

**PROJEKT: SYMULACJA RUCHU DROGOWEGO NA  
PRZYKŁADZIE RONDA GRUNWALDZKIEGO W  
KRAKOWIE**

**KOD KURSU: 120-ISI-1S-777**  
**PRZEDMIOT: SYMULACJA SYSTEMÓW DYSKRETNYCH  
(SSD)**

Autorzy projektu

Łukasz Łabuz  
Dawid Małecki  
Mateusz Mazur

Opiekun projektu

Dr inż. MARCIN PIEKARCZYK



EAIIIB / Katedra Informatyki Stosowanej  
Akademia Górnictwo-Hutnicza im. Stanisława Staszica w  
Krakowie  
Kraków, Polska

25 stycznia 2024 r.

# Spis treści

<b>Abstract</b>	<b>iv</b>
<b>1 CEL I ZAKRES PROJEKTU</b>	<b>1</b>
1.1 Cel projektu . . . . .	1
1.2 Oczekiwane rezultaty . . . . .	1
<b>2 CHARAKTERYSTYKA PROBLEMU</b>	<b>3</b>
<b>3 DANE PORÓWNAWCZE</b>	<b>4</b>
<b>4 MODEL FORMALNY</b>	<b>8</b>
4.1 Model TSF . . . . .	8
4.1.1 Sieć drogowa w modelu TSF . . . . .	8
4.1.2 Sygnalizacja świetlna . . . . .	9
4.1.3 Pojazdy . . . . .	9
4.1.4 Piesi . . . . .	10
4.2 Automat komórkowy . . . . .	10
4.2.1 Zbiór komórek . . . . .	11
4.2.2 Funkcja otoczenia . . . . .	11
4.2.3 Zbiór stanów . . . . .	11
4.2.4 Funkcja stanu początkowego . . . . .	12
4.2.5 Funkcja przejścia . . . . .	12
<b>5 REALIZACJA PRAKTYCZNA</b>	<b>15</b>
5.1 Założenia projektowe . . . . .	15
5.2 Komponenty sprzętowe . . . . .	16
5.3 Oprogramowanie . . . . .	16
5.4 Szczegóły implementacji . . . . .	16
5.4.1 Klasa Simulator . . . . .	17
5.4.2 Klasa Plotter . . . . .	17
5.5 Czas trwania symulacji . . . . .	17
5.6 Optymalizacja działania . . . . .	18

5.7	Repozytorium kodu . . . . .	18
<b>6</b>	<b>REZULTATY SYMULACJI</b>	<b>19</b>
6.1	Scenariusz walidacyjny . . . . .	19
6.2	Scenariusz badawczy 1 . . . . .	20
6.3	Scenariusz badawczy 2 . . . . .	20
6.4	Scenariusz badawczy 3 . . . . .	23
<b>7</b>	<b>DYSKUSJA WYNIKÓW</b>	<b>25</b>
7.1	Omówienie wyników . . . . .	25
7.1.1	Scenariusz walidacyjny . . . . .	25
7.1.2	Scenariusze badawcze . . . . .	25
7.2	Wnioski i obserwacje . . . . .	26
<b>8</b>	<b>PODSUMOWANIE</b>	<b>27</b>
8.1	Napotkane problemy . . . . .	27
8.1.1	Określenie na którym pasie powinien ustawić się samochód przed skrzyżowaniem. . . . .	27
8.1.2	Rysowanie jezdni - odstęp między pasami oraz przesunięcie w przypadku ulicy dwukierunkowej. . . . .	27
8.1.3	Implementacja macierzy prawdopodobieństwa zmiany pozycji pieszych. . . . .	27
8.2	Kierunki rozwoju . . . . .	28
8.3	Słabe strony . . . . .	28
<b>References</b>		<b>29</b>

# **Abstract**

Celem niniejszego projektu było zaprojektowanie i implementacja symulatora ruchu ulicznego na Rondzie Grunwaldzkim w Krakowie. Uzyskane wyniki zostały poddane analizie i stwierdzono, że w zadowalający sposób odpowiadają one rzeczywistości. W pracy starano się odpowiedzieć na pytanie, jakie są przyczyny częstego powstania korków na tym węźle komunikacyjnym. Ustalono, że wynikają one z wadliwej organizacji ruchu na rondzie.

# **1. CEL I ZAKRES PROJEKTU**

## **1.1 Cel projektu**

Celem projektu jest stworzenie oprogramowania symulacyjnego do analizy modeli koordynacji ruchu drogowego z udziałem samochodów, pieszych oraz świadeł drogowych. Oprogramowanie ma generować dane, które będą mogły być wykorzystane do późniejszych analiz badanych modeli.

Cele szczegółowe:

1. Stworzenie kwerendy literaturowej, na bazie której stworzymy model formalny symulatora.
2. Przygotowanie modelu formalnego, który będzie stanowił podstawę do implementacji symulatora.
3. Implementacja prototypu symulatora, weryfikacja jego działania.
4. Finalizacja implementacji symulatora, weryfikacja jego działania.
5. Stworzenie modelu koordynacji ruchu krakowskiego Ronda Grunwaldzkiego i jego okolicy, w tym zebranie danych empirycznych określających jego cechy, walidacja wyników symulacji.
6. Przygotowanie przykładowej analizy czasu stania samochodów (w korku, przed światłami drogowymi) podczas ich przejazdu przez badane rondo.

## **1.2 Oczekiwane rezultaty**

Wobec finalnego rozwiązania oczekuje się, że będzie w stanie symulować w zadowalającym stopniu zgodności z rzeczywistością ruch uliczny na modelowanym obszarze.

Ponadto powinna być możliwa łatwa konfiguracja modelu, zarówno dotycząca organizacji ruchu drogowego, jak i liczby uczestników ruchu w celu udostępniania możliwości symulacji różnych dni tygodnia oraz pór dnia.

## **2. CHARAKTERYSTYKA PROBLEMU**

Współcześnie najskuteczniejsze modele symulacji ruchu drogowego [1] opierają się na teorii automatów komórkowych i modelu Nagela-Schreckenberga [2]. Model ten służy on do symulacji ruchu pojazdów na prostym odcinku drogi.

Jego istotnym rozszerzeniem jest model TSF [1]. Umożliwia on przeprowadzanie symulacji ruchu na rzeczywistych sieciach drogowych i rzeczywistych danych. Uwzględnia m.in. takie elementy jak:

- skrzyżowania,
- rozróżnialność kierowców,
- rozróżnialność typów dróg,
- światła drogowe,
- wielopasmowość dróg.

Model TSF jest modelem mikroskopowym, co oznacza, że symuluje on ruch drogowy na poziomie indywidualnych pojazdów. W modelu tym każdy pojazd jest reprezentowany przez pojedynczą komórkę.

Podejście do ruchu pieszych jest współcześnie bardzo zróżnicowane [3]. Najczęściej stosowane okazuje się podejście również oparte na automatach komórkowych w skali mikroskopowej.

### **3. DANE PORÓWNAWCZE**

Organizację ruchu drogowego w symulacji charakteryzują następujące parametry:

- częstość pojawiania się samochodów na granicach modelowanego układu.
- częstość pojawiania się pieszych na granicach modelowanego układu.
- lokalizacja skrzyżowań
- długość, liczba pasów oraz prędkość maksymalna ulic
- czasy trwania stanów świateł drogowych oraz ich lokalizacja

Wartości służące do walidacji modelu z warunkami świata rzeczywistego, miały w naszym przypadku dwa źródła.

Dane w dużej mierze niezmienne w czasie, jak organizacja ruchu, zostały ustalone na podstawie map z OpenStreetMap [4] oraz obserwacji w terenie.

Pozostałe dane, na moment pisania pracy, nie są nigdzie udostępnione. Zostały one zatem zebrane przez członków zespołu z terenu. Za referencyjny okres przyjęto przedpołudnie dnia roboczego. Zebrane informacje miały zarówno charakter ilościowy (liczba samochodów i pieszych uczestniczących w ruchu ulicznym, czasy trwania stanów świateł drogowych) oraz jakościowy (ocena, czy w danej chwili na którejś ulicy występuje korek uliczny). Zebrane wartości prezentują tabele 3.1, 3.2 oraz 3.3. Dla ułatwienia rysunek 3.1 prezentuje mapę lokalizacji świateł drogowych.

Tabela 3.1: Częstość pojawiania się samochodów na granicach modelowanego układu zebrane metodą obserwacji w terenie.

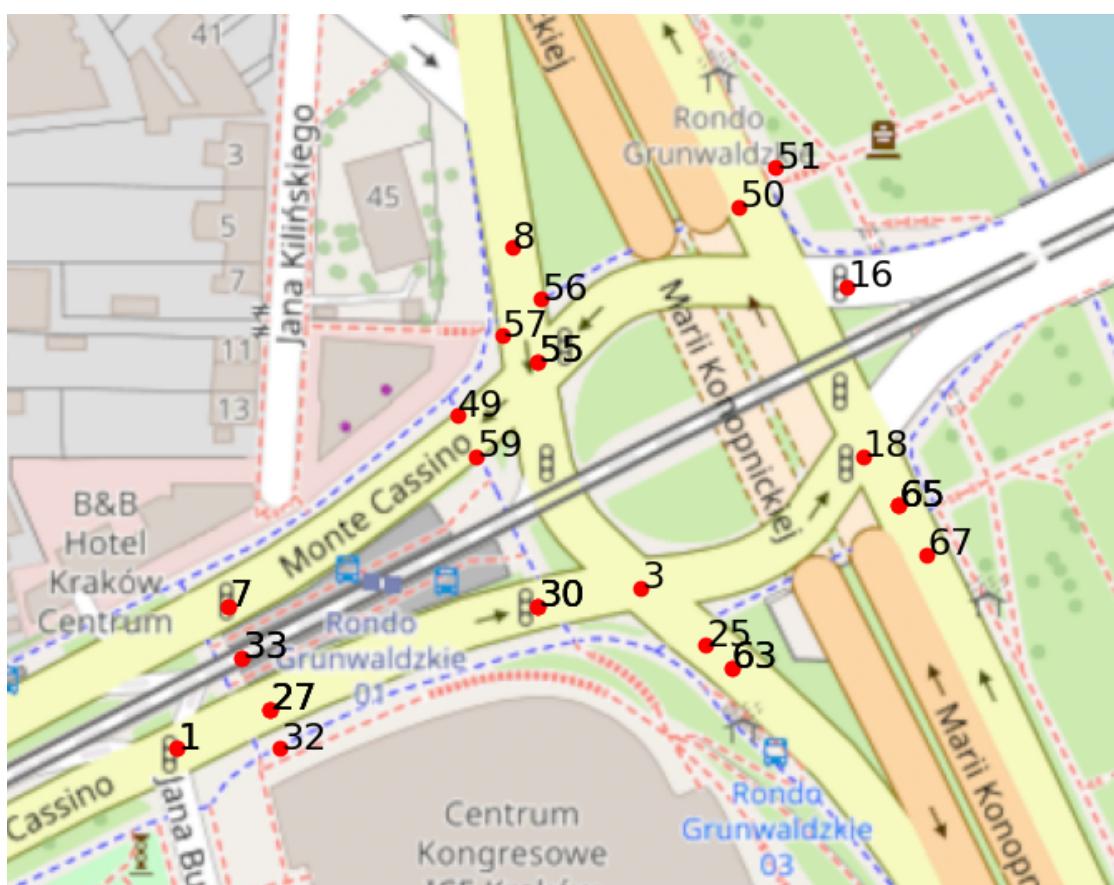
Lokalizacja granicy modelu	$f [1/min]$
Wjazd z ul. Skwerowej na ul. Monte Cassino	0,067
Ul. Monte Cassino na wysokości budynku nr 15	10,2
Wjazd z ul. Jana Bułhaka na ul. Monte Cassino	1,3
Ul. Marii Konopnickiej na wysokości budynku nr 19	28,53
Wjazd na Most Grunwaldzki	12,2
Ul. Marii Konopnickiej na wysokości budynku nr 16	32,3
Wjazd z ul. Barskiej na ul. Marii Konopnickiej	3,467

Tabela 3.2: Częstość pojawiania się pieszych na granicach modelowanego układu zebrane metodą obserwacji w terenie.

Lokalizacja	$f [1/min]$
Wyjście z ul. Jana Bułhaka na ul. Monte Cassino	3,1
Przystanek Rondo Grunwaldzkie 03	1,2
Przystanek Rondo Grunwaldzkie 01	3,0
Wschodni chodnik przy ul. M. Konopnickiej na wysokości bud. nr 19	1,4
Zachodni chodnik przy ul. M. Konopnickiej na wysokości bud. nr 19	1,3
Wejście z ul. Konfederackiej na ul. Monte Cassino	0,2
Wejście z ul. Edmunda Wasilewskiego na ul. Monte Cassino	0,3
Wschodnie wejście na południowy chodnik na Moście Grunwaldzkim	2,4
Wschodnie wejście na północny chodnik na Moście Grunwaldzkim	2,9

Tabela 3.3: Czasy trwania stanów świateł drogowych w modelowanym układzie zebra-  
ne metodą obserwacji w terenie.

Lokalizacja	$t_{zielone}$ [s]	$t_{czterwone}$ [s]
1	32	78
1	38	13
27	114	30
7	29	57
8	109	48
55	60	50
14	38	52
14	52	38
16	63	31
65	52	28
18	70	75
67	107	39
30	107	39
27	13	38
27	13	38
33	15	15
33	15	15
7	57	29
30	39	107
30	39	107
32	18	18
59	50	60
49	50	60
51	28	52
56	48	109
57	48	109
50	28	52
65	39	107
65	39	107
3	39	107
63	107	39
63	107	39
25	39	107
55	50	60



Rysunek 3.1: Lokalizacja świateł drogowych naniesiona na mapę z [4].

## 4. MODEL FORMALNY

### 4.1 Model TSF

Współcześnie najskuteczniejsze modele symulacji ruchu drogowego [1] opierają się na teorii automatów komórkowych i modelu Nagela-Schreckenberga. Nasz model bazuje na modelu TSF.

#### 4.1.1 Sieć drogowa w modelu TSF

Sieć drogowa w modelu TSF reprezentowana jest jako graf skierowany  $G = (V, E)$ , gdzie  $V$  to zbiór wierzchołków, a  $E \subseteq V \times V$  to zbiór krawędzi. Krawędzie reprezentują odcinki dróg, posiadają określone atrybuty:

- $d$  - długość,
- $lanes$  - liczba pasów ruchu,
- $v_{avg}$  - średnią maksymalną prędkość jazdy,
- $v_{std}$  odchylenie standardowe wartości  $v_{avg}$ ,
- $type$  - typ drogi.

W obrębie danej krawędzi pasy ruchu są numerowane liczbami naturalnymi. W obrębie pojedynczego pasa, komórki również są numerowane. Komórkę jednoznacznie wyznacza więc trójka: *edge*, *lane*, *cell*. Wszystkie komórki na tej samej krawędzi mają jednakowość długość i wszystkie pasy ruchu na tej krawędzi są podzielone na tyle samo komórek o takiej samej długości.

### 4.1.2 Sygnalizacja świetlna

W modelu TSF sygnalizacja świetlna jest zlokalizowana w wierzchołkach grafu. Każdą sygnalizację charakteryzują następujące parametry:

- $id$  - unikalny identyfikator,
- $position$  - lokalizacja (krawędź, na końcu której znajduje się sygnalizacja),
- $t_{green}$  - czas trwania fazy zielonej,
- $t_{red}$  - czas trwania fazy czerwonej,
- $t_{change}$  - czas trwania zmiany fazy,
- $state$  - aktualny stan (*GREEN/RED*).

Wartość  $t_{change}$  jest nieujemna i zmniejsza się o 1 w każdym kroku symulacji aż do osiągnięcia wartości 0. Wtedy następuje zmiana stanu sygnalizacji na przeciwny i  $t_{change}$  przyjmuje wartość  $t_{green}$  lub  $t_{red}$  w zależności od stanu, do którego zmieniła sygnalizacja.

Zbiór wszystkich sygnalizacji w modelu oznaczamy jako *SIGNALS*.

### 4.1.3 Pojazdy

W modelu TSF pojazdy mają następujące atrybuty:

- $id$  - unikalny identyfikator,
- $edge$  - krawędź, na której znajduje się pojazd,
- $distance$  - odległość od początku krawędzi,
- $id_{lane}$  - pas ruchu, na którym znajduje się pojazd,
- $id_{cell}$  - komórka, w której znajduje się pojazd,
- $profile$  - parametry ruchu pojazdu (wartość z przedziału  $[0, 1]$ ),
- $velocity$  - aktualna prędkość pojazdu,
- $path$  - ścieżka, którą pokonuje pojazd,
- $start$  - czas rozpoczęcia symulacji pojazdu.

Zbiór wszystkich pojazdów w modelu oznaczamy jako *CARS*.

#### 4.1.4 Piesi

Dodani przez nas do modelu piesi mają następujące atrybuty:

- $id$  - unikalny identyfikator,
- $edge$  - krawędź, na której znajduje się pieszy,
- $distance$  - odległość od początku krawędzi,
- $id_{cell}$  - komórka, w której znajduje się pieszy,
- $r_1$  - prawdopodobieństwo, że pieszy przestrzega prawa,
- $r_2$  - prawdopodobieństwo, że pieszy zatrzyma się na czerwonym świetle,
- $t_{walk}$  - czas, przez który pieszy uzna za bezpieczny, aby przejść na drugą stronę ulicy,
- $velocity$  - aktualna prędkość pieszego,
- $path$  - ścieżka, którą pokonuje pieszy,
- $start$  - czas rozpoczęcia symulacji pieszego.

Zbiór wszystkich pieszych w modelu oznaczamy jako  $PEDESTRIANS$ .

## 4.2 Automat komórkowy

Automat komórkowy w naszym modelu to krotka:

$$CA = \langle T, C, N, S, S_0, F \rangle \quad (4.1)$$

gdzie:

- $T$  - przedział czasu, w którym odbywa się ewolucja automatu,
- $C$  - zbiór komórek,
- $N : C \rightarrow 2^C$  - funkcja, która każdej komórce przyporządkowuje jej otoczenie,
- $S$  - zbiór możliwych stanów komórek,

- $S_0 : C \rightarrow S$  - funkcja, która każdej komórce przyporządkowuje jej stan początkowy,
- $F : T \times C \rightarrow S$  - reguła przejścia - funkcja, która każdej komórce przyporządkowuje jej stan po ewolucji.

### 4.2.1 Zbiór komórek

W naszym modelu komórki są reprezentowane przez pojazdy, pieszych oraz sygnalizacje świetlne. Dla ułatwienia, zastosujemy oznaczenie 4.2.

$$CELLS = CARS \cup PEDESTRIANS \quad (4.2)$$

Wtedy:

$$C = CELLS \cup SIGNALS \quad (4.3)$$

### 4.2.2 Funkcja otoczenia

O ruchu pojazdów oraz pieszych decyduje przede wszystkim ich otoczenia, ale również mogą to być sygnały globalne, pochodzące np. z nawigacji.

Stan sygnalizacji w chwili  $t + 1$  zależy nie tylko od jej stanu w chwili  $t$ , ale również od stanu innych sygnalizacji w chwili  $t$ , ponieważ sygnalizacja może być ustawiane globalnie, w celu optymalizacji ruchu.

Zatem, w ogólnym przypadku:

$$\forall_{c \in C} N(c) = C \quad (4.4)$$

### 4.2.3 Zbiór stanów

Komórki znajdujące się na krawędziach grafu mogą być zajmowane przez pojazdy, pieszych, lub być puste. Jeżeli w komórce znajduje się pojazd lub pieszy, to jest on jej stanem. Jeżeli komórka jest pusta, to jest stanem *null*. W przypadku sygnalizacji, stanem komórki jest fasą sygnalizacji (*GREEN/RED*). Zatem:

---

**Algorithm 1** Algorytm przejścia dla sygnalizacji świetlnej  $s$  w kroku  $t$

---

**Require:**  $s \in SIGNALS(G), t \in T$

**Ensure:**  $state(s, t + 1) \in \{GREEN, RED\}$

```

if  $t_{change}(s) > 0$  then
     $t_{change}(s) := t_{change}(s) - 1$ 
     $state(s, t + 1) := state(s, t)$ 
    return  $state(s, t + 1)$ 
else
    if  $state(s, t) = RED$  then
         $t_{change}(s) := t_{green}(s)$ 
         $state(s, t + 1) := GREEN$ 
    else
         $t_{change}(s) := t_{red}(s)$ 
         $state(s, t + 1) := RED$ 
    end if
    return  $state(s, t + 1)$ 
end if

```

---

Rysunek 4.1: Algorytm opisujący funkcję przejścia dla komórek ze zbioru  $SIGNALS$ .  
Źródło: [1]

$$S = \{null\} \cup CARS \cup PEDESTRIANS \cup \{GREEN, RED\} \quad (4.5)$$

#### 4.2.4 Funkcja stanu początkowego

Stan początkowy komórki jest zależny od tego, czy jest to sygnalizacja, czy komórka na krawędzi. W przypadku sygnalizacji, stan początkowy jest dowolnym elementem ze zbioru  $\{GREEN, RED\}$ . Konfiguracja początkowa komórek ze zbioru CELLS to rozmieszczenie pojazdów i pieszych, którzy rozpoczynają ruch wraz z początkiem symulacji na odpowiednich komórkach CELLS.

#### 4.2.5 Funkcja przejścia

Funkcja przejścia jest najbardziej złożonym elementem automatu komórkowego.

Dla komórek ze zbioru SIGNALS reguły przejścia definiuje algorytm przedstawiony na rysunku 4.1.

Dla komórek ze zbioru CARS reguły przejścia definiuje algorytm przedstawiony na rysunku 4.2. W tej funkcji wykorzystywane są następujące parametry pomocnicze:

---

**Algorithm 2** Algorytm ruchu pojazdu *car* w kroku *t*

---

```

Require:  $G = (V, E)$ ,  $t \in T$ ,  $car \in CARS(t)$ ,  $turnPenalty$ ,  $crossroadPenalty$ ,  $prob$ 
     $increaseVelocity(car, t)$ ;
    if  $stopOnSignal(car, t)$  then
         $reduceVelocityOnSignal(car, t)$ 
    else
        if  $turnOnCrossroad(car, t)$  then
             $reduceVelocity(car, t, turnParameter)$ ;
        else
            if  $crossroad(car, t)$  then
                 $reduceVelocity(car, t, crossroadParameter)$ ;
            end if
        end if
    end if
    if  $shouldChangeLane(car, t)$  then
         $changeLane(car, t)$ ;
    end if
     $safeReduceVelocity(car, t)$ 
    with probability  $prob$ :  $reduceVelocity(car, t)$ ;
     $makeMove(car, t)$ ;

```

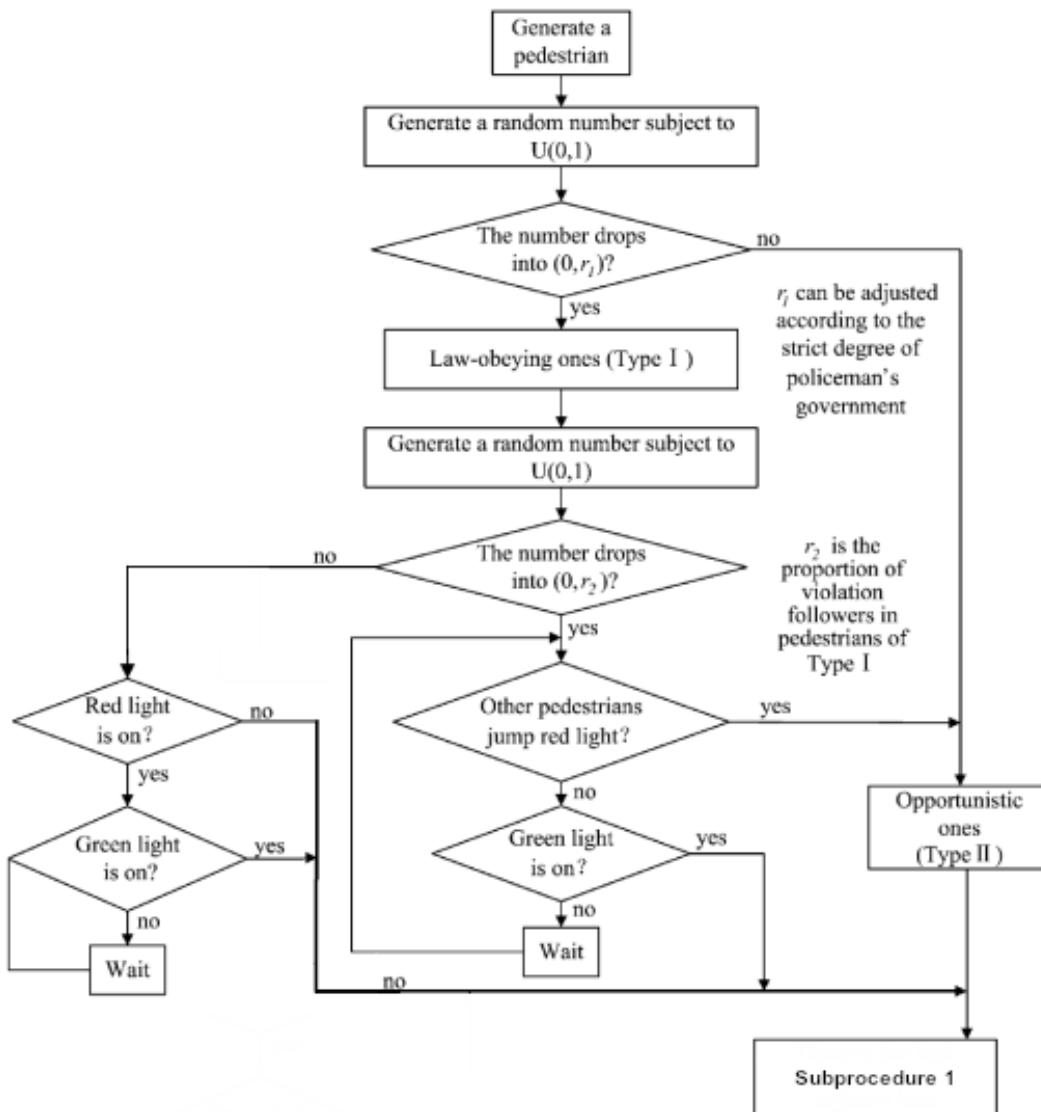
---

Rysunek 4.2: Algorytm opisujący funkcję przejścia dla komórek ze zbioru *CARS*. Źródło: [1]

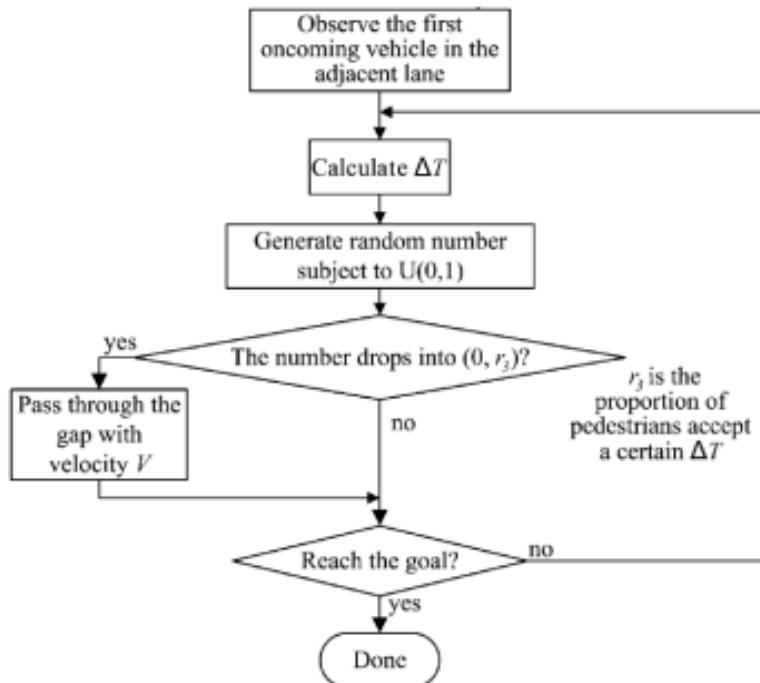
- $turnPenalty \in [0, 1]$  - zachowanie pojazdu przed skrętem,
- $crossroadPenalty \in [0, 1]$  - zachowanie pojazdu w trakcie przejazdu przez skrzyżowanie (bez skręcania),
- $prob \in [0, 1]$  - losowa redukcja prędkości pojazdu.

Dla komórek ze zbioru PEDESTRIANS reguły przejścia definiuje głównie macierz prawdopodobieństw  $P$  oraz, w przypadku przechodzenia przez przejście dla pieszych, fragment algorytmu przedstawionego na rysunkach 4.3, 4.4.

$P$  jest macierzą o wymiarach  $3 \times 3$ , która tworzona jest na bazie najbliższego otoczenia pieszego. Pieszy może zmienić pozycję na jedną z 8 sąsiednich komórek. W zależności od tego, czy pieszy chce skręcić, wyprzedzić innego pieszego, czy dalej kontynuować ruch prosto, prawdopodobieństwo zajęcia danej komórki jest różne.



Rysunek 4.3: Algorytm opisujący część funkcji przejścia dla komórek ze zbioru *PEDESTRIANS*. Źródło: [3]



Rysunek 4.4: Algorytm opisujący część funkcji przejścia dla komórek ze zbioru PEDESTRIANS - podproces 1. Źródło: [3]

## 5. REALIZACJA PRAKTYCZNA

### 5.1 Założenia projektowe

Aby symulacja działała poprawnie, użytkownik musi zapewnić poprawny model opisujący modelowany fragment infrastruktury drogowej. Model taki może zawierać, opis infrastruktury (skrzyżowania, drogi, światła), definicje samochodów i pieszych oraz ich generatory. Brak jest możliwości definicji znaków drogowych, czy opisu typów pojazdów uprzywilejowanych (np. komunikacji miejskiej czy służb specjalnych).

## 5.2 Komponenty sprzętowe

Symulacja uruchamiana była na laptopie z następującymi podzespołami:

- CPU: AMD® Ryzen 7 5700u with radeon graphics × 16
- RAM: 16 GB
- GPU: zintegrowana
- OS: Ubuntu 22.04.3 LTS 64-bit

W trakcie działania symulacji CPU było wykorzystane w około 10%, a RAM w około 60%. Każe to sądzić, że w przypadku uruchamiania symulacji na innym sprzęcie, procesor może być trochę mniej wydajny, lecz pamięci RAM powinno być nie mniej niż 16 GB.

Warto zwrócić uwagę, że dla symulacji większych obszarów wartości te, szczególnie zapotrzebowanie na pamięć RAM, mogą znacząco wzrosnąć.

## 5.3 Oprogramowanie

Oprogramowanie zostało napisane w języku Python [5] w wersji 3.9.18 z wykorzystaniem następujących bibliotek:

- NumPy (v. 1.26.2) [6]
- NetworkX (v. 3.1) [7]
- Pygame (v. 2.5.2) [8]
- Pandas (v. 2.1.4) [9]
- Matplotlib (v. 3.8.0) [10]

## 5.4 Szczegóły implementacji

Implementacja projektu oparta jest na dwóch głównych klasach: `Simulator` oraz `Plotter`. Projekt wersjonowany był przy użyciu GitHub'a. Do uruchomienia wymagane są biblioteki wymienione w pliku `requirements.txt` załączonym do repozytorium. Oprogramowanie uruchamia się polecением `python src/main.py`.

### 5.4.1 Klasa Simulator

`Simulator` jest klasą reprezentującą symulator ruchu drogowego. Konfigurację z pliku JSON modelu załadować można za pomocą metody `load()`. Do uruchamiania symulacji służy metoda `step()`, której podać można argumenty `steps` (liczba kroków, które należy wykonać) oraz `t_gap` (minimalny odstęp czasowy pomiędzy krokami).

Metoda `step()` wywołuje zadaną liczbę razy metodę pomocniczą `_step()` odczekując daną ilość czasu między wywołaniami. Metoda `_step()` wywołuje kolejne metody pomocnicze odpowiedzialne za aktualizację niepustych komórek symulatora: samochodów, pieszych, świateł i generatorów.

Klasa `Simulator` oferuje również szereg metod zwracających wartości swoich pól (m.in. krok czasowy, obecny krok, krok maksymalny, czas trwania symulacji) oraz metody zwracające następujące obiekty typu `Pandas.DataFrame`:

- zbiór skrzyżowań symulacji (statyczna)
- zbiór dróg symulacji (statyczna)
- zbiór informacji o samochodach (dynamiczna, aktualizowana po każdym kroku)
- zbiór informacji o pieszych (dynamiczna, aktualizowana po każdym kroku)

### 5.4.2 Klasa Plotter

`Plotter` jest klasą reprezentującą interaktywny interfejs graficzny symulatora. Pozwala on na obrazowanie stanu wewnętrznego komórek symulatora w czasie rzeczywistym, wyświetlanie obrazu mapy w tle, a także manipulację widokiem (przybliżanie, przesuwanie). Dla ułatwienia tworzenia i weryfikacji modelu wsadowego, interfejs umożliwia wyświetlanie id elementów składowych oraz wyświetla pozycję kurSORA względem całej mapy.

## 5.5 Czas trwania symulacji

Argument `t_gap` metody `stop()` określa minimalny odstęp czasowy pomiędzy krokami. Domyślnie przyjmuje wartość 0 i w takim przypadku symulacja wykonywana jest tak szybko, jak jest to możliwe.

Czas symulacji 16 min 40 s ruchu (1000 jednosekundowych kroków) jest równy 278.6 s. Oznacza to, że czas potrzebny na symulację jednego kroku (a więc symulację kroku wszystkich samochodów, pieszych oraz świateł) wynosi **średnio 0.28 s**. Jest to zadowalający wynik, który sprawia, że korzystanie z symulatora nie musi być czasochłonne.

Należy jednak zwrócić uwagę na fakt, że z uwagi na dużą złożoność procesu oraz brak możliwości większej optymalizacji, wyświetlanie stanu symulacji nie jest w stanie dorównać takiej prędkości. Powoduje to przeskoki w wizualizacji po kilka-kilkanaście kroków. Opóźnienie zwiększa się znacznie, gdy wyświetlany jest dodatkowo cały graf, na którym operuje klasa symulatora.

## 5.6 Optymalizacja działania

Aby zagwarantować niskie czasy trwania obliczeń kolejnych kroków klasa symulatora została poddana procesowi optymalizacji. W głównej mierze polegała ona na przejściu z obliczeń w pętlach na działania macierzowe. Zdecydowaliśmy się również na wydzielenie przypadków brzegowych z algorytmów. Sprawdzanie, czy dane warunki zachodzą, i finalne stosowanie prostych formuł ogólnych okazało się być znacznie mniej wymagające obliczeniowo. Zastosowanie dużej ilości małych zmian przyniosło, w skali globalnej, znaczną poprawę.

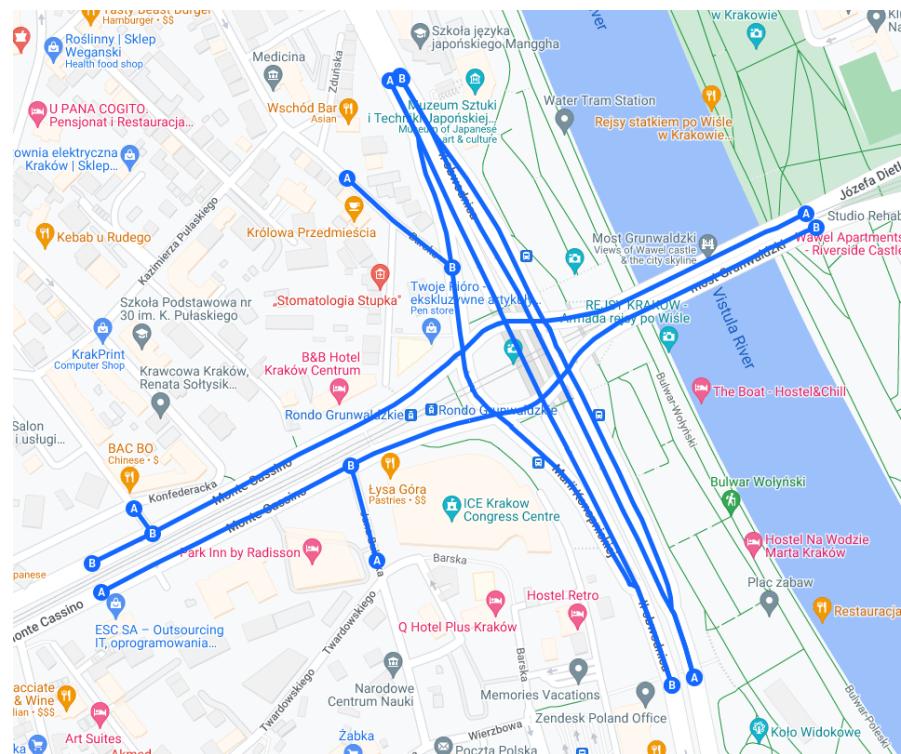
Niestety, mimo wprowadzonych poprawek, wizualizacja stanu symulacji nie jest idealna. Wynika to w głównej mierze z konieczności każdorazowego obliczania pozycji każdej komórki automatu, co jest bardzo czasochłonne.

## 5.7 Repozytorium kodu

Repozytorium kodu udostępnione jest publicznie na platformie GitHub [11].

## 6. REZULTATY SYMULACJI

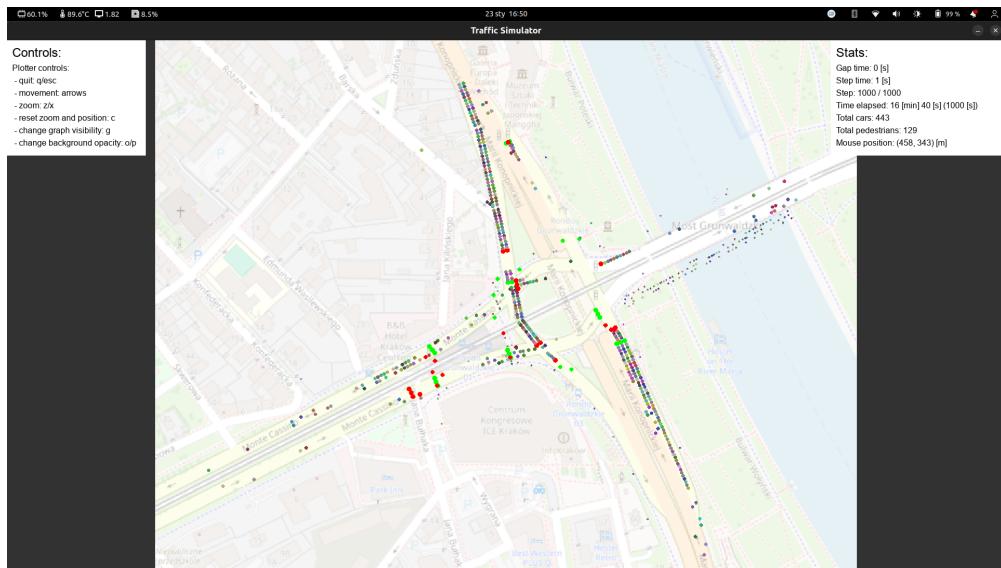
Na zaimplementowanym symulatorze przeprowadzono symulację testową oraz trzy symulacje badawcze. Za obszar modelu zgodnie ze wstępny problemem pracy przyjęto Rondo Grunwaldzkie w Krakowie (dokładny obszar przedstawiono na 6.1), a za porę przyjęto przedpołudnie dnia roboczego.



Rysunek 6.1: Obszar symulacji. Źródło: Google My Maps

### 6.1 Scenariusz walidacyjny

Pierwsza symulacja miała charakter walidacyjny. Model został więc skonfigurowany za pomocą zebranych przez zespół danych zaprezentowanych w tabelach 3.1, 3.2 oraz 3.3. Stan symulacji po 16 min 40 s (1000 s) przedstawia rysunek 6.2.



Rysunek 6.2: Stan symulacji walidacyjnej po 1000 sekundach.

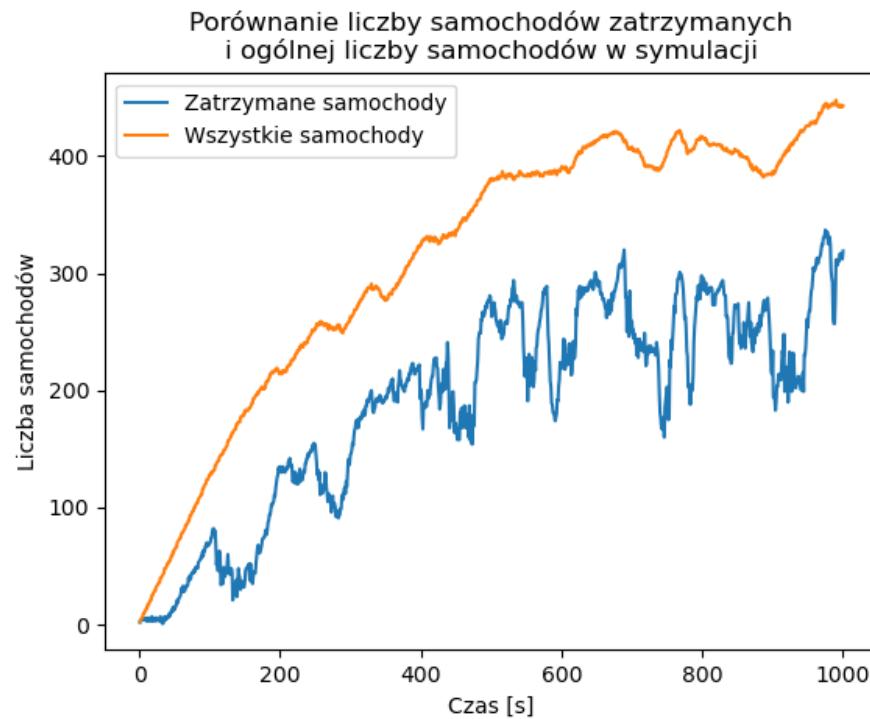
Zgodnie z planem dane z symulacji wykorzystaliśmy do analizy czasu zatrzymania samochodów przed wjazdem na rondo. Wyniki, dla zwiększenia czytelności, zdecydowaliśmy się przedstawić w formie wykresu, który przedstawia rysunek 6.3.

## 6.2 Scenariusz badawczy 1

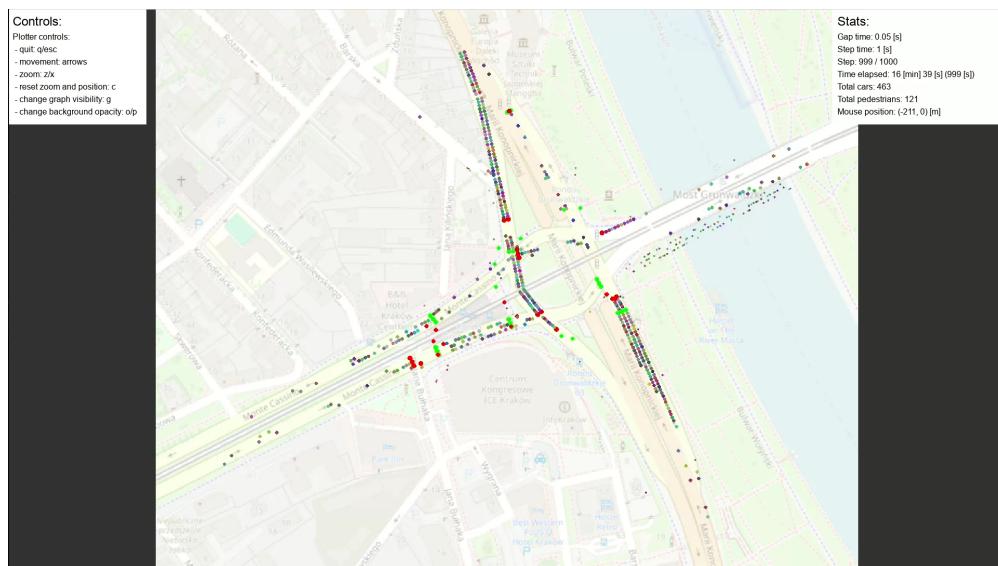
Pierwsza symulacja badawcza sprawdzała przystosowanie badanego ronda do sytuacji nasilenia ruchu ze strony zachodniej. W tym celu, dane wejściowe zmodyfikowano dwukrotnie zwiększającczęstość pojawiania się samochodów na ul. Monte Cassino na wysokości budynku nr 15 oraz uruchomiono symulację na 1000 sekund. Końcowy stan symulacji przedstawia rysunek 6.4. Dane do analizy czasu zatrzymania samochodów przed wjazdem na rondo ponownie zwizualizowano i przedstawiono na rysunku 6.5.

## 6.3 Scenariusz badawczy 2

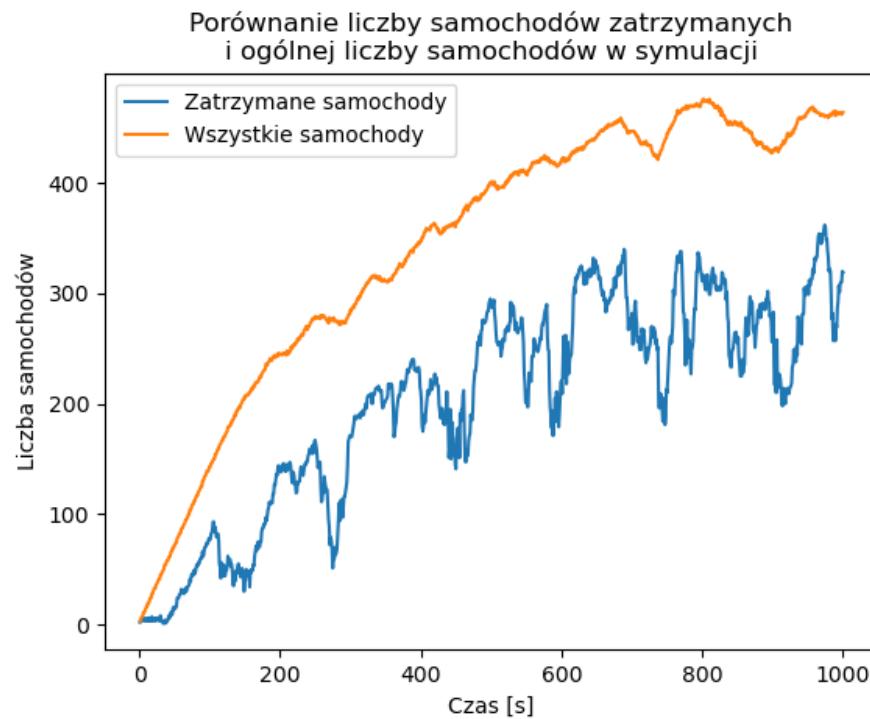
Druga symulacja badawcza sprawdzała przystosowanie badanego ronda do sytuacji nasilenia ruchu ze strony wschodniej. W tym celu, dane wejściowe zmodyfikowano dwukrotnie zwiększającczęstość pojawiania się samochodów przy wjeździe na Most Grunwaldzki oraz uruchomiono symulację na 1000 sekund. Końcowy stan symulacji przedstawia rysunek 6.6. Dane do analizy zwizualizowano i przedstawiono na rysunku 6.7.



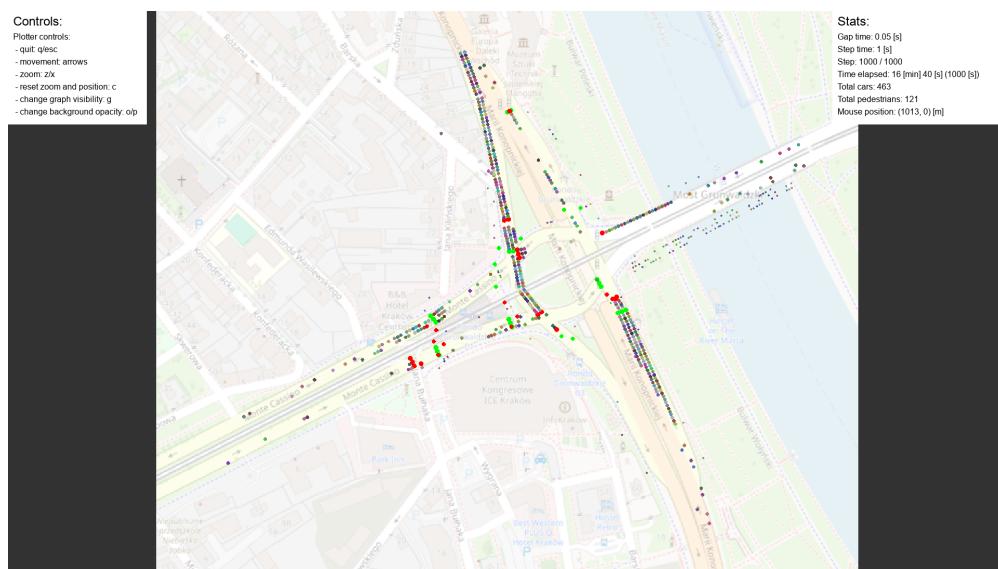
Rysunek 6.3: Analiza szczegółowa scenariusza walidacyjnego: Wykres przedstawiający liczbę stojących samochodów w porównaniu do liczby wszystkich samochodów.



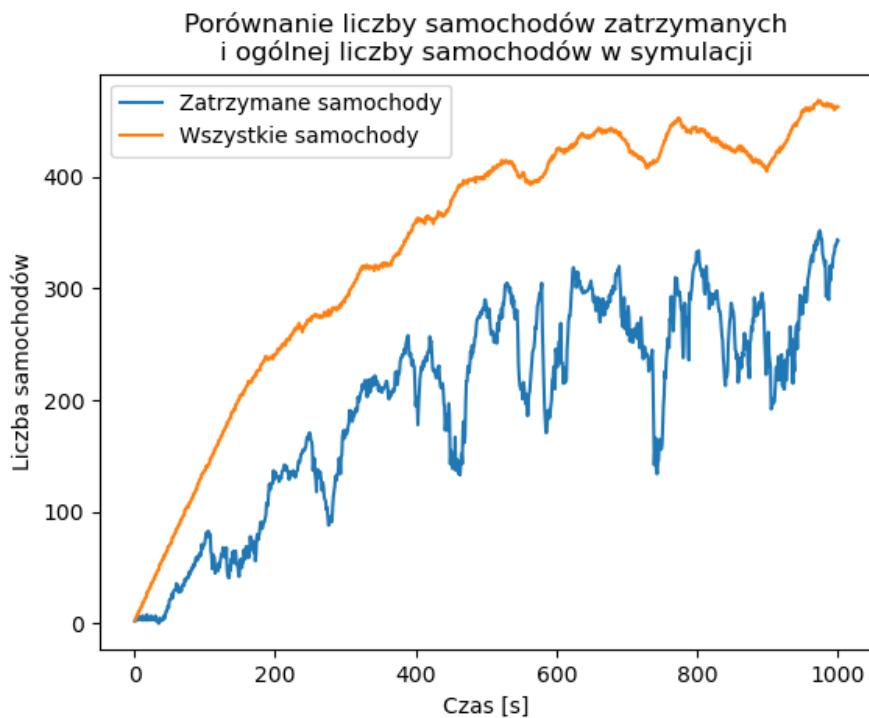
Rysunek 6.4: Stan symulacji scenariusza badawczego 1 po 1000 sekundach.



Rysunek 6.5: Analiza szczegółowa scenariusza badawczego 1: Wykres przedstawiający liczbę stojących samochodów w porównaniu do liczby wszystkich samochodów.



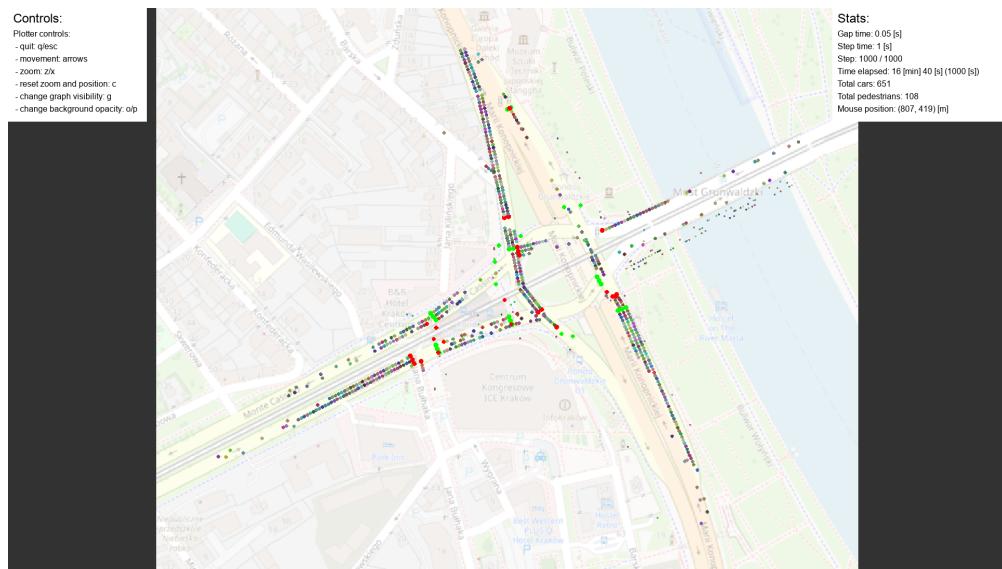
Rysunek 6.6: Stan symulacji scenariusza badawczego 2 po 1000 sekundach.



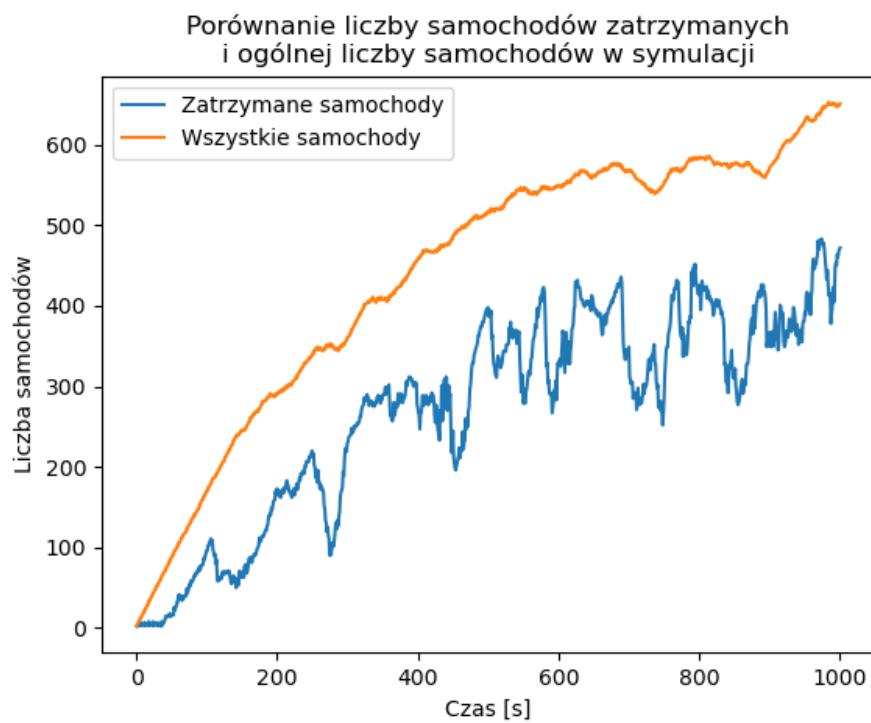
Rysunek 6.7: Analiza szczegółowa scenariusza badawczego 2: Wykres przedstawiający liczbę stojących samochodów w porównaniu do liczby wszystkich samochodów.

## 6.4 Scenariusz badawczy 3

Trzecia symulacja badawcza sprawdzała przystosowanie badanego ronda do sytuacji nasilenia ruchu jednocześnie ze strony zachodniej oraz wschodniej. W tym celu, dane wejściowe zmodyfikowano zwiększając pięciokrotnieczęstość pojawiania się samochodów na ul. Monte Cassino na wysokości budynku nr 15 oraz dwukrotnie przy wjeździe na Most Grunwaldzki. Następnie uruchomiono symulację na 1000 sekund. Końcowy stan symulacji przedstawia rysunek 6.8. Zwizualizowane dane do analizy przedstawia rysunek 6.9.



Rysunek 6.8: Stan symulacji scenariusza badawczego 3 po 1000 sekundach.



Rysunek 6.9: Analiza szczegółowa scenariusza badawczego 3: Wykres przedstawiający liczbę stojących samochodów w porównaniu do liczby wszystkich samochodów.

## 7. DYSKUSJA WYNIKÓW

### 7.1 Omówienie wyników

#### 7.1.1 Scenariusz walidacyjny

Uzyskane z symulacji walidacyjnej wyniki z jakościowego punktu widzenia są zgodne z doświadczeniem Ronda Grunwaldzkiego przez członków zespołu. Samochody oraz piesi poruszają się w sposób w zadowalającym stopniu realistyczny.

#### 7.1.2 Scenariusze badawcze

Analiza wykresów [6.3](#), [6.5](#), [6.5](#) oraz [6.9](#) obrazuje zależność między liczbą zatrzymanych i wszystkich samochodów. Niezależnie od scenariusza oraz chwili czasowej, około **66%** samochodów jest zatrzymanych.

Zwrócić uwagę należy również na okresowe wahnięcia w liczbie zatrzymanych samochodów. Pokazują one globalną cykliczność zmiany stanów świata w symulowanym środowisku.

Wyniki scenariusza badawczego 1 pokazują, że wzmożony ruch z zachodniego wjazdu nie ma znacznie negatywnego wpływu na komfort przejazdu przez Rondo Grunwaldzkie. Jak pokazuje rysunek [6.5](#), na drodze wjazdowej tworzy się mały korek.

Jak pokazuje analiza scenariusza badawczego 2, wzmożony ruch z Mostu Grunwaldzkiego nie ma wpływu na ruch na samym rondzie. Sytuacja taka jest jednak uciążliwa dla kierowców, ponieważ tworzący się na wschodnim wjeździe korek wydłuża się około dwukrotnie.

Rezultaty scenariusza badawczego 3 wskazują jednoznacznie, że gwałtowny, jednoczesny wzrost natężenia ruchu ze wschodu oraz, w jeszcze większym stopniu, z zachodu powoduje duże problemy z płynnością przejazdu przez Rondo Grunwaldzkie. W takim przypadku zakorkowana jest praktycznie cała badana część ulicy Monte Cassino oraz

część ulicy dojazdowej z Mostu Grunwaldzkiego.

## 7.2 Wnioski i obserwacje

Analiza czasu zatrzymania samochodów przed wjazdem na rondo pokazuje, że organizacja ruchu Rondzie Grunwaldzkim pozostawia wiele do życzenia. Rondo, szczególnie względem wjazdu północnego, nie jest wdrożne, co sprawia, że znajdujące się na nim samochody blokują wjazd samochodom czekającym na wjazd. Powoduje to, że zdecydowana większość samochodów zmuszona jest stać w długim korku, zanim jeszcze wjedzie na rondo.

Z kierunku zachodniego i wschodniego wzmożony ruch jest akceptowalnie przyjmowany i rozkładany przez rondo. Niestety, w przypadku kilkukrotnych wzrostów natężenia, które mogą pojawić się np. w godzinach szczytu, rondo okazuje się nie być wystarczająco drożne i znacznie się korkuje.

Największym problemem zdaje się być dramatyczna niedrożność ruchu z południa oraz, szczególnie, północy. Najprawdopodobniej jest to spowodowane nieoptimalnym umiejscowieniem świateł, przez które na rondo wjechać może zaledwie ułamek samochodów, które na to czekają.

## **8. PODSUMOWANIE**

Po przeprowadzeniu testowej symulacji sporządzono podsumowanie całokształtu pracy nad projektem z uwzględnieniem problemów napotkanych podczas implementacji oraz możliwych kierunków usprawnienia oraz dalszego rozwoju symulatora.

### **8.1 Napotkane problemy**

#### **8.1.1 Określenie na którym pasie powinien ustawić się samochód przed skrzyżowaniem.**

Problem ten okazał się sprowadzać do zdefiniowania części skrętu "w prawo" lub "w lewo" względem kierunku, z którego nadjeżdża samochód. Rozwiążanie polegało na analizie kątów, pod którymi rozchodzą się drogi od danego skrzyżowania, z uwzględnieniem kąta natarcia samochodu, który musi podjąć decyzję o ewentualnej zmianie pasa ruchu.

#### **8.1.2 Rysowanie jezdni - odstęp między pasami oraz przesunięcie w przypadku ulicy dwukierunkowej.**

W tym przypadku rozwiązanie polegało na wyznaczeniu kąta nachylenia jezdni. Następnie, korzystając z zależności trygonometrycznych mogliśmy wygodnie operować wartościami przesunięć poszczególnych pasów, a także całych jezdni.

#### **8.1.3 Implementacja macierzy prawdopodobieństwa zmiany pozycji pieszych.**

Rozwiązaniem tego problemu okazała się być zmiana podejścia. Z uwagi na dużą ilość przypadków szczególnych, które zerują całą macierz oprócz jednego pola, wygodniej

było takie przypadki opisać w ciągu instrukcji warunkowych sprawdzających występowanie poszczególnych przypadków. Dodatkowa przypadkowość podejmowania części decyzji realizowana jest poprzez zmienne pseudolosowe. W przypadku, gdy żaden warunek szczególny nie jest spełniony, stosowana jest ogólna reguła przejścia.

## 8.2 Kierunki rozwoju

Po przeprowadzeniu analizy korzystania z symulatora można wskazać następujące kierunki rozwoju:

- skonfigurowanie kolejnych profili czasowych (na przykład odpowiadających godzinom szczytu, weekendom itd.) wraz z możliwością ich dynamicznej podmiany podczas symulacji,
- implementacja bardziej realistycznego algorytmu sygnalizacji świetlnej,
- wprowadzenie ruchu komunikacji miejskiej (w szczególności tramwaju), pojazdów uprzywilejowanych oraz różnych profili ruchu samochodów osobowych,
- ulepszenie interfejsu graficznego, zarówno pod względem wygody użytkowania, jak i estetycznym.

## 8.3 Słabe strony

- Proces tworzenia modelu wsadowego jest żmudny i czasochłonny.
- Światła określone są dla całej jezdni, a nie dla pojedynczych pasów ruchu.
- Symulacja nie uwzględnia części zasad ruchu drogowego jak m.in. znaki, czy zasada prawej ręki.

# References

- [1] P. Gora, “Adaptacyjne planowanie ruchu drogowego,” Praca Magisterska, Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki, 2010.
- [2] K. Nagel and M. Schreckenberg, “A cellular automaton model for freeway traffic,” *Journal de physique I*, vol. 2, no. 12, pp. 2221–2229, 1992.
- [3] A. Rasouli, “Pedestrian simulation: A review,” *arXiv preprint arXiv:2102.03289*, 2021.
- [4] “Openstreetmap,” <https://www.openstreetmap.org>, accessed: 2024-01-25.
- [5] “Python 3,” <https://www.python.org>, accessed: 2024-01-25.
- [6] “Numpy,” <https://numpy.org/>, accessed: 2024-01-25.
- [7] “Networkx,” <https://networkx.org/>, accessed: 2024-01-25.
- [8] “Python 3,” <https://www.pygame.org>, accessed: 2024-01-25.
- [9] “Pandas,” <https://pandas.pydata.org/>, accessed: 2024-01-25.
- [10] “Matplotlib,” <https://matplotlib.org/>, accessed: 2024-01-25.
- [11] “Repozytorium kodu na platformie github,” <https://github.com/coado/discrete-systems-simulation/>, accessed: 2024-01-25.