. (2018). Operational quality measures of vehicles applied for the transport services evaluation using artificial neural networks. *Eksploatacja i Niezawodnosc – Maintenance and Reliability, tom (20 (2):2920299).*