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 hab. inż. JAROSŁAW WĄS prof. AGH

Dr inż. MARCIN PIEKARCZYK

EAIiIB / Katedra Informatyki Stosowanej

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Kraków, Polska

20 stycznia 2024 r.

Spis treści

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 świateł 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 opierają się na teorii automatów komórkowych i modelu Nagela-Schreckenberga ([1], [2]). Model ten służy on do symulacji ruchu pojazdów na prostym odcinku drogi.

Jego istotnym rozszerzeniem jest model TSF ([2]). 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:

- 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

Ich wartości, które służą 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 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 powszedniego. Zebrane informacje miały zarówno charakter ilościowy (liczba samochodów i pieszych uczestniczących w ruchu ulicznym) oraz jakościowy (ocena, czy w danej chwili na którejś ulicy występuje korek uliczny).

4. MODEL FORMALNY

4.1 Model TSF

Współcześnie najskuteczniejsze modele symulacji ruchu drogowego 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 krawedzi 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:

$$CA =< T, C, N, S, S_0, F >$$

gdzie:

- T - przedział czasu, w którym odbywa się ewolucja automatu,
- C - zbiór komórek,
- $N : C \rightarrow 2^C$ - funkcja, która każdej komórce przyporządkowuje zbiór 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 dalszego ułatwienia, zastosujemy oznaczenie $CELLS = CARS \cup PEDESTRIANS$. Wtedy:

$$C = CELLS \cup SIGNALS$$

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.

Ponieważ 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:

$$\forall_{c \in C} N(c) = C \cup SIGNALS$$

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 jest $null$. W przypadku sygnalizacji, stanem komórki jest fasą sygnalizacji ($GREEN$/RED). Zatem:

$$S = \{null\} \cup CARS \cup PEDESTRIANS \cup \{GREEN, RED\}$$

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: [2]

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 zbiołu $\{GREEN, RED\}$. 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 regułę przejścia definiuje algorytm przedstawiony na rysunku 4.1.

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

- $turnPenalty \in [0, 1]$ - zachowanie pojazdu przed skrętem,
- $crossroadPenalty \in [0, 1]$ - zachowanie pojazdu w trakcie przejazdu przez

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);
```

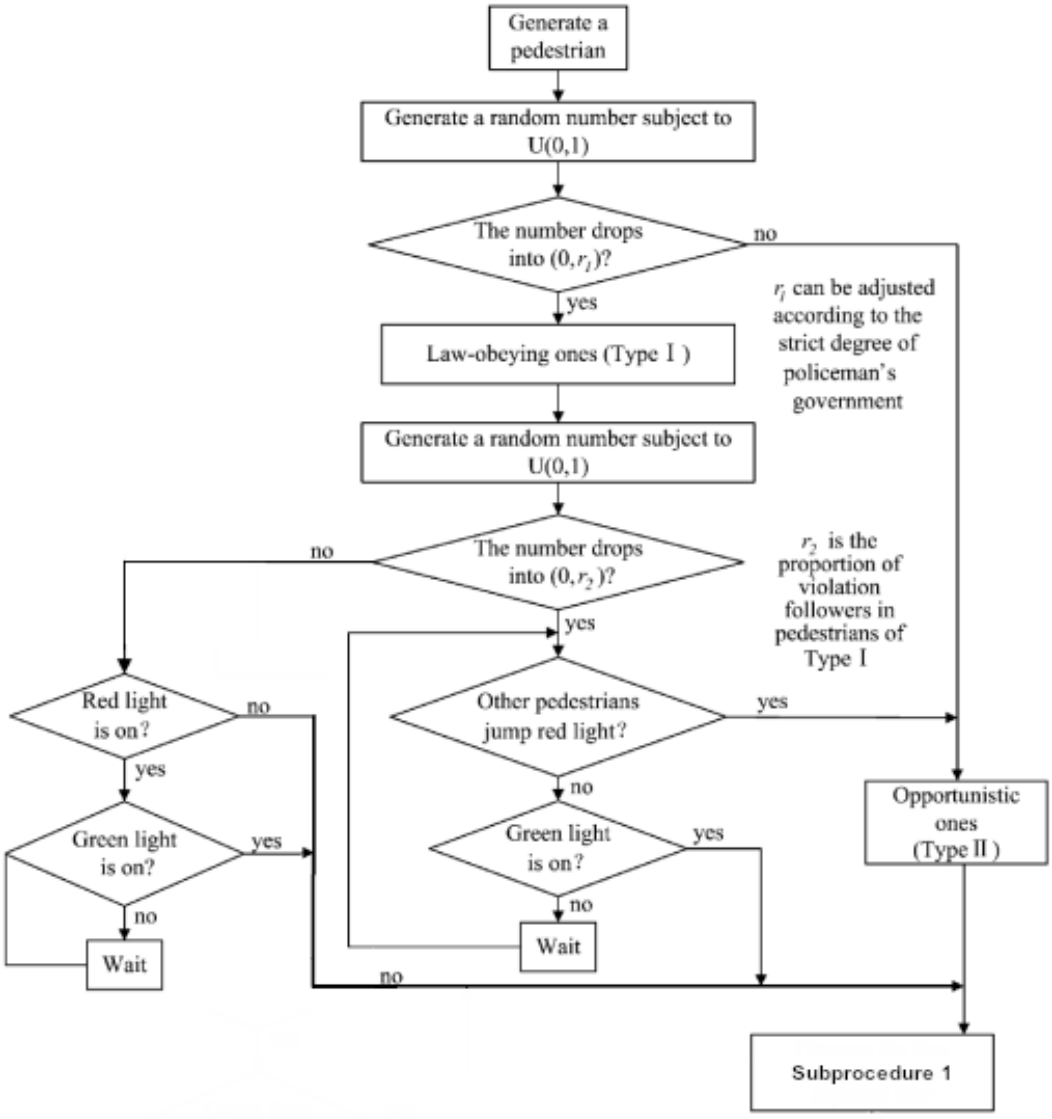
Rysunek 4.2: Algorytm opisujący funkcję przejścia dla komórek ze zbioru $CARS$. Źródło: [2]

skrzyżowanie (bez skręcania),

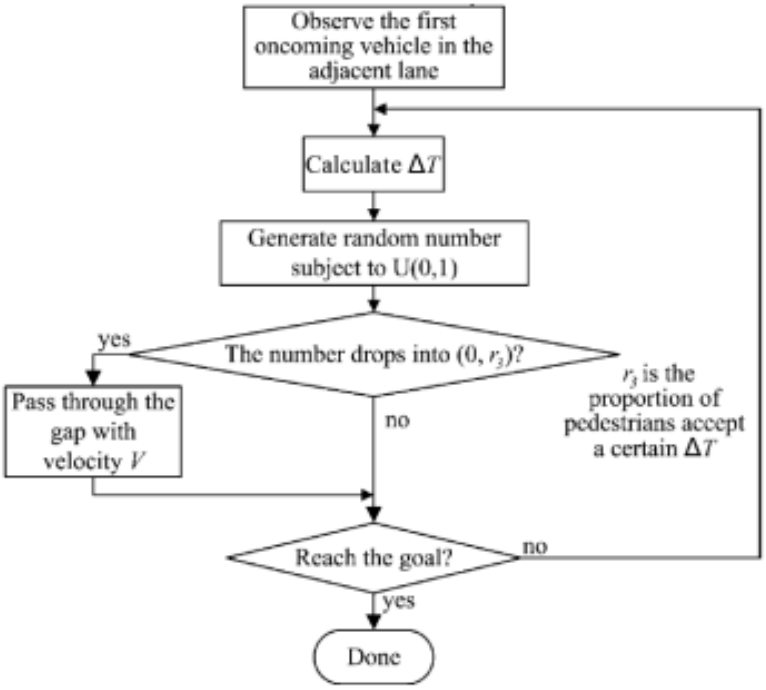
- $prob \in [0, 1]$ - losowa redukcja prędkości pojazdu.

Dla komórek ze zbioru PEDESTRIANS regułę przejścia definiuje głownie 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 3×3, 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 w wersji 3.9.18 z wykorzystaniem następujących bibliotek:

- numpy (v. 1.26.2)
- networkx (v. 3.1)
- pygame (v. 2.5.2)
- pandas (v. 2.1.4)
- matplotlib (v. 3.8.0)

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 poleceniem *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* (ilość kroków, które należy wykonać) oraz t_gap (minimalny odstęp czasowy pomiędzy krokami).

Metoda *step* wywołuje zadaną ilość 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 getterów do swoich cech (m.in. krok czasowy, obecny krok, krok maksymalny, czas trwania symulacji) oraz następujące gettery do obiektów *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*

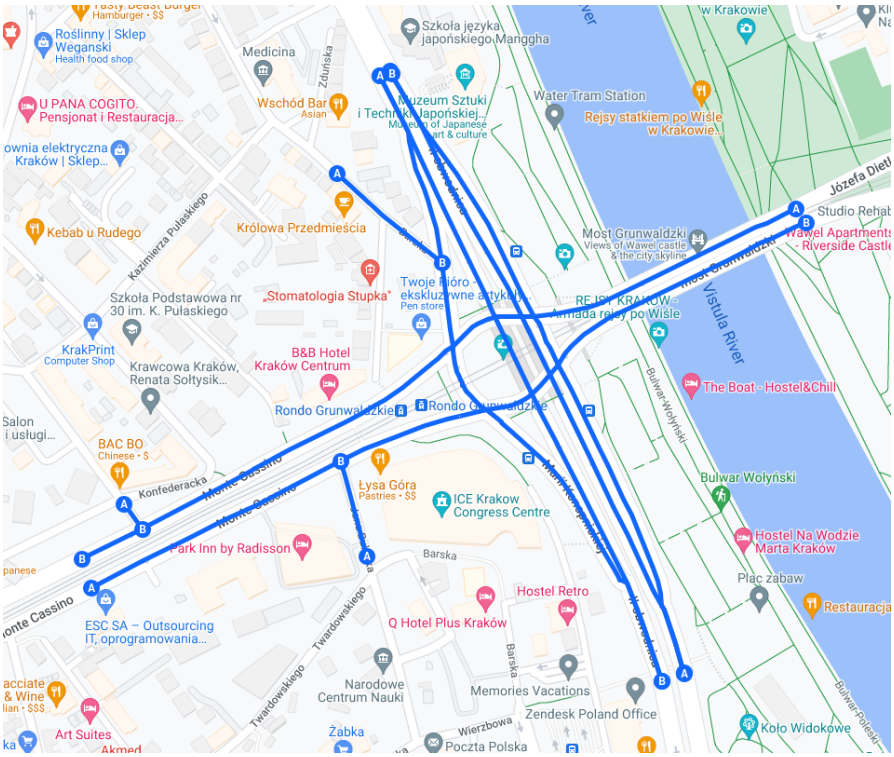
Plotter jest klasą reprezentującą interaktywny interfejs graficzny symulatora. Pozwala on na obrazowanie stanu wewnętrznego komórek symulatora w czasie rzeczywistym, wyświetlanie obrazu mapy w tle, a także manipulację widokiem (przybliżanie, przesuwanie). Dla ułatwienia tworzenia i weryfikacji modelu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

5.5 Repozytorium kodu

Kod dostępny jest do wglądu pod linkiem: github.com/coado/discrete-systems-simulation

6. REZULTATY SYMULACJI

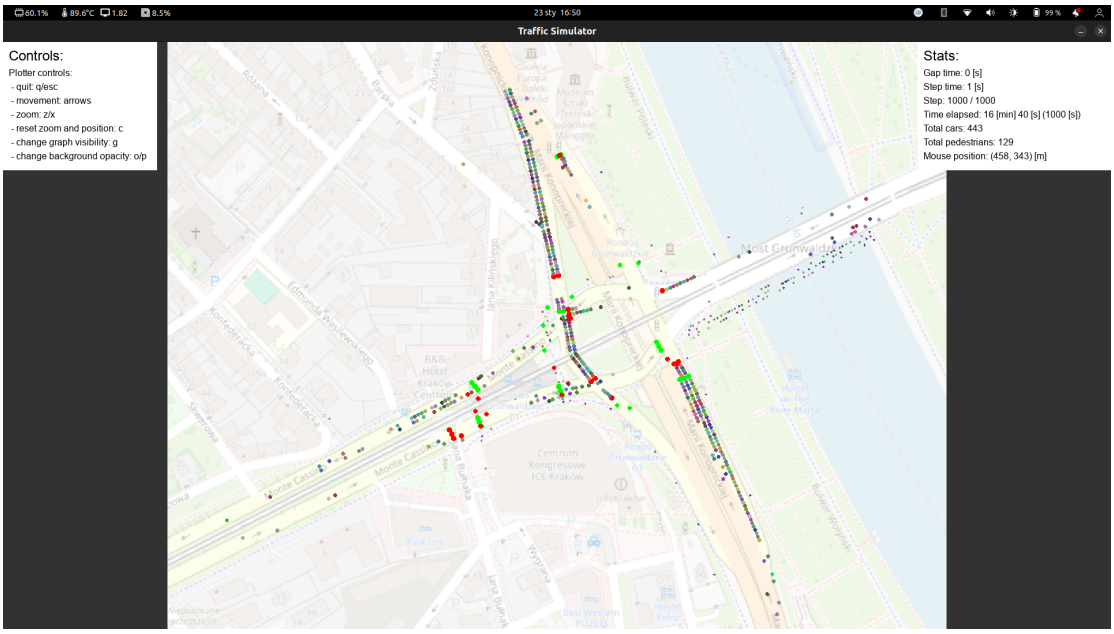
Na zaimplementowanym symulatorze przeprowadzono jedną symulację testową. Za obszar modelu zgodnie ze wstępnym problemem pracy przyjęto Rondo Grunwaldzkie w Krakowie (dokładny obszar przedstawiono na 6.1), a za porę przyjęto przedpołudnie dnia powszedniego.



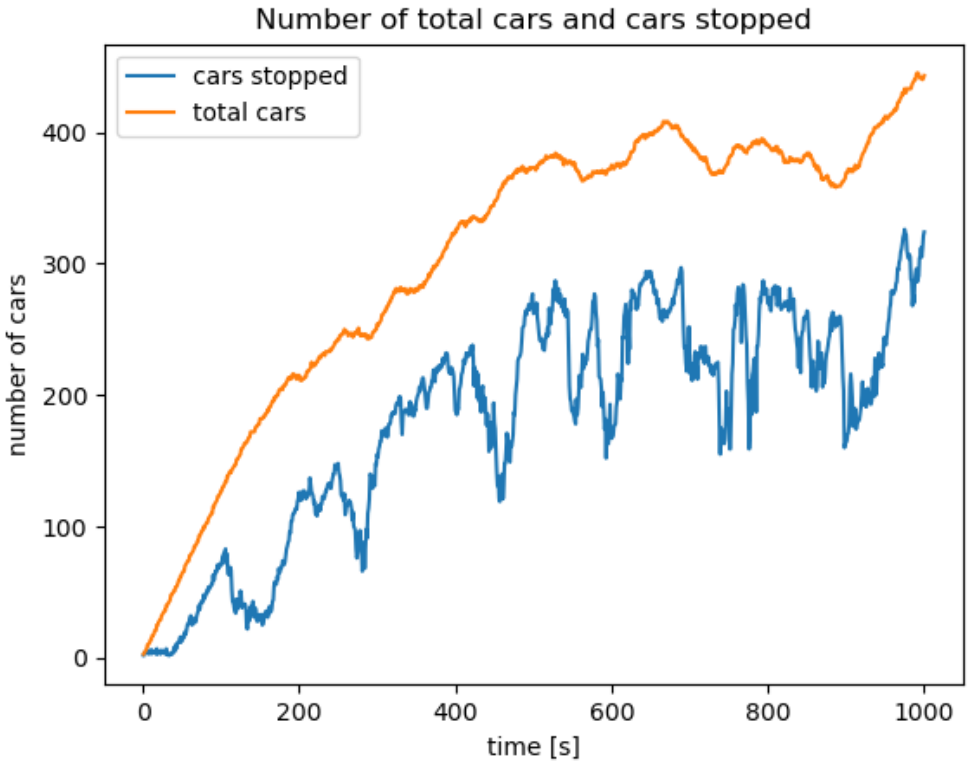
Rysunek 6.1: Obszar symulacji. Źródło: Google My Maps

Stan symulacji po 16 min 40 s (1000 s) przedstawia rysunek 6.2

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.



Rysunek 6.2: Stan symulacji po 1000 sekundach.



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

7. DYSKUSJA WYNIKÓW

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

Analiza czasu zatrzymania samochodów przed wjazdem na rondo pokazuje, że organizacja ruchu na badanym rondzie nie jest idealna. Zdecydowana większość samochodów jest zmuszona stać w długim korku przed światłami, zanim wjedzie na rondo. Samo rondo również nie jest w pełni drożne, co sprawia, że samochody, które chcą wykonać inny manewr niż jazda prosto, muszą czekać również na rondzie.

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 chęci skrętu "w prawo" lub "w lewo" względem kierunku, z którego nadjeżdża samochód. Rozwiązanie 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 miedzy 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] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal de physique I*, vol. 2, no. 12, pp. 2221–2229, 1992.

[2] P. Gora, "Adaptacyjne planowanie ruchu drogowego," Praca Magisterska, Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki, 2010.

[3] A. Rasouli, "Pedestrian simulation: A review," *arXiv preprint arXiv:2102.03289*, 2021.