Symulacja ruchu drogowego na IV obwodnicy Krakowa

Szymon Gałuszka, Michał Worsowicz, Maciej Nalepa

11 czerwca 2020

Część I
Omówienie projektu

1 Definicja problemu

Nasz cel to symulacja ruchu drogowego na IV obwodnicy Krakowa. Konieczna jest definicja tras oraz sposobu poruszania się po nich. Obwodnica tworzy zamkniętą pętlę (uwzględniając odcinek w budowie) i dzieli się na ruch zgodny i przeciwny do wsazówek zegara.

Ruch drogowy powinien uwzględniać pojazdy osobowe, ciężarowe i transport publiczny. Ciężkie pojazdy obowiązują inne ograniczenia prędkości oraz zajmują więcej przestrzeni na jezdni. Napływ ruchu powinien odbywać się przez wjazdy na obwodnicę, które wpuszczają samochody z określoną częstotliwością, która może zależeć od pory dnia.

Symulacja może obejmować wydarzenia losowe, takie jak:

- blokada pasa ruchu,
- zamknięcie zjazdu,
- nagłe hamowanie.

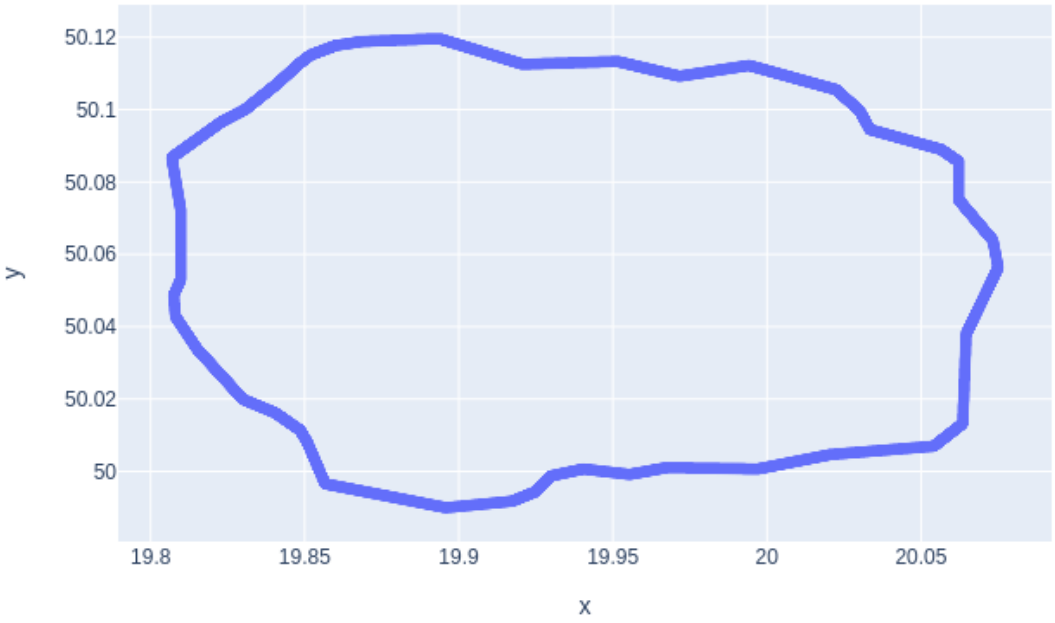
2 Obszar symulacji

Symulowany przez nas obszar to IV obwodnica Krakowa, znana także jako obwodnica autostradowa Krakowa, ponieważ większość jej odcinka stanowi autostrada A4.

Na obwodnicy nie ma sygnalizacji świetlnej, skrzyżowania znajdują się najczęściej pod wiaduktem i najpierw należy zjechać z drogi szybkiego ruchu. Trasa ma zróżnicowane ograniczenia prędkości oraz różną ilość pasów ruchu.

Fragment ten na zachodnich obrzeżach miasta jest dwupasmowy, a na południowych - trójpasmowy. W najbliższym czasie przewidywana jest budowa odcinka północnego, który uwzględniliśmy w naszej symulacji, odtwarzając i dodając ten fragment do projektu. Poniżej przedstawiony jest wykaz węzłów drogowych, które obejmuje symulacja (zgodnie z kierunkiem ruchu wskazówek zegara):

- Kraków Nowa Huta,
- Kraków Przewóz,
- Kraków Bieżanów,
- Kraków Wieliczka,
- Kraków Łagiewniki,
- Kraków Południe,
- Kraków Skawina,
- Kraków Tyniec,
- Kraków Bielany,
- Kraków Balice (lotnisko),
- Balice I,
- Modlniczka,
- Modlnica,
- Kraków Północ – dla uproszczenia przyjęto taką nazwę planowanego węzła (wariant I)



Rysunek 1: Mapa obwodnicy użytej w symulacji.

3 Algorytm

Zastosowany przez nas algorytm ruchu drogowego to zmodyfikowany model Nagela-Schreckenberga, będący teoretycznym modelem mikroskopowym o charakterze dyskretnym, w którym obie jednokierunkowe jezdnie są podzielone na komórki, które zawierają informację na temat ilości pasów oraz ilości pojazdów w danej chwili na jezdni. Dzięki takiemu uproszczeniu, nie jest konieczna symulacja zmiany pasu ruchu, ponieważ nie jest on określony wprost.

Każda z komórek może być interpretowana na jeden ze sposobów:

- komórka pusta – fragment pustej drogi
- komórka używana – fragment zawierający pojazd na nieokreślonym pasie
- komórka pełna – fragment drogi z pojazdem na wszystkich pasach ruchu

Każdy samochód n posiada swoją prędkość $v(n)$ o wartości liczby naturalnej, nie większej od prędkości maksymalnej v_{max} – symbolizuje ona maksymalną prędkość osiąganą przez samochód.

Czas t w modelu przyjmuje wartość dyskretną o stałym kroku. Nasz algorytm można podzielić na następujące po sobie czynności:

1. Losowość:
   Czynność ta symbolizuje wszelkie przypadki losowe z jakimi kierowca może spotkać się na drodze. Dla każdego samochodu n prędkość może zostać zmniejszona, zwiększona lub dopasowana do ograniczenia na drodze (hamowanie lub przyspieszenie o wartość 1 w kierunku przepisowej prędkości).

2. Zwalnianie:
   Jeżeli odległość $d(n)$ samochodu n od samochodu znajdującego się przed nim jest mniejsza od prędkości $v(n)$ tego samochodu to prędkość samochodu ulega zmniejszeniu. Jako że odległość mierzona jest w liczbie komórek, a prędkość w liczbie komórek na jednostkę czasu, to prędkość ostateczna może wynosić maksymalnie $d(n)$ na jednostkę czasu, a minimalnie zero.

3. Ruch samochodu:
   W ostatnim kroku każdy z samochodów zostaje przesunięty do przodu o odpowiednią ilość komórek, wynikającą z jego prędkości.

Część II
Realizacja

4 Struktury danych

Obsługiwany plik mapy `SimpleMap` jest w formacie `json` i przybiera następujący format:

```
{
        "nazwa wjazdu;nazwa zjazdu": {
                "lanes": liczba_pasow,
                "maxspeed": ograniczenie_predkosci,
                "geometry": [
                        [lat, lon],
                        ...
                ]
        },
        ...
}
```

Rysunek 2: Przykład pliku w formacie SimpleMap.

Dla kompatybilności z pierwszą realizacją mapy za pomocą formatu OpenStreetMap-json powstał moduł `OpenMap`. Użycie tego formatu wymaga konwersji z formatu `.osm` do `.json`, może do tego posłużyć program JOSM.

Automat komórkowy – `Cellular` przechowuje listę następujących obiektów: pojazdów (agentów), komórek, punktów startowych i końcowych. Umożliwia budowanie mapy na podstawie pliku `json` kompatybilnego z modułem opracowania własnego `SimpleMap`.

Komórka – `Cell` przechowuje wskaźnik do następnej komórki, ilość dostępnych pasów, dopuszczalną prędkość oraz aktualne wykorzystanie.

Logi – `Stat` zapisuje dane w postaci ramki danych `pandas`. W każdej iteracji dopisywane są wiersze z monitorowanymi danymi.

Konfiguracja – `CONFIG` udostępniony przez moduł utils pozwala na podłączenie klasy definiującej statycznie następujące parametry:

- Timestep – krok czasu t,
- Radius – rozmiar komórki,
- Spawn rate – domyślne prawdopodobieństwo pojawienia się nowego pojazdu w danym kroku,
- Agent driveoff – prawdopodobieństwo opuszczenia obwodnicy,
- Agent slow – domyślne prawdopodobieństwo hamowania,
- Agent fast – domyślne prawdopodobieństwo przyspieszenia,
- Agent limit – domyślne prawdopodobieństwo dążenia do ograniczenia prędkości,
- Agent Vmax – domyślna maksymalna prędkość samochodu

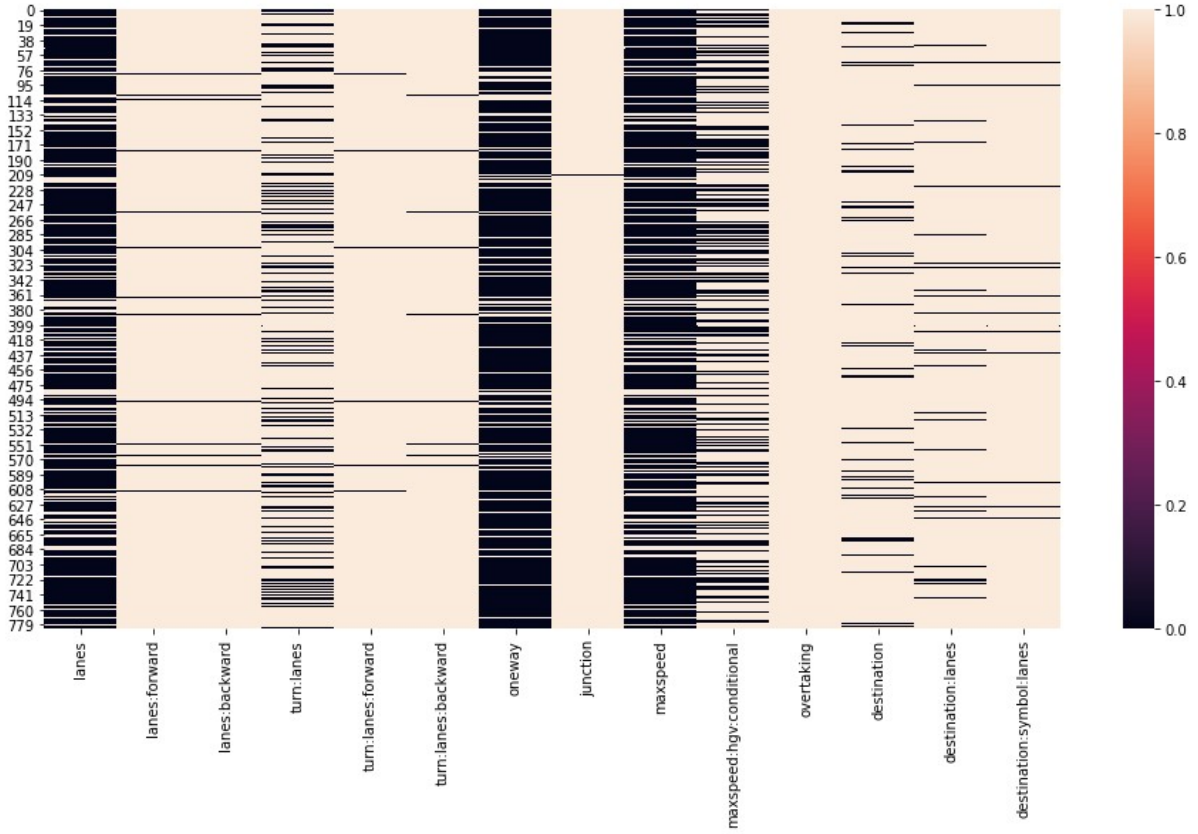
5 Implementacja

5.1 Mapa

5.1.1 OpenStreetMap

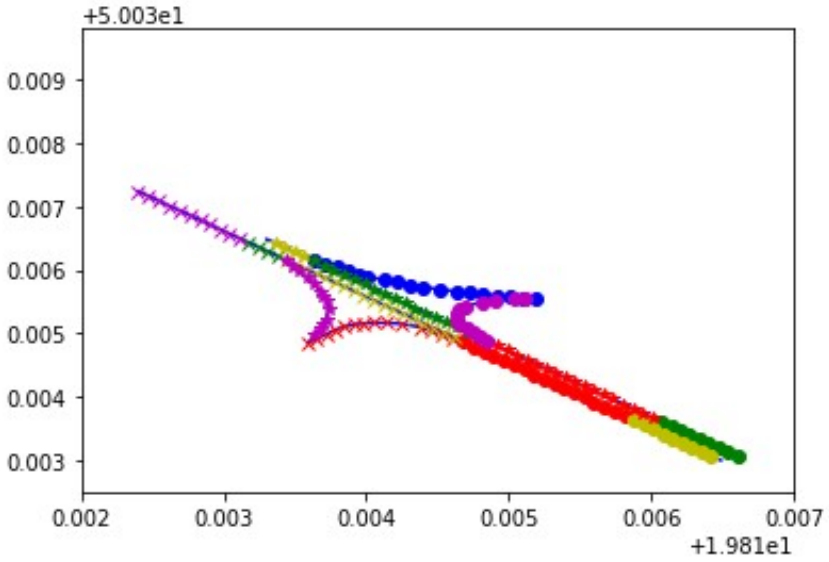
Początkowo projekt zakładał wykorzystanie formatu OSM do budowania mapy. Okazało się, że dane wymagają dużego nakładu pracy w celu poprawek i wypełnienia braków

(3).



Rysunek 3: Brakujące dane w pliku OSM. (biały – oznacza brak danych | czarny – dane występują)

Ponadto, drogi OSM były podzielone na dużo bardzo krótkich odcinków (4). Wymuszało to implementację algorytmu, który rozwiązałby istniejące relacje między odcinkami, tak aby je odpowiednio połączyć.

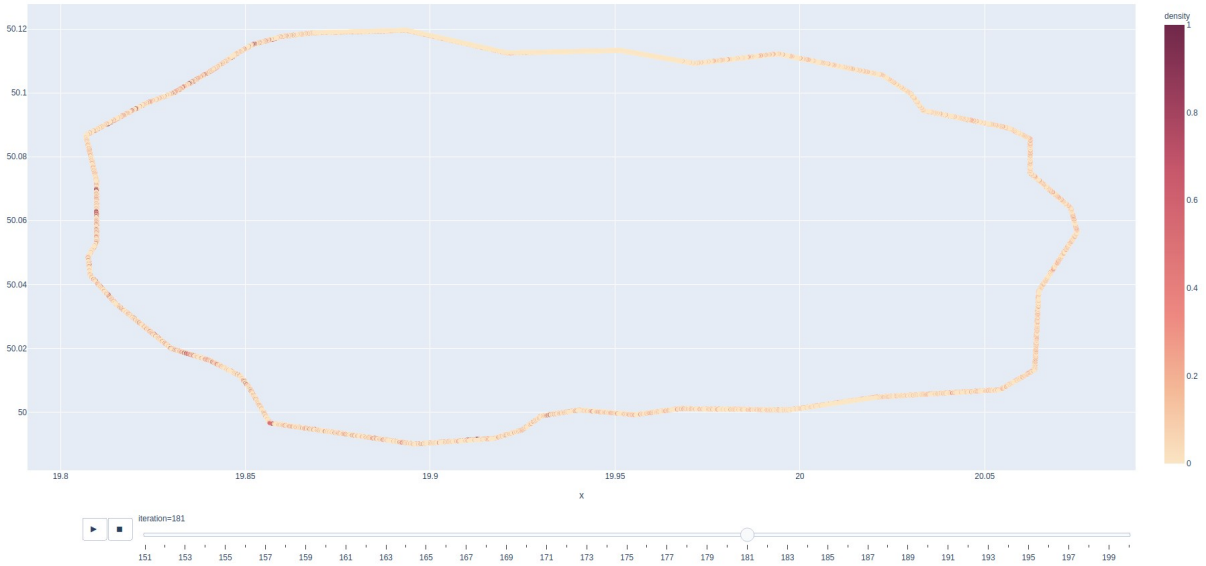


Rysunek 4: Każdy kolor i kształt oznacza oddzielny odcinek drogi (OSM).

5.1.2 SimpleMap

`SimpleMap` to moduł opracowania własnego, którego celem jest odczyt i preprocessing danych w formacie zdefiniowanym w sekcji Struktur danych (2). Stanowi to duże ułatwienie w pracy z mapą, ponieważ ten format jasno określa początek i koniec odcinka, co w przypadku OSM było nietrywialnym problemem.

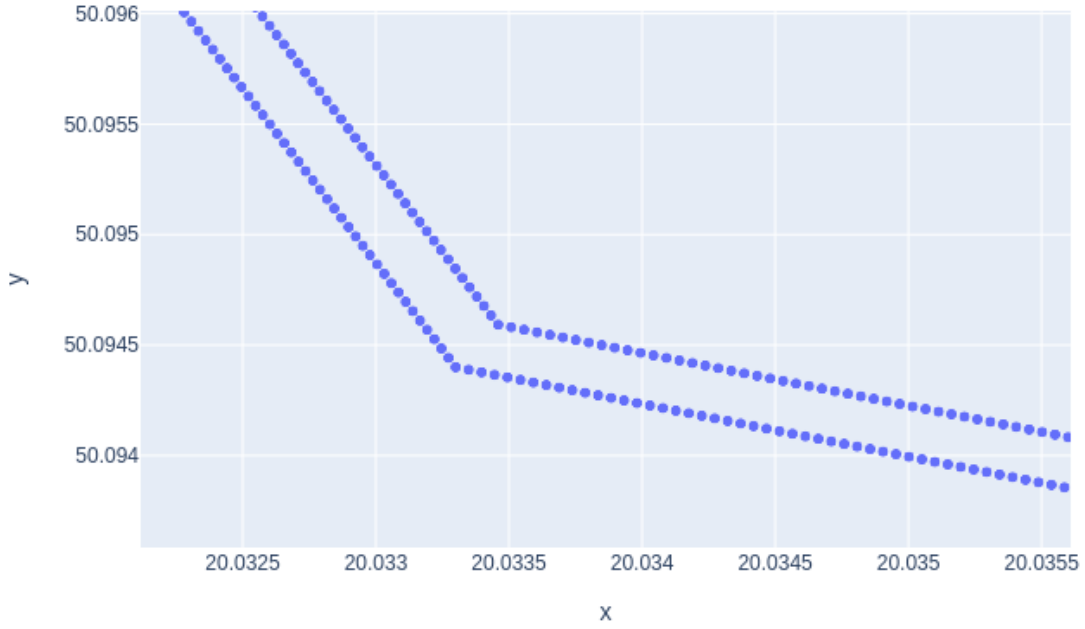
Dane dzielimy na odcinki drogi, określając sortowanie współrzędnych zgodnie lub przeciwnie do ruchu wskazówek zegara. Dodatkowe parametry to ilość pasów oraz dopuszczalna prędkość maksymalna.



Rysunek 5: Animowana mapa symulacji.

5.2 Automat komórkowy

Automat zdefiniowany jako `Cellular` w module `core` odpowiada za budowę mapy bazując na obiekcie `SimpleMap`. Generuje on siatkę odpowiednio połączonych komórek, tworzy przeciwny, przesunięty pas ruchu, oznacza początek i koniec drogi (od wjazdu do zjazdu) i rozwiązuje połączenia komórek między odcinkami.



Rysunek 6: Zbliżenie na komórki mapy.

Funkcja `step` odpowiada za obliczenie kroku symulacji. Przyjmuje ona jako parametr listę metryk (`Stat`), co umożliwia wydajne i w pełni określone przez użytkownika monitorowanie symulacji.

5.3 Metryki

W celu prowadzenia statystyk w wygodny i uniwersalny sposób opracowaliśmy klasę `Stat`. Aby ją wykorzystać, należy stworzyć nową klasę dziedziczącą po niej i nadpisującą definicję `extract`, która decyduje o tym jakie dane wyciągamy z aktualnego stanu symulacji i zapisujemy jako wiersze w ramce danych (4).

Każda metryka dziedzicząca po klasie `Stat` umożliwia zapis i odczyt z pliku.

Predefiniowaliśmy następujące metryki:

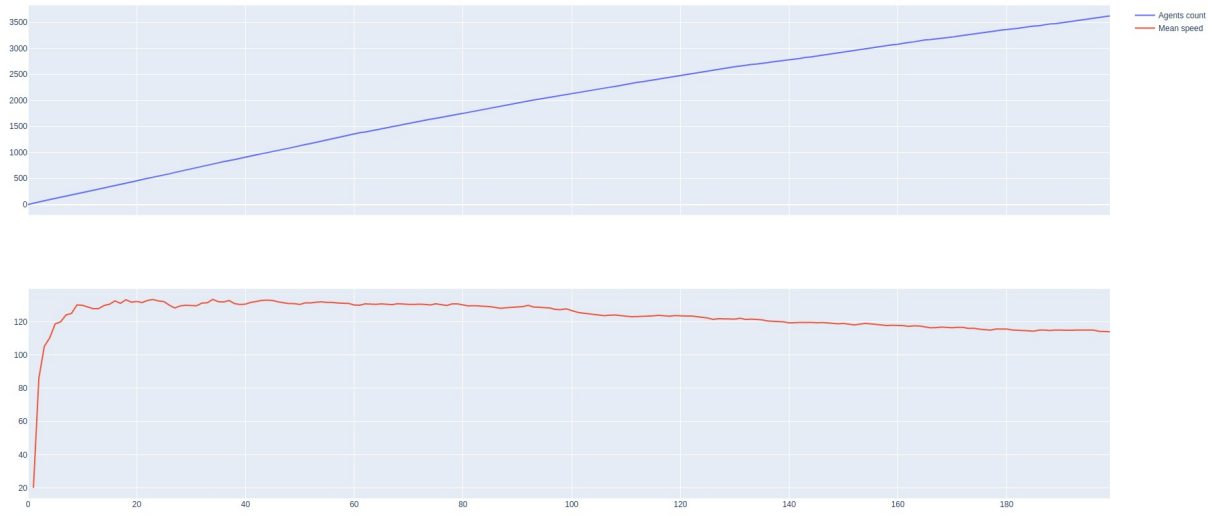
- InOutFlowStat – zapisuje ile pojazdów i gdzie opuściło lub wjechało na obwodnicę,
- CellStat – zapisuje aktualny stan wszystkich komórek mapy,
- LastCellStat – CellStat przechowujący określoną ilość iteracji wstecz,
- AgentStat – zapisuje stany wszystkich pojazdów na drodze.

5.4 Wykresy

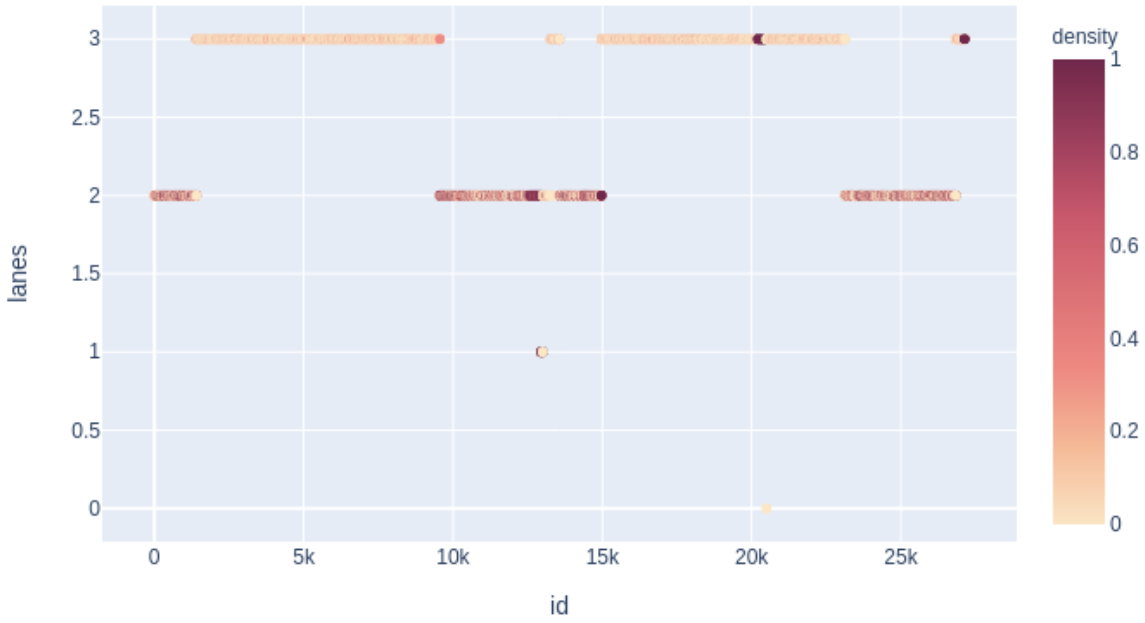
Moduł `renderer` zawiera implementację klasy `Plotter`, która umożliwia w łatwy sposób na tworzenie interaktywnych wykresów wykorzystujących bibliotekę Plotly [4].

Dane dostarczane są w ramce danych `pandas`. Podstawowa klasa umożliwia tworzenie wykresów na podstawie słownika `fields`, który decyduje jakie parametry przekazujemy do Plotly Express. Zapewnia to dużą swobodę i jest bardzo prostym rozwiązaniem.

Stworzyliśmy też trzy predefiniowane klasy do szybkiego rysowania podstawowych statystyk: AgentMetrics, FlowMetrics oraz CellularMap do rysowania aktualnej mapy.



Rysunek 7: AgentMetrics: Wykres na podstawie wybranych danych zebranych przez AgentStat.



Rysunek 8: Przykładowy wykres zdefiniowany przy użyciu pola `fields`. Kod dostępny w notebooku.

6 Usprawnienie

Najsłabsze punkty implementacji:

- Zapis jednego stanu mapy to ramka danych o rozmiarze ilości komórek na mapie. Oznacza to, że pojedyńcza sekunda symulacji generuje nawet 30000 wpisów do ramki. W celu uniknięcia problemu z zapełnianiem pamięci stworzono specjalną klasę `LastCellStat` przechowującą określoną ilość stanów wstecz. Lepszym rozwiązeniem może być zapis logów bezpośrednio do pliku.
- Konfiguracja prawdopodobieństwa opuszczenia skrzyżowania jest globalna, powinna być zależna od węzła.

Część III
Podsumowanie

7 Wyniki

Przeprowadziliśmy symulację w trzech wariantach: normalnego ruchu, ruchu ze zwężeniem na odcinku Bielany-Balice, blokada drogi na odcinku Bieżanów-Przewóz.

Symulację można powtórzyć samodzielnie za pomocą jupyter notebook `automata.ipynb` z możliwością modyfikacji parametrów.

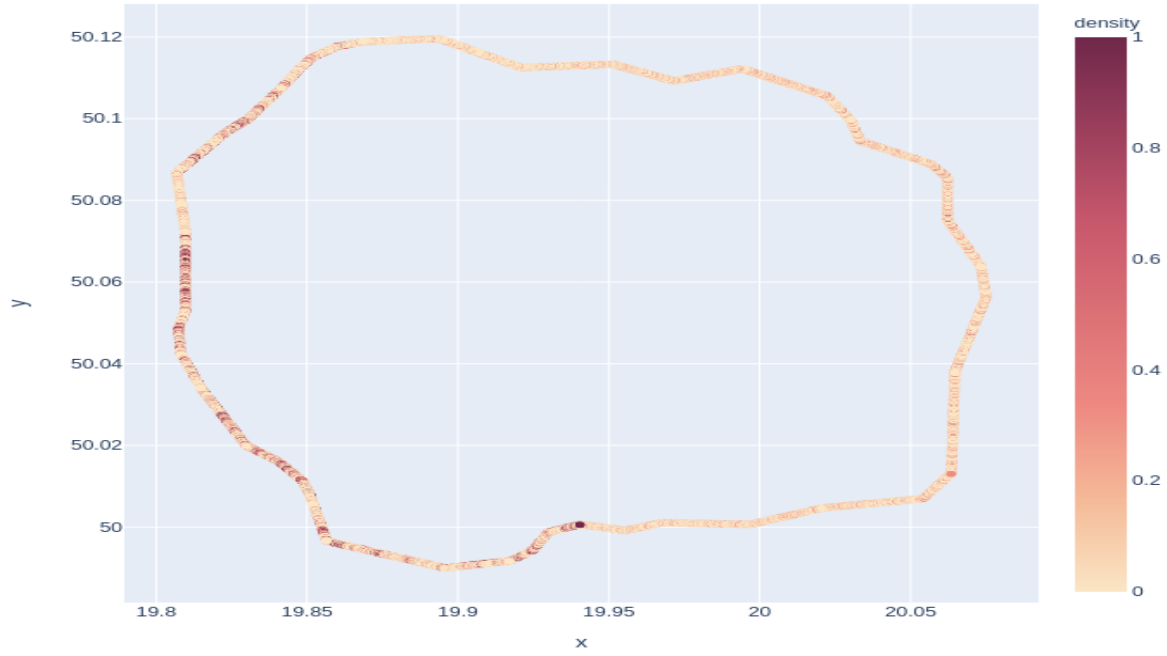
Domyślne parametry definiują intensywny ruch drogowy (średnio 48 samochodów na minutę wjeżdża z każdego wjazdu).

Każdy wariant był symulowany przez 500 sekund, a symulacja nie była resetowana. Oznacza to, że każdy wariant był budowany na poprzednim.

1. wariant ruchu normalnego (iteracje: do 500),
2. wariant ruchu z utrudnieniem w postaci zwężenia (iteracje: od 500 do 1000),
3. wariant ruchu z całkowitą blokadą drogi (iteracje: od 1000 do 1500).

7.1 Ruch normalny

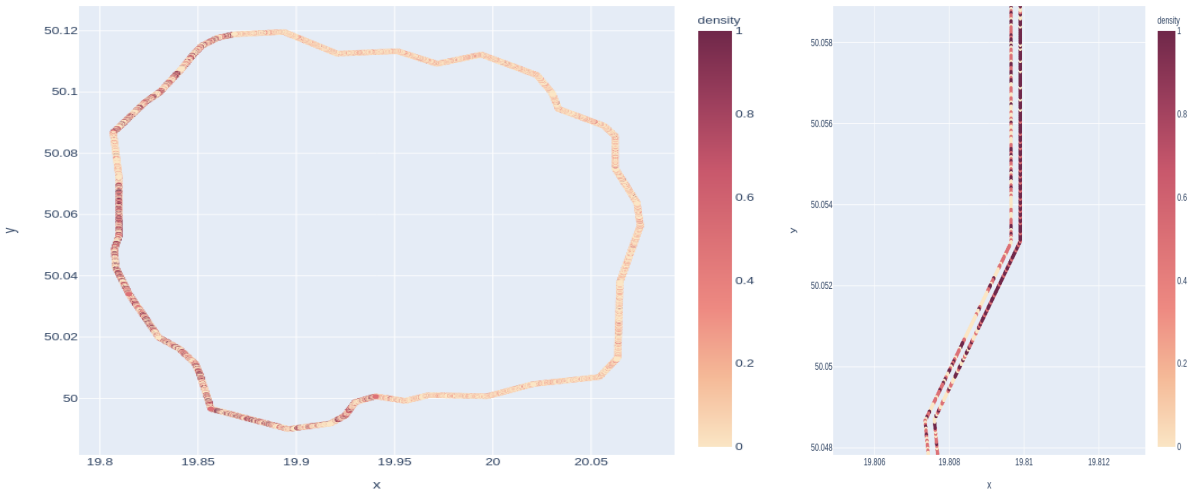
W przypadku ruchu bez utrudnień widać, że największe zagęszczenie pojazdów jest na południowo-zachodniej części obwodnicy, a najmniejsze na północno-wschodniej.



Rysunek 9: Ruch bez utrudnień.

7.2 Zwężenie

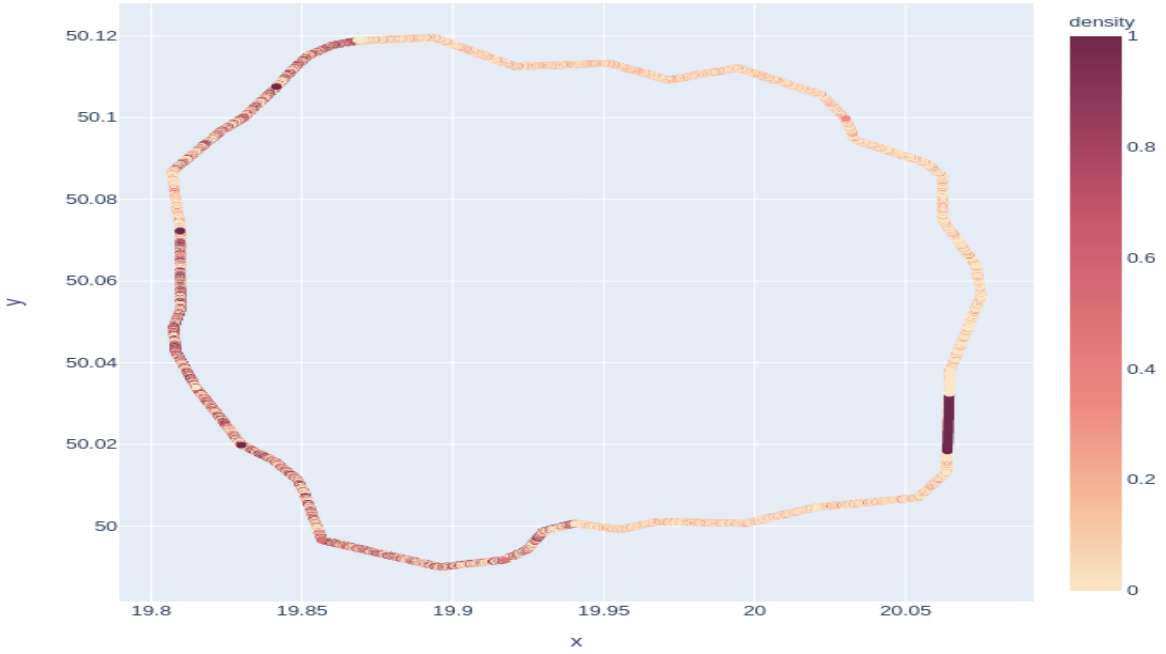
Po zwężeniu na odcinku Bielany-Balice i ograniczeniu prędkości, szybko powstaje korek, co widać no rysunku. utrudnienie wpływa na ruch przed zwężeniem, zatrzymując pojazdy długo przed ograniczeniem szerokości drogi.



Rysunek 10: Ruch po zwężeniu na odcinku Bielany-Balice.

7.3 Blokada

Stworzenie blokady spowodowało stworzenie długiej kolumny pojazdów nie mogących się ruszyć.



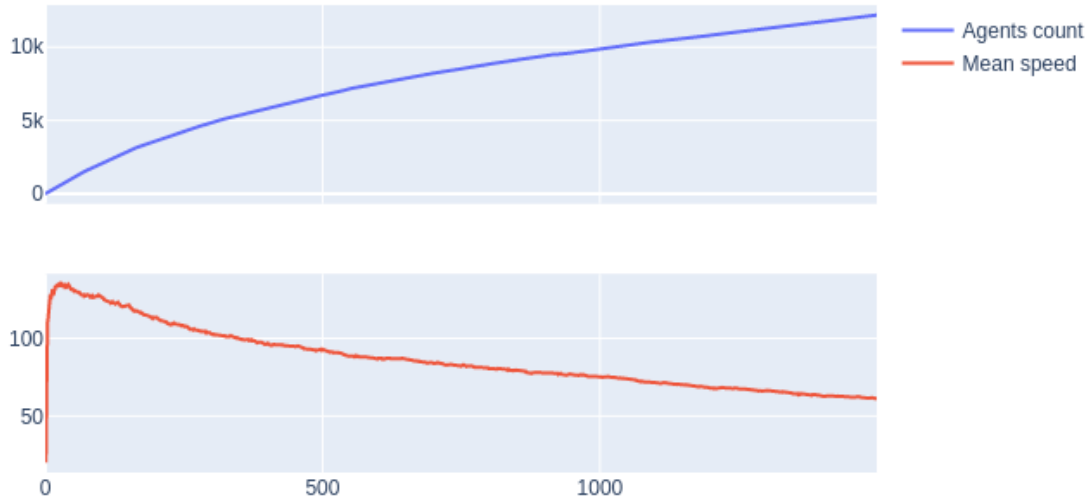
Rysunek 11: Blokada drogi na odcinku Bieżanów-Przewóz.

7.4 Statystyki

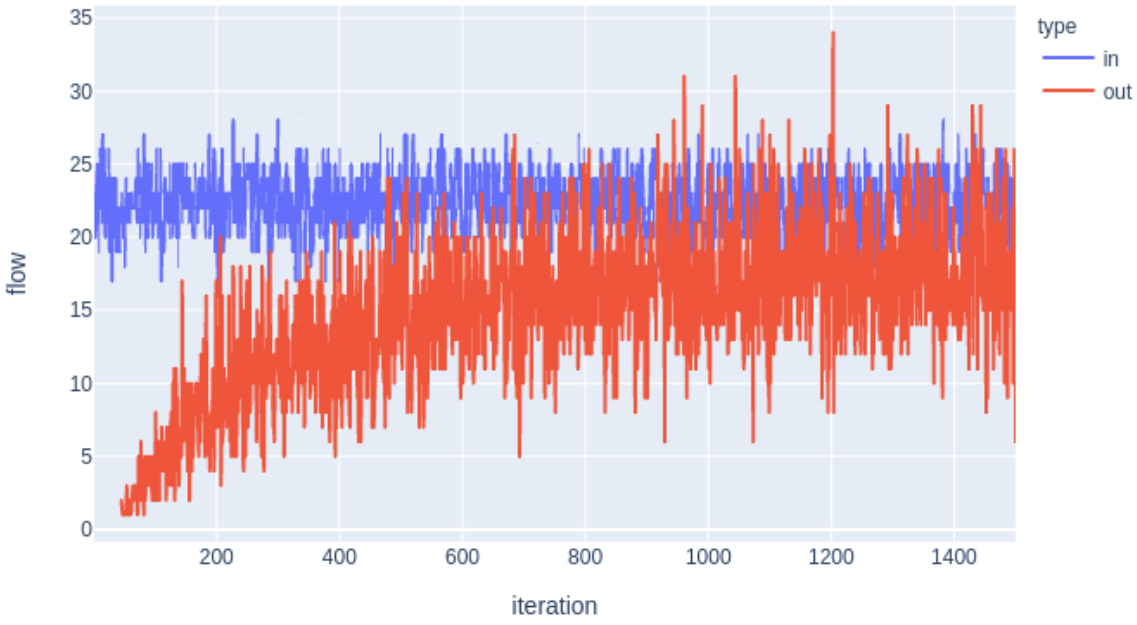
Uzyskane statystyki wskazują na zmniejszanie się średniej prędkości pojazdów, im więcej pojazdów znajduje się na obwodnicy. Początkowo samochody poruszały się najszybciej, kiedy na drodze nie było utrudnień i było ich najmniej.

Efekty zwężenia i blokady na pewno zmniejszyły średnią prędkość znacznie. Nie widać tego jako skok, ale płynne zmniejszenie się wartości, co oznacza, że efekty utrudnień ruchu są widoczne dopiero po upływie pewnego czasu.

Przepływ na węźle Przewóz uległ wyraźnemu pogroszeniu po zablokowaniu drogi.



Rysunek 12: Statystyki pojazdów po symulacji wszystkich wariantów.

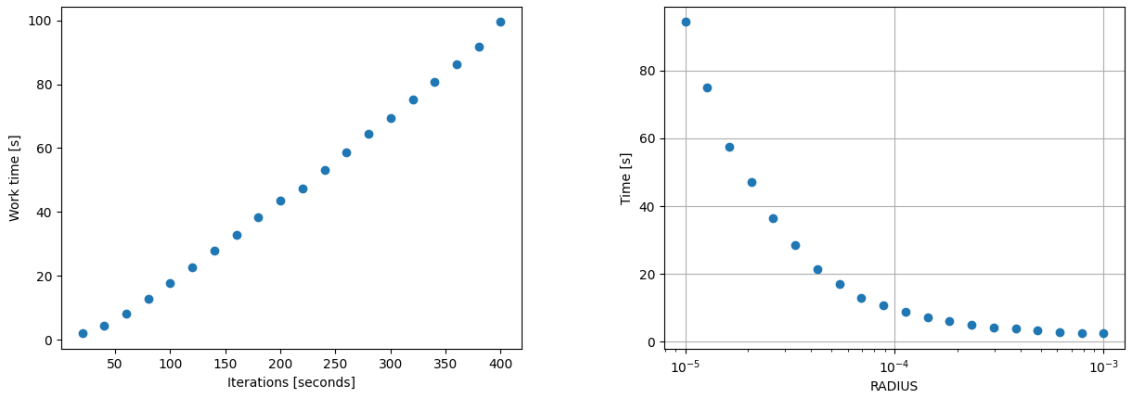


Rysunek 13: Sumaryczny przepływ.

7.5 Wydajność

Wydajność naszego algorytmu zmierzyliśmy w zależności od czasu symulacji i gęstości komórek, czyli parametru RADIUS definiującego odległość między komórkami.

Uzyskane wyniki wskazują na zależność bliską liniowej dla wybranej liczby iteracji, natomiast eksponencjalnie rośnie złożoność w przypadku zwiększenia gęstości komórek.



Rysunek 14: Wykresy czasów obliczeń w stosunku do parametrów symulacji. Skala logarytmiczna na wykresie z prawej.

8 Od autorów

Nasz framework umożliwa w obecnym stanie analizę interesującego parametru: średniego czasu jaki samochód spędza na obwodnicy. Postanowiliśmy zostawić to jako propozycję do samodzielnego eksperymentu z tą symulacją.

W czasie pracy napotkaliśmy duże trudności w implementacji mapy. Chcieliśmy wykorzystać istniejące, dostępne mapy, co okazało się być bardzo trudne w implementacji. Straciliśmy kilka tygodni tylko na szukanie rozwiązań umożliwiających wykorzystanie tego formatu. Ostatecznie stworzyliśmy własny, prosty format mapy, który przygotowaliśmy ręcznie na podstawie uzyskanych już wcześniej danych na temat obwodnicy.

Początkowo podczas projektu stosowana była metodologia Test Driven Development (TDD), jednak tylko do czasu prac nad OpenStreetMap, z którego zrezygnowaliśmy. Ograniczenia czasowe zmusiły nas do zrezygnowania z metody, aby ograniczyć ilość pisanego kodu. Mimo to, uważamy, że skutecznie pracuje się w ten sposób i ma on potencjał do zastosowania w przyszłych projektach.

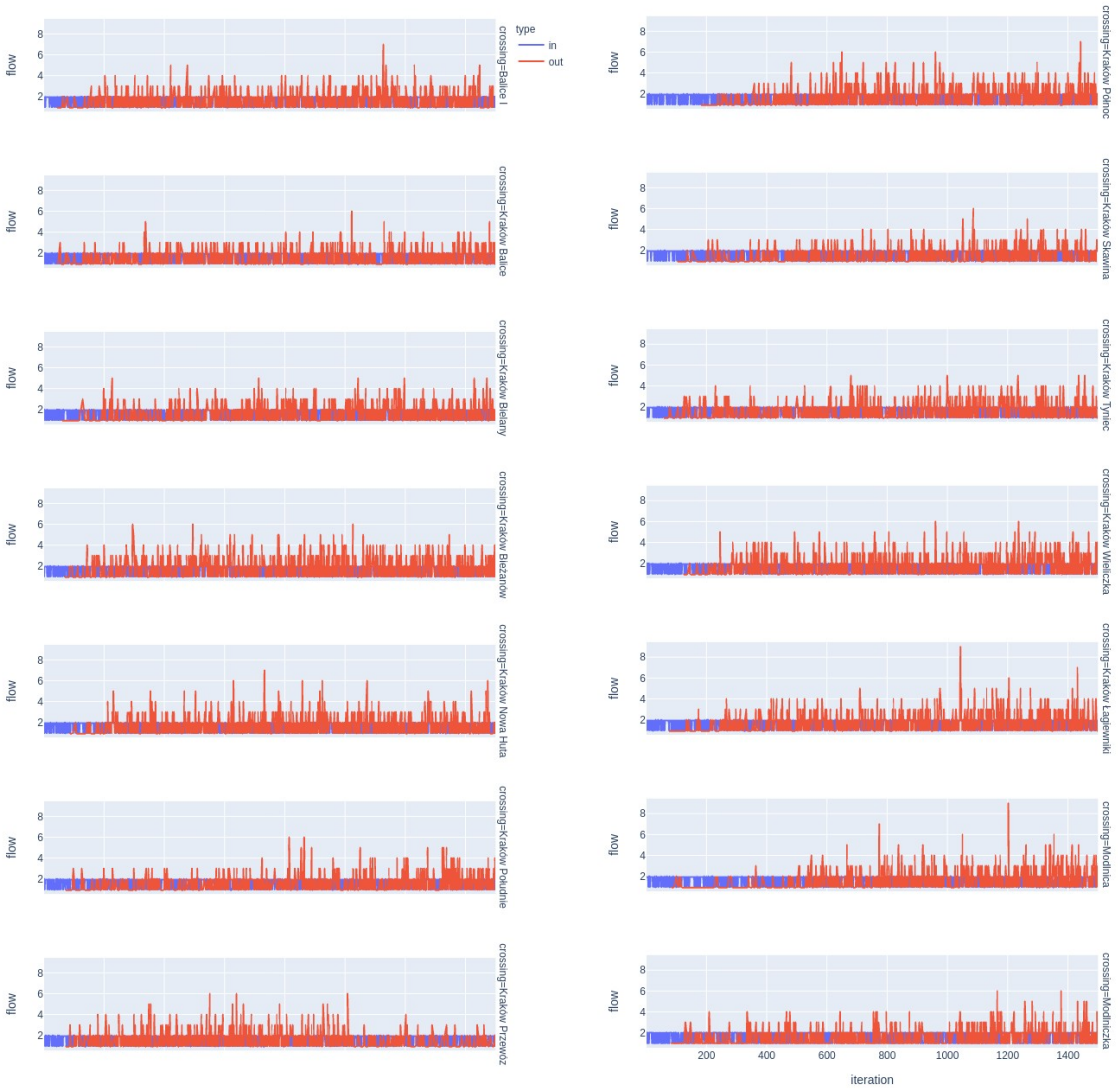
9 Dalsza praca

Aby uczynić projekt bardziej uniwersalnym, należy dodać obsługę innych zaawansowanych map. Można to zrobić poprzez rozbudowanie obecnego modułu `SimpleMap` lub zaimplementować obsługę znanych już formatów, np. OpenStreetMap, Google MyMaps.

Projekt nie posiada innych pojazdów, niż domyślny osobowy. Zatem kolejnym usprawnieniem jest wprowadzenie pojazdów, które:

- obowiązują inne ograniczenia prędkości
- zajmują więcej, niż jedną komórkę

Obsługa całej platformy może zostać zawarta w wygodnej dla użytkownika oprawie graficznej, np. `kivy`. Na chwilę obecną najbardziej interaktywny interfejs uzyskujemy poprzez `jupyter notebook`, który ma duże ograniczenia wydajności.



Rysunek 15: Przepływ w zależności od skrzyżowania.

Literatura

[1] Wiki, *Kraków Obwodnica IV*, `wikipedia.org`

[2] OpenStreet, *mapa*, `openstreetmap.org`

[3] K. Nagel, M. Schreckenberg, *Two lane traffic simulations using cellular automata*, `arxiv.org`

[4] Plotly, *interaktywne wykresy*, `plotly.com`