ZWiAD

SGH

Agent-based simulation modelling of out-of-home advertising viewing opportunity

Wprowadzenie

- Celem projektu było stworzenie narzędzia umożliwiającego przeprowadzenie symulacji wieloagentowych ruchu drogowego na wielką skalę.
- Cechą szczególną zaproponowanego modelu jest wykorzystanie rzeczywistych danych socjodemograficznych, dzięki czemu możliwe jest bliskie rzeczywistości odwzorowanie wzorców przemieszczania się mieszkańców danego obszaru oraz zdobycie informacji na temat profilu demograficznego osób przejeżdżających przez poszczególne części miasta.

Wprowadzenie

- Motywacją do rozpoczęcia tego projektu był problem efektywnego rozmieszczenia reklam zewnętrznych (bilboardów, słupów ogłoszeniowych) na terenie miasta.
- W przypadku reklamy kluczowe jest dodarcie do grupy docelowej, która jest zainteresowana zakupem produktu.
- Jednak zdobycie danych na temat profili demograficznych osób poruszających się po danym obszarze jest trudne i kosztowne.
- W takiej sytuacji symulacja jest efektywnym narzędziem pozwalającym na zdobycie tego typu informacji.

https://github.com/pszufe/OpenStreetMapX.jl

OpenStreetMapX

- Biblioteka stworzona w celu umożliwienia efektywnej pracy na plikach OpenStreetMap (rozszerzenie .osm) i ich integrację z narzędziami udostępnianymi przez Google Maps.
- Zgodna z Julia 1.0.0.
- Zoptymalizowana pod kątem udostępnienia użytkownikowi możliwie jak największej gamy narzędzi przy zachowaniu wysokiej wydajności.

https://github.com/pszufe/OpenStreetMapXSim.jl

OpenStreetMapXSim

- Biblioteka będąca głównym „rdzeniem” symulacji.
- Opiera się na wymienionych powyżej założeniach, jednak dzięki swojej konstrukcji pozwala na relatywnie łatwe ich modyfikowanie i implementowanie zupełnie nowych modeli.
- Bazowo posiada wbudowaną obsługę obliczeń rozproszonych, dzięki czemu możliwe jest dalsze przyśpieszenie działania kodu.

Opis modelu

Start location selector *function*

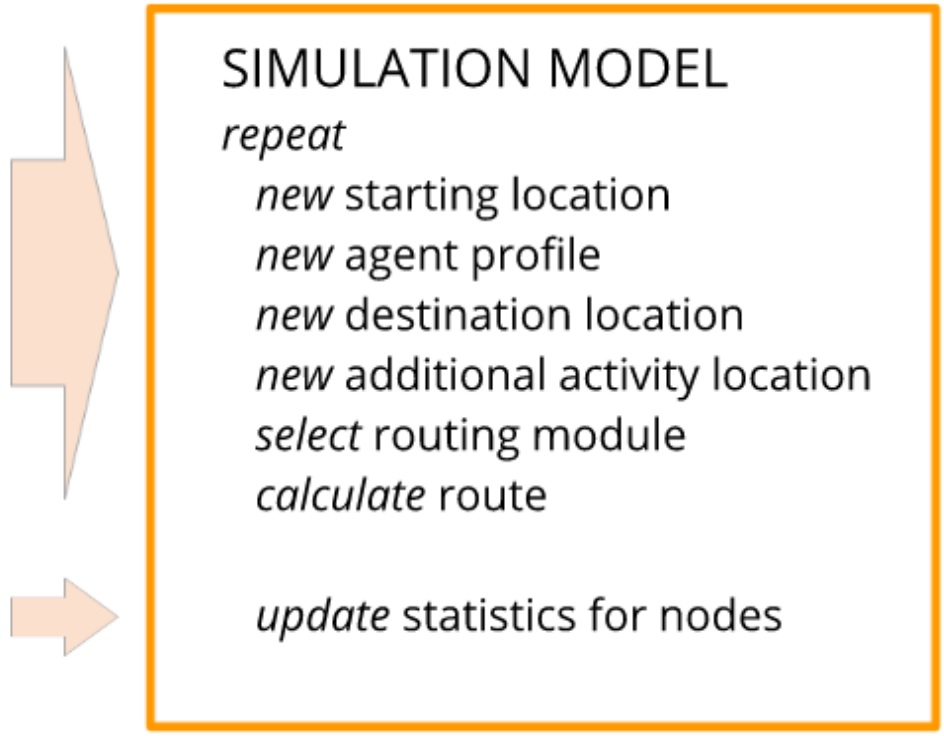
Agent profile generator *function*

Destination location selector *function*

Additional activity selector *function*

Routing module *function*
shortest / fastest / GoogleMapsAPI

Statistics aggregator *function*

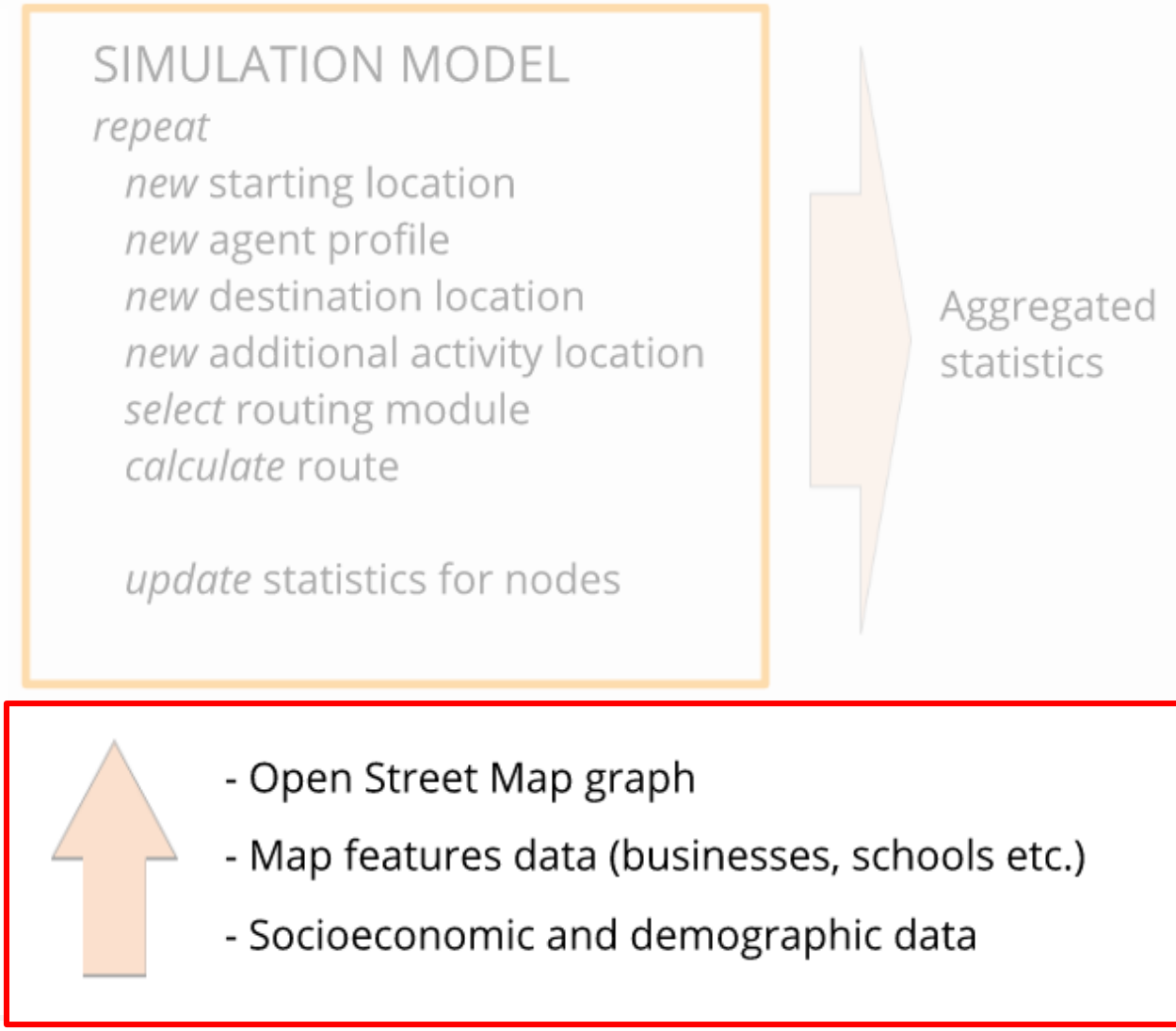


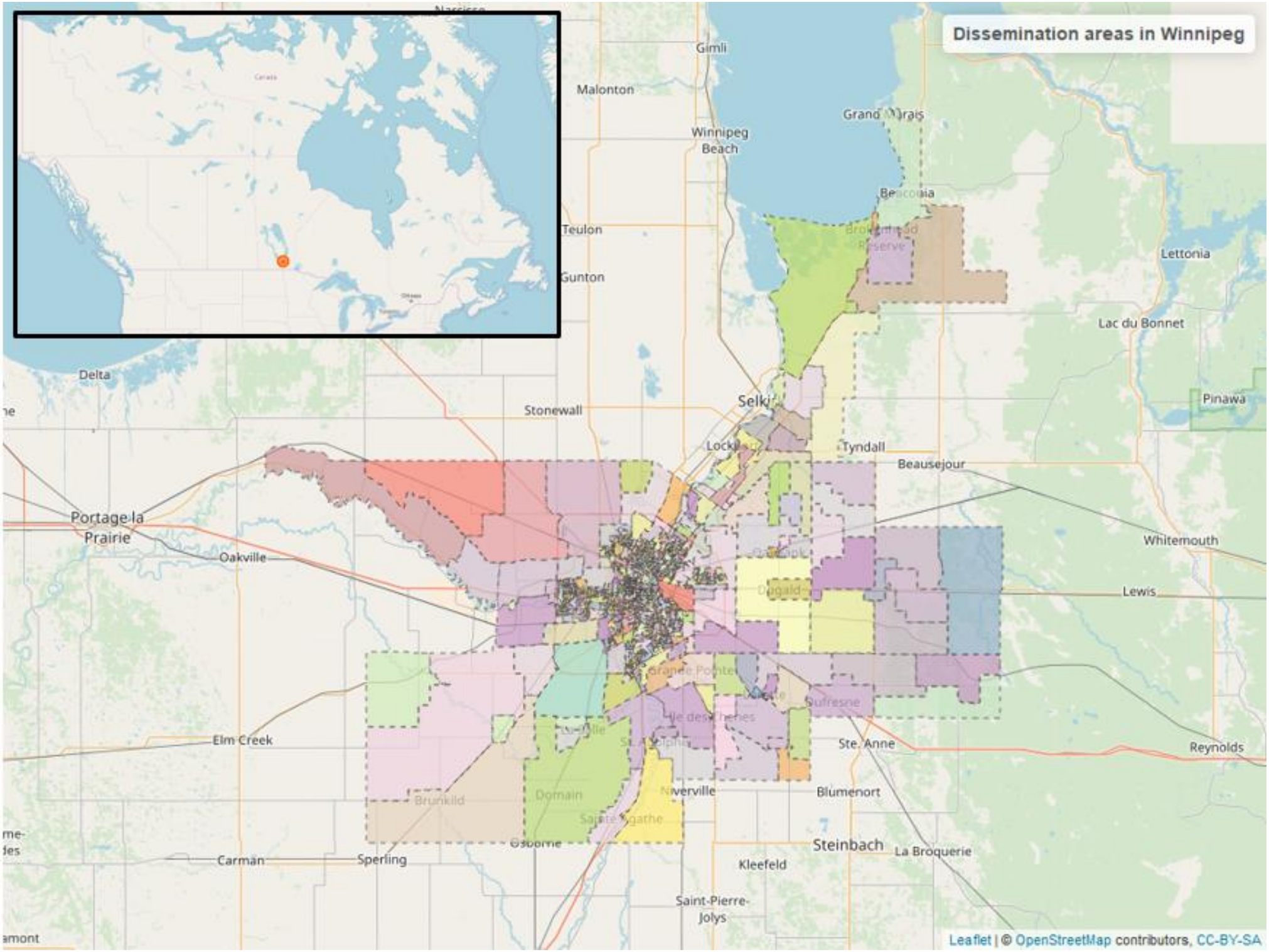
Aggregated statistics

- Open Street Map graph
- Map features data (businesses, schools etc.)
- Socioeconomic and demographic data

Opis modelu

- Do poprawnego działania symulacji konieczne jest zebranie 3 rodzajów danych:
  - Danych mapowych (plik .osm)
  - Danych z informacjami na temat rozmieszczenia obiektów na mapie (szkół, sklepów, firm etc.)
  - Danych demograficznych dotyczących mieszkańców odpowiednio wyznaczonych stref *(dissemination areas - DAs)*




Dissemination areas in Winnipeg
Gimli
Malonton
Winnipeg Beach
Grand Marais
Teulon
Gunton
Lettonia
Lac du Bonnet
Pinawa
Delta
Stonewall
Selkirk
Tyndall
Beausejour
Portage la Prairie
Oakville
Whitemouth
Lewis
Elm Creek
Ste. Anne
Reynolds
Blumenort
Steinbach
La Broquerie
Carman
Sperling
Kleefeld
Saint-Pierre-Jolys
Leaflet | © OpenStreetMap contributors, CC-BY-SA

OSM

- **OpenStreetMap (OSM)** – open source'owy projekt mający na celu utworzenie darmowej i ogólnodostępnej mapy świata.
- W rzeczywistości jest to baza danych przechowująca dokładne dane na temat obiektów i struktur występujących na powierzchni Ziemi.

OSM

- **OpenStreetMap (OSM)** – open source'owy projekt mający na celu utworzenie darmowej i ogólnodostępnej mapy świata.
- W rzeczywistości jest to baza danych przechowująca dokładne dane na temat obiektów i struktur występujących na powierzchni Ziemi.
- OSM opiera się na czterech podstawowych **typach prymitywnych**:
  - **Węzły** *(nodes)* – pojedyncze punkty określone za pomocą ich współrzędnych geograficznych.
  - **Linie** *(ways)* – uporządkowane listy węzłów służące do opisania polilinii (linii łamanych) i wielokątów (np. dróg, budynków).
  - **Relacje** *(relations)* – uporządkowane listy zawierające w sobie zarówno węzły, linie jak i inne relacja (nazywane członkami *(members)*), które dodatkowo mogą mieć przypisaną rolę *(role)*. Służą do opisywania złożonych elementów przestrzeni, np. autostrad czy też linii autobusowych.
  - **Tagi** *(tags)* – para klucz-wartość służąca do przechowywania metadanych (charakterystyk – *features*) powyższych elementów mapy. Posiada określony sposób kodowania.

OSM

- **OpenStreetMap (OSM)** – open source'owy projekt mający na celu utworzenie darmowej i ogólnodostępnej mapy świata.
- Podstawowym sposobem przechowywania danych osm jest baza danych **PostgreSQL.**
- Najczęściej wykorzystywanym formatem do pracy z danymi osm (w tym w omawianej bibliotece) jest format **XML**.

Opis modelu

- Na podstawie danych mapowych i informacji na temat położenia obiektów na mapie tworzona jest reprezentacja miasta w formie grafu skierowanego.
- Każdy węzeł grafu odpowiada jednemu skrzyżowaniu, wagi krawędzi oznaczają odległości pomiędzy skrzyżowaniami.

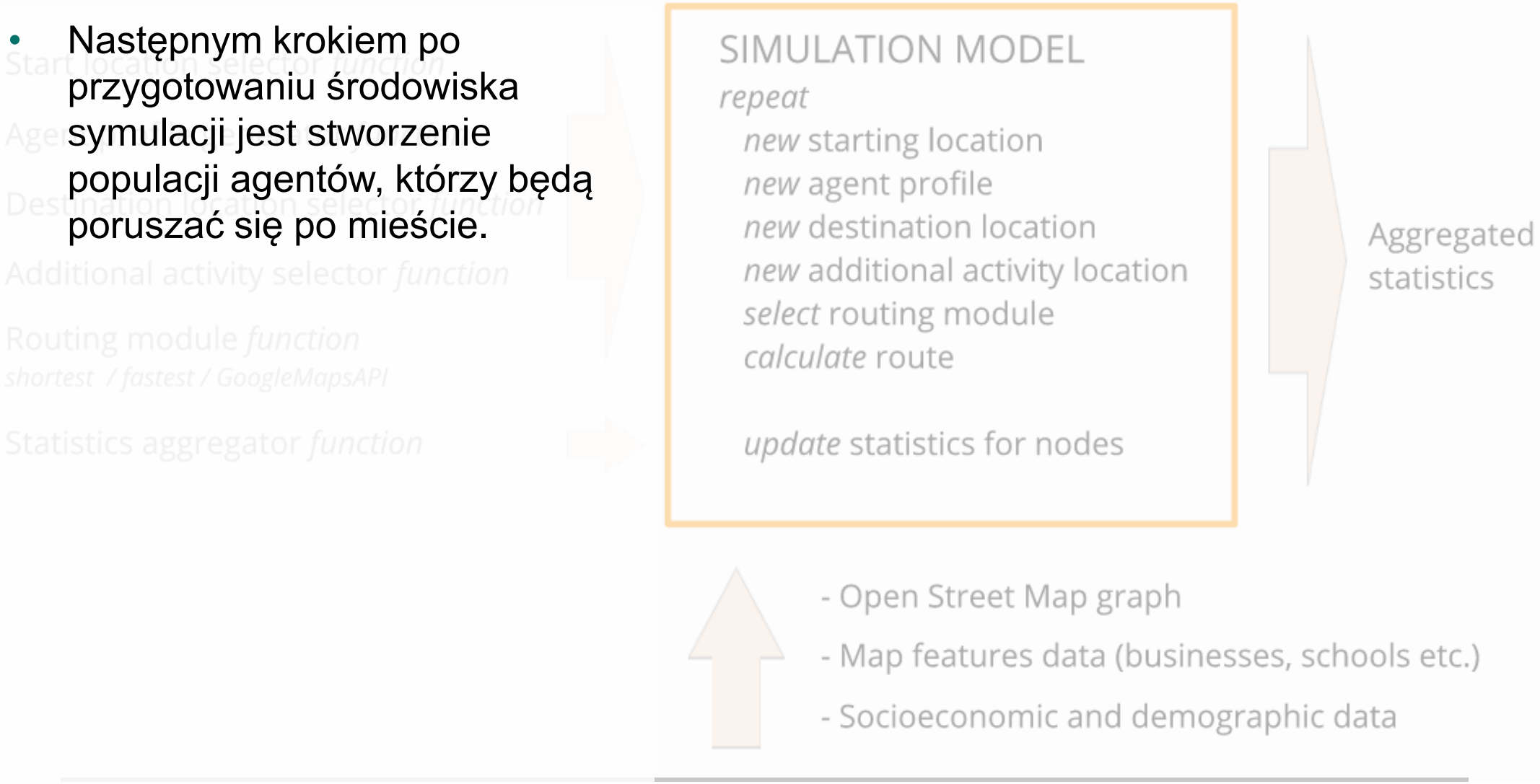
Opis modelu

- Na podstawie danych mapowych i informacji na temat położenia obiektów na mapie tworzona jest reprezentacja miasta w formie grafu skierowanego.
- Każdy węzeł grafu odpowiada jednemu skrzyżowaniu, wagi krawędzi oznaczają odległości pomiędzy skrzyżowaniami.

Źródło: http://www.marcinkossakowski.com/finding-shortest-path-using-dijkstras-algorithm/

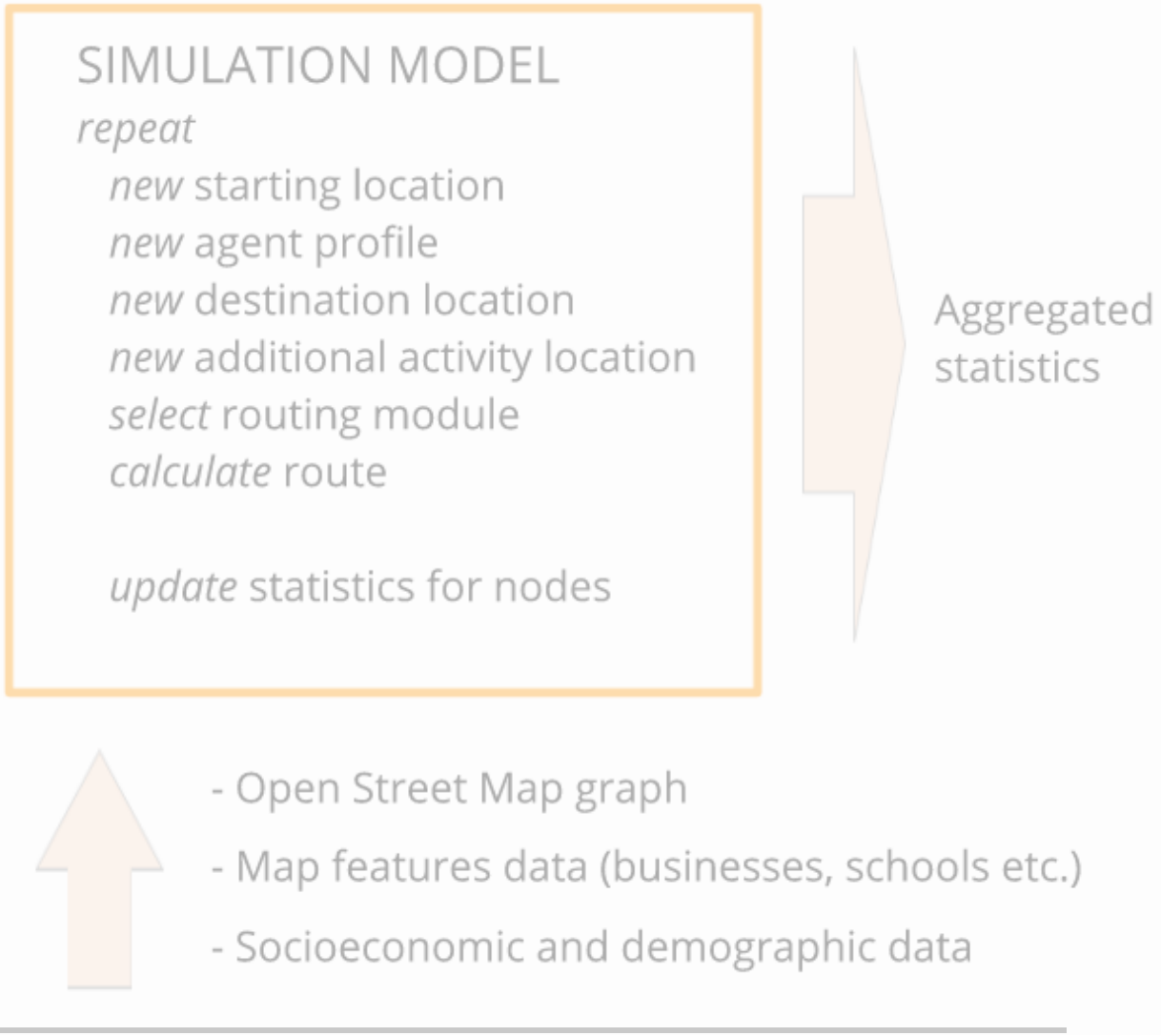
Opis modelu

- Następnym krokiem po przygotowaniu środowiska symulacji jest stworzenie populacji agentów, którzy będą poruszać się po mieście.



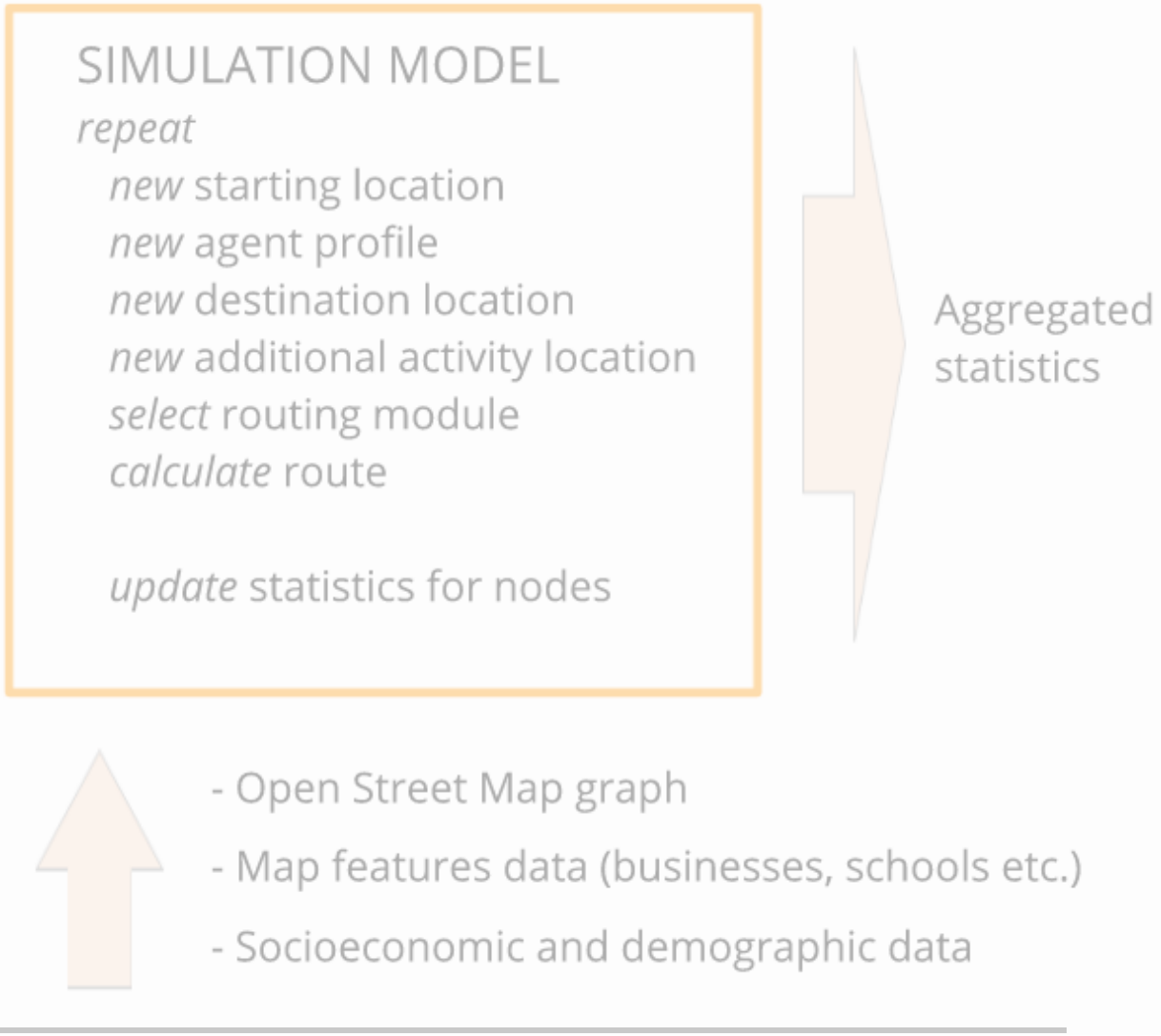
Opis modelu

- Następnym krokiem po przygotowaniu środowiska symulacji jest stworzenie populacji agentów, którzy będą poruszać się po mieście.
- Założenia co do ich zachowania są proste. Agenci jeżdżą z domu do pracy i z pracy do domu.

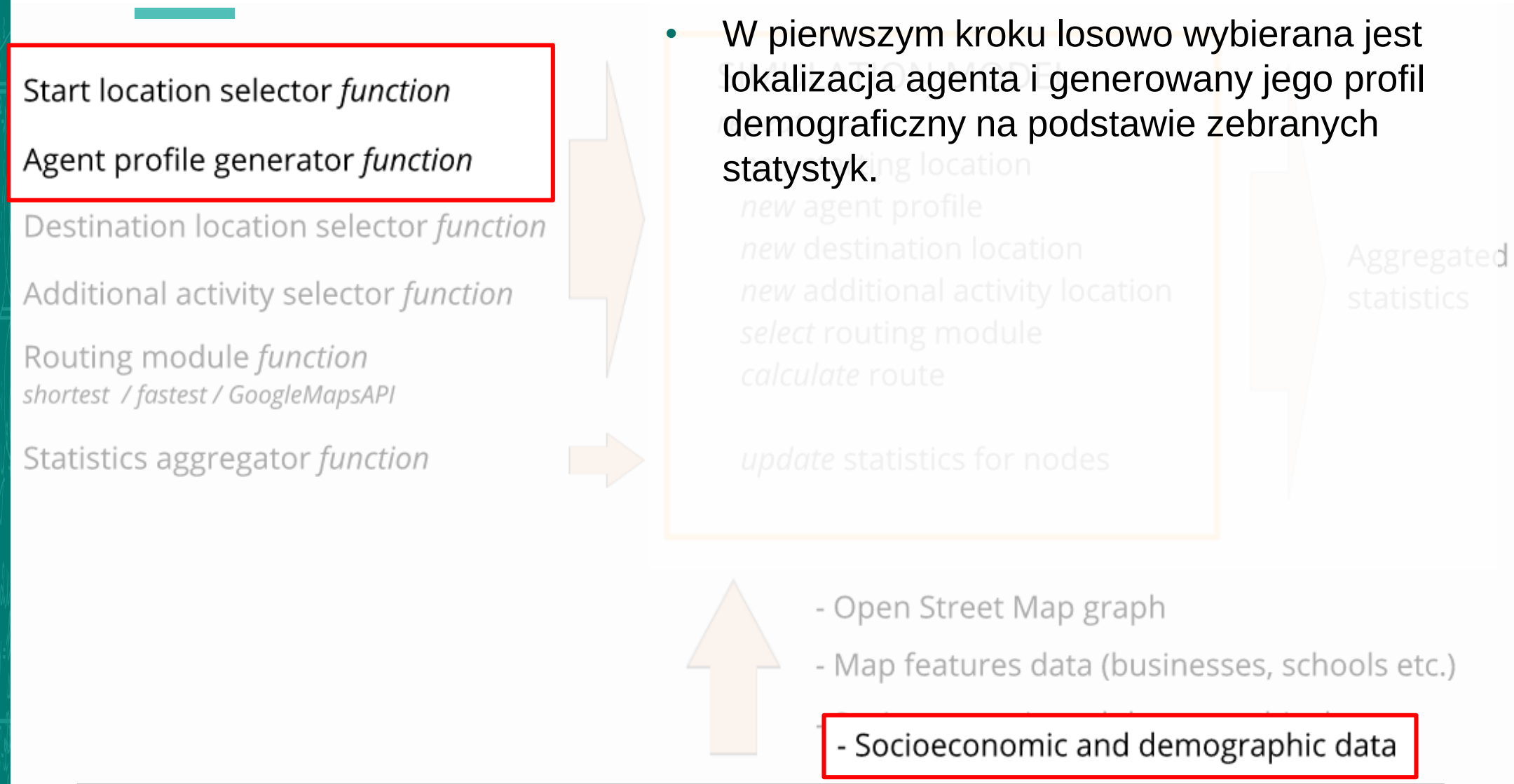


Opis modelu

- Następnym krokiem po przygotowaniu środowiska symulacji jest stworzenie populacji agentów, którzy będą poruszać się po mieście.
- Założenia co do ich zachowania są proste. Agenci jeżdżą z domu do pracy i z pracy do domu.
- Podczas pokonywania jednej z tras mogą dodatkowo pojechać do jednego pośredniego punktu (odwieźć dzieci do szkoły, zrobić zakupy, etc.).

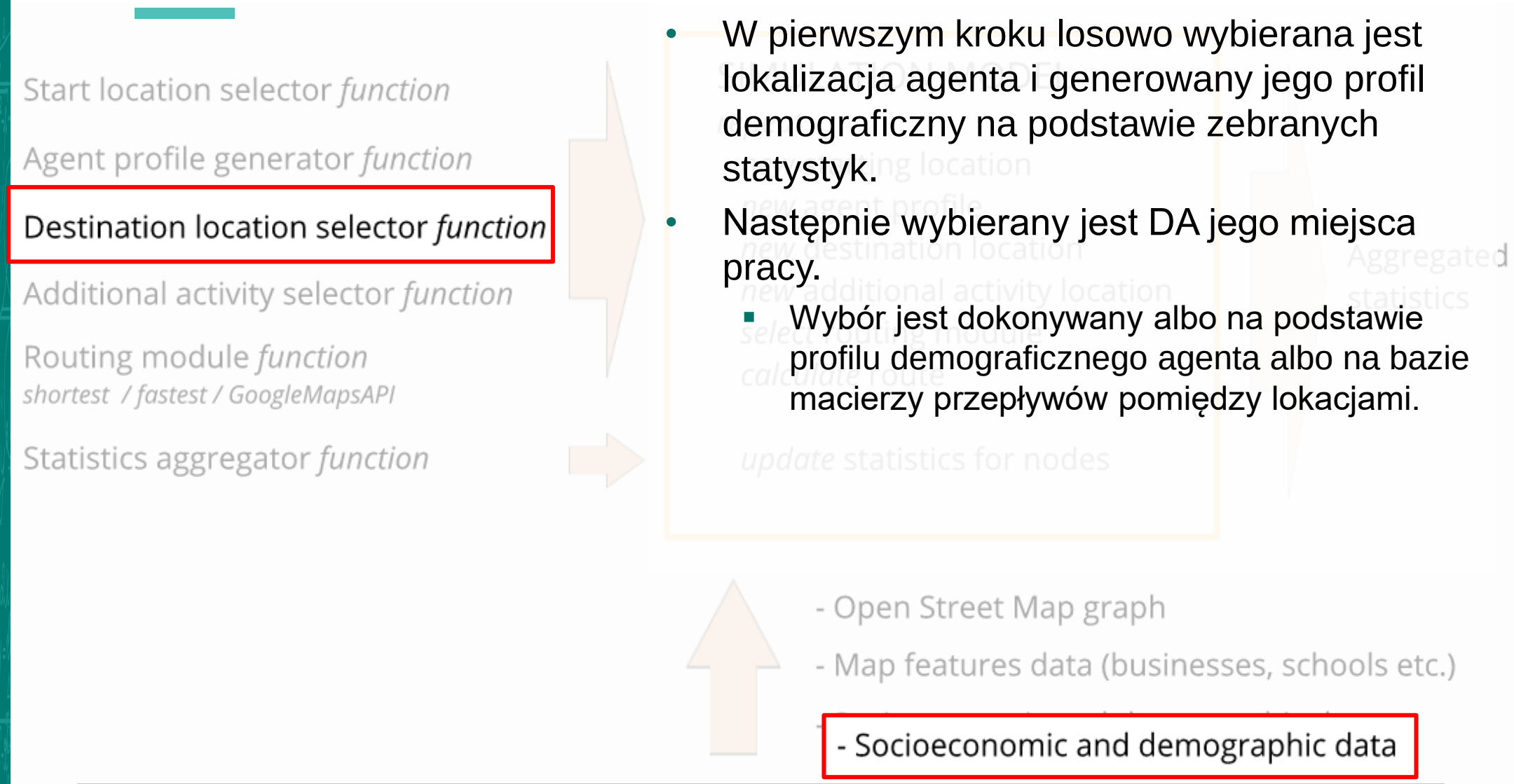


Opis modelu



- W pierwszym kroku losowo wybierana jest lokalizacja agenta i generowany jego profil demograficzny na podstawie zebranych statystyk.

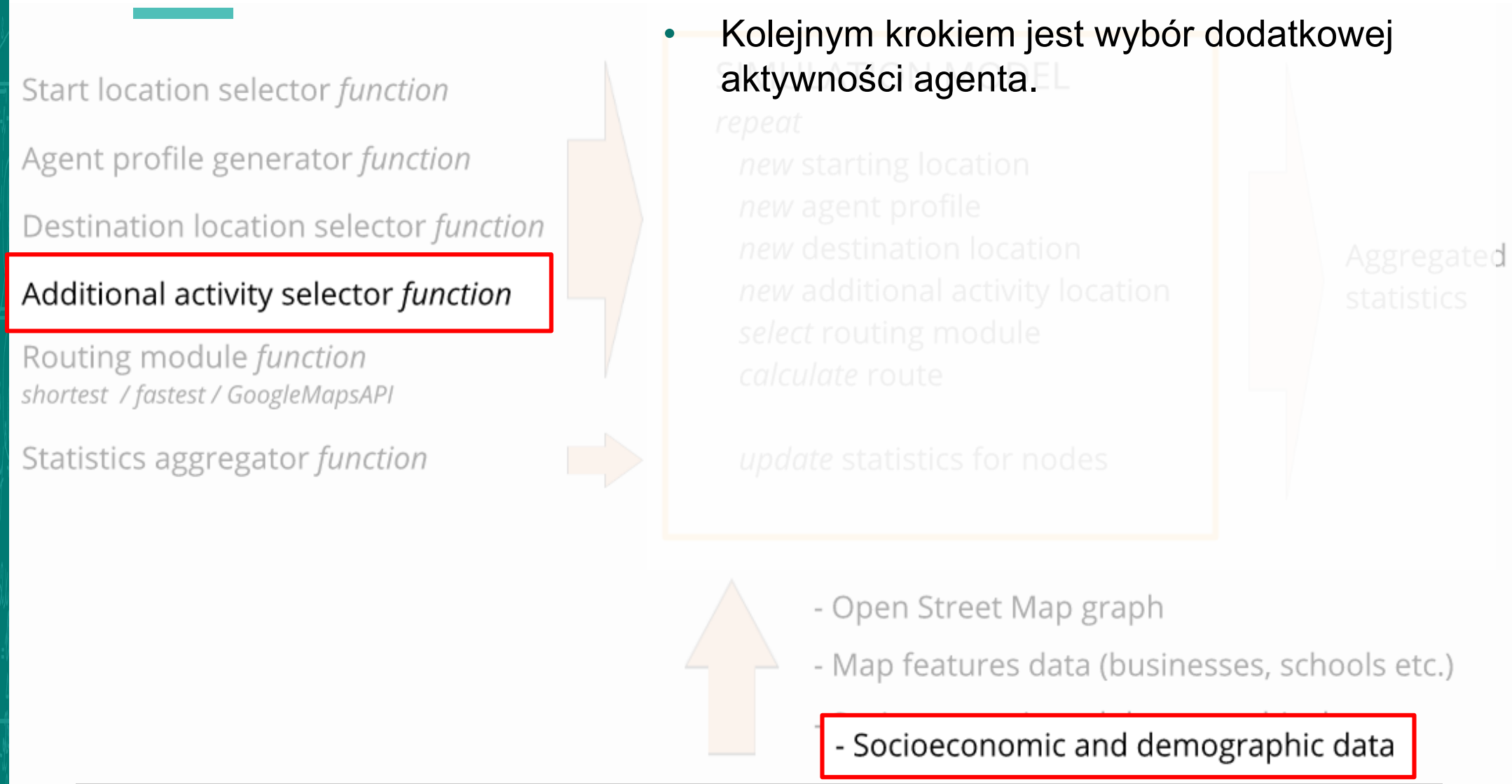
Opis modelu



- W pierwszym kroku losowo wybierana jest lokalizacja agenta i generowany jego profil demograficzny na podstawie zebranych statystyk.
- Następnie wybierany jest DA jego miejsca pracy.
  - Wybór jest dokonywany albo na podstawie profilu demograficznego agenta albo na bazie macierzy przepływów pomiędzy lokacjami.

Opis modelu

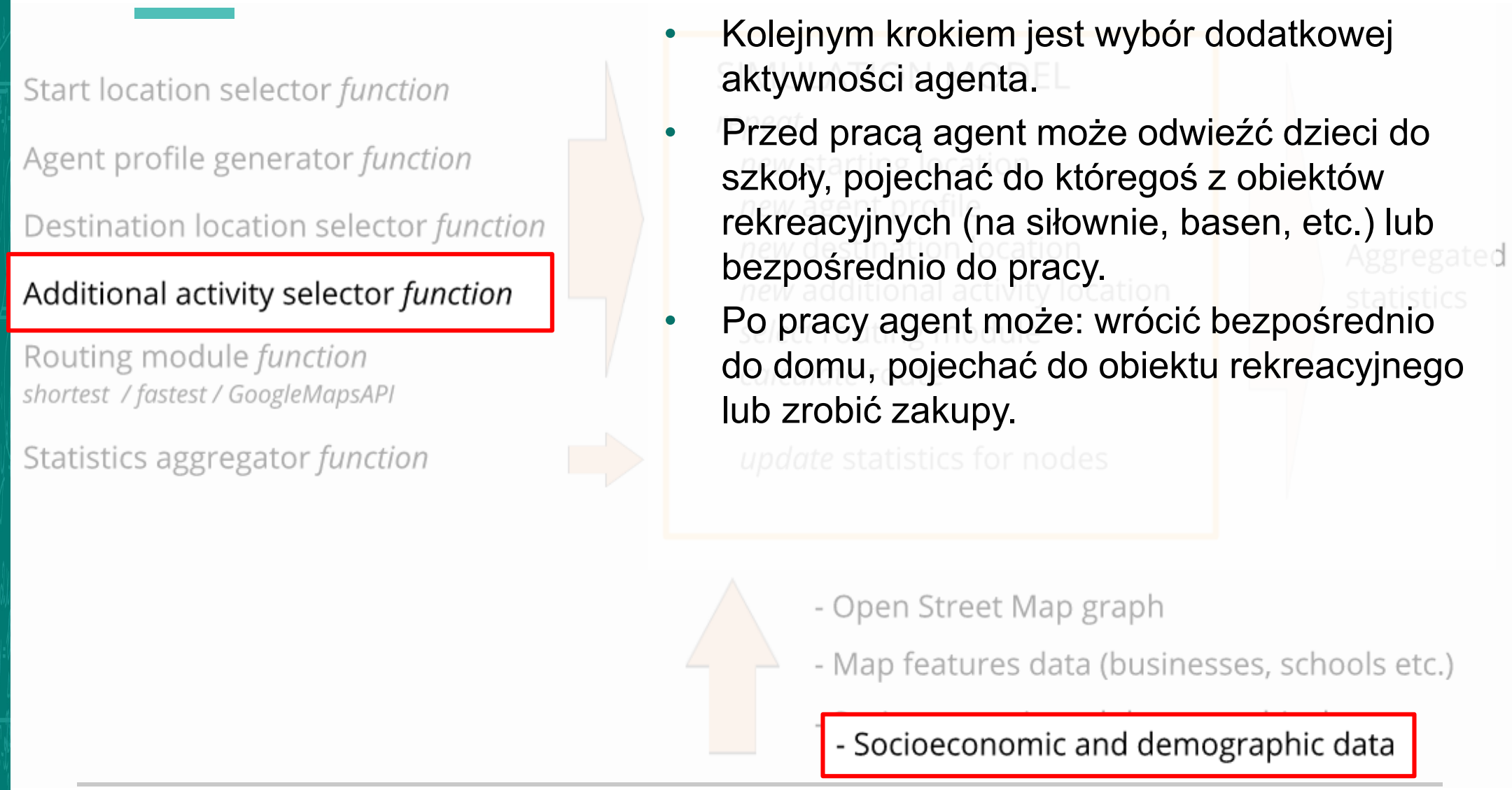
- Kolejnym krokiem jest wybór dodatkowej aktywności agenta.



Opis modelu

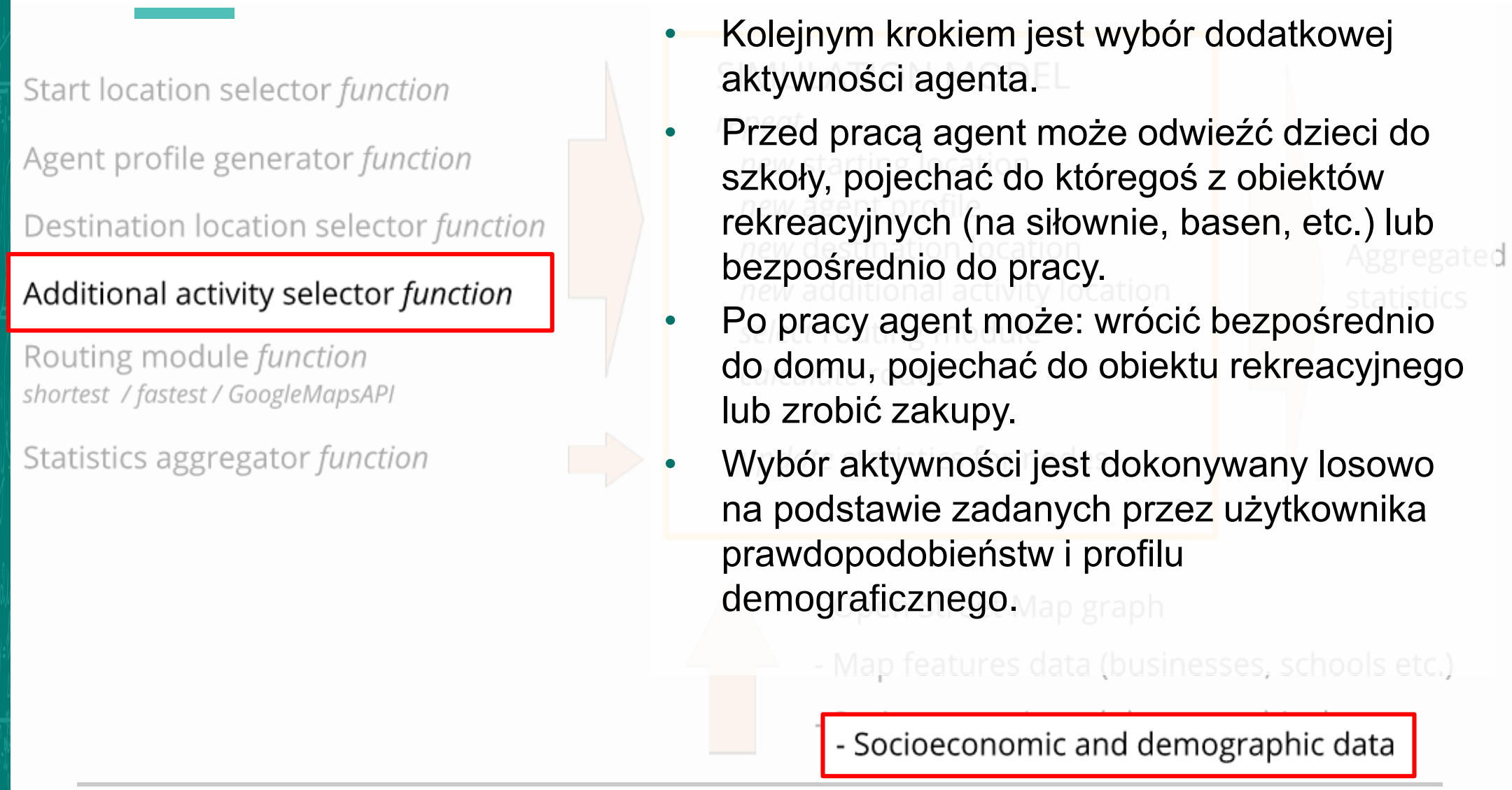
- Kolejnym krokiem jest wybór dodatkowej aktywności agenta.
- Przed pracą agent może odwieźć dzieci do szkoły, pojechać do któregoś z obiektów rekreacyjnych (na siłownie, basen, etc.) lub bezpośrednio do pracy.

Opis modelu



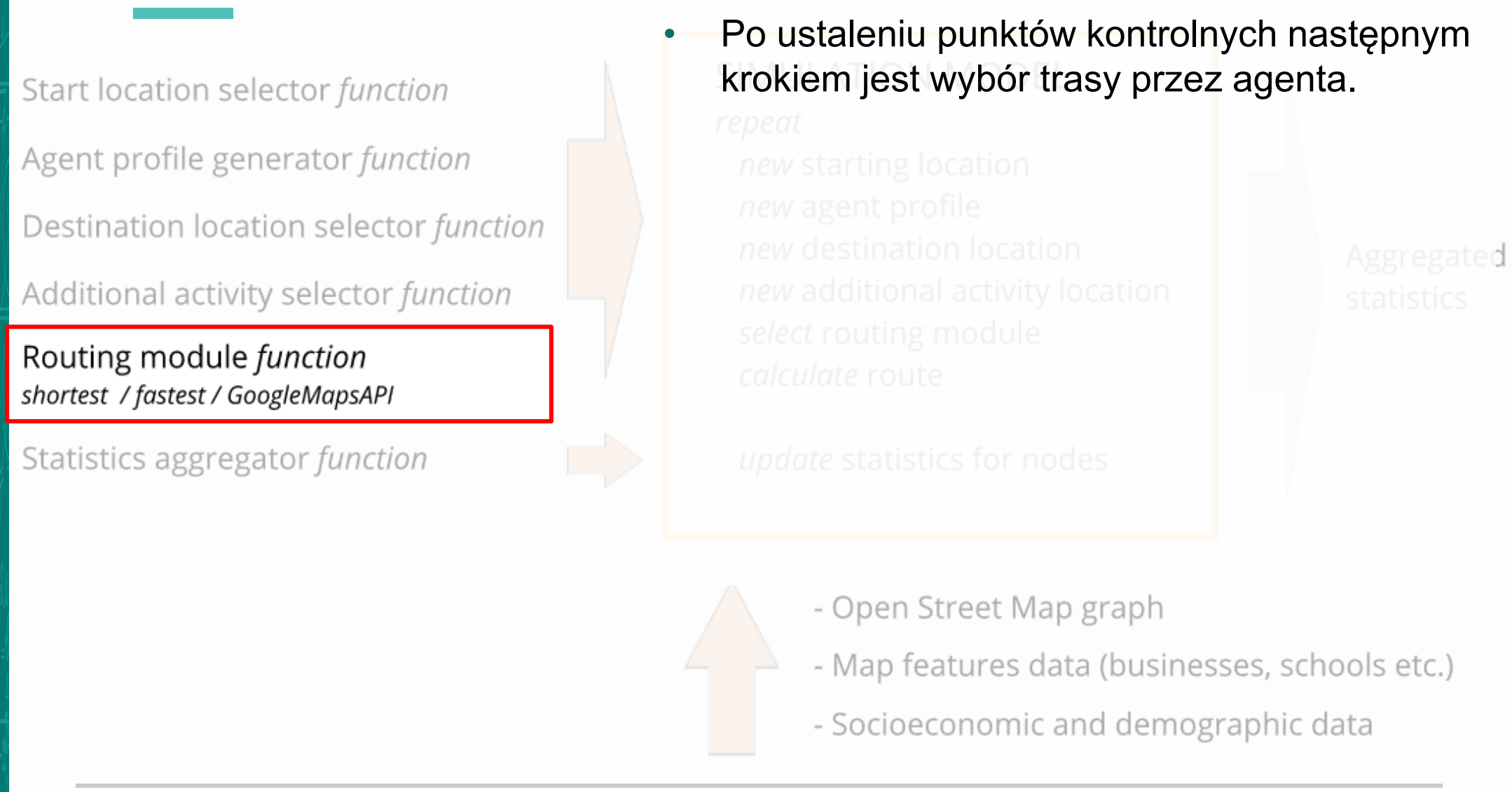
- Kolejnym krokiem jest wybór dodatkowej aktywności agenta.
- Przed pracą agent może odwieźć dzieci do szkoły, pojechać do któregoś z obiektów rekreacyjnych (na siłownie, basen, etc.) lub bezpośrednio do pracy.
- Po pracy agent może: wrócić bezpośrednio do domu, pojechać do obiektu rekreacyjnego lub zrobić zakupy.

Opis modelu



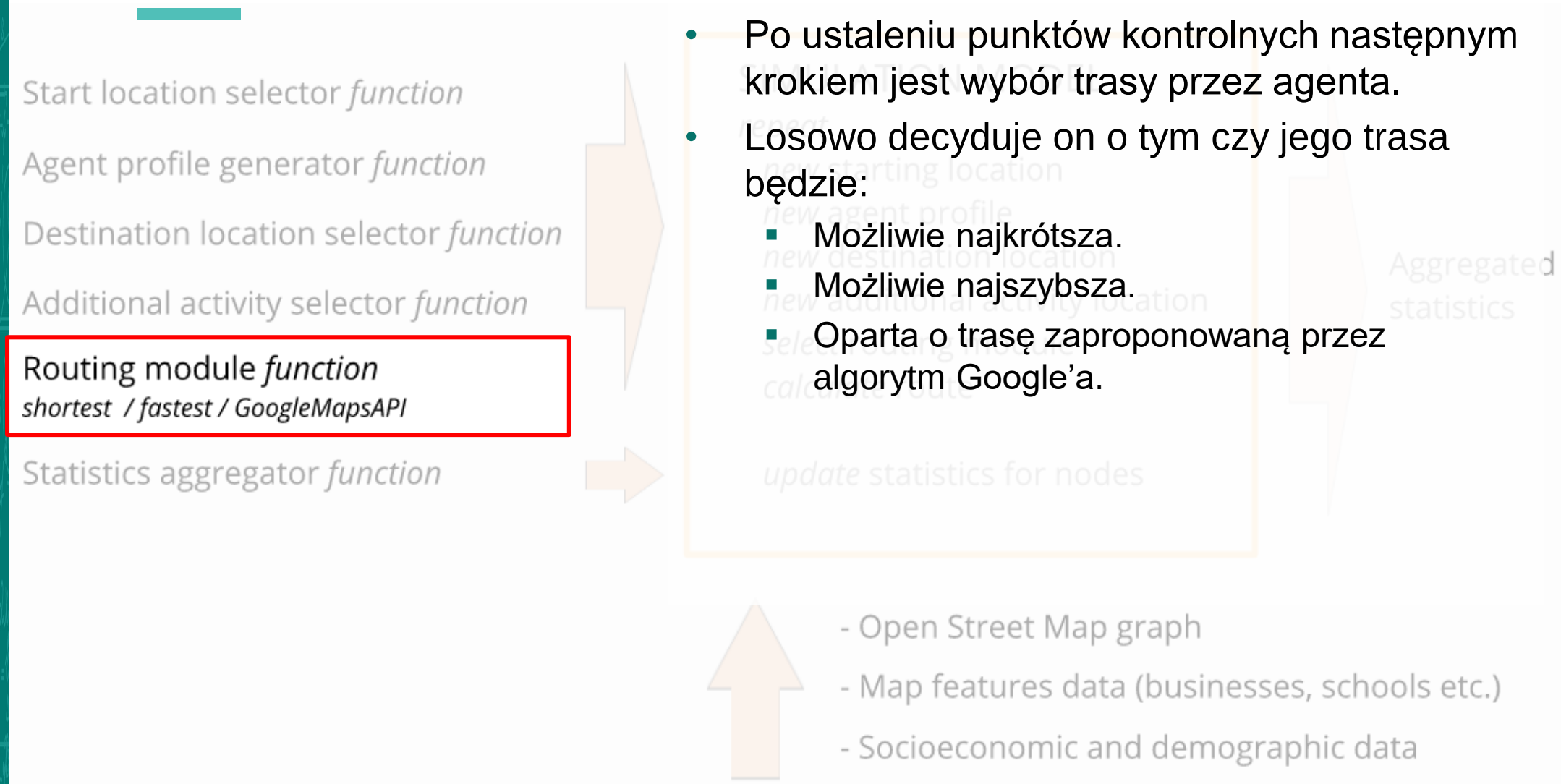
- Kolejnym krokiem jest wybór dodatkowej aktywności agenta.
- Przed pracą agent może odwieźć dzieci do szkoły, pojechać do któregoś z obiektów rekreacyjnych (na siłownie, basen, etc.) lub bezpośrednio do pracy.
- Po pracy agent może: wrócić bezpośrednio do domu, pojechać do obiektu rekreacyjnego lub zrobić zakupy.
- Wybór aktywności jest dokonywany losowo na podstawie zadanych przez użytkownika prawdopodobieństw i profilu demograficznego.

Opis modelu



- Po ustaleniu punktów kontrolnych następnym krokiem jest wybór trasy przez agenta.

Opis modelu

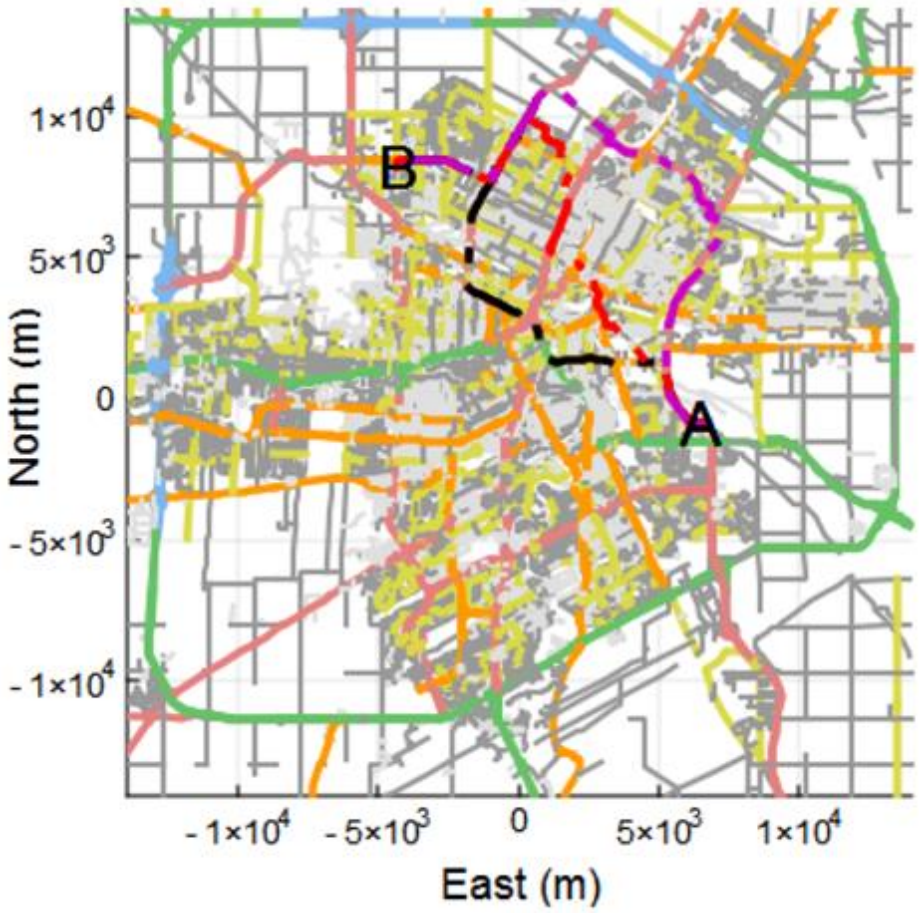


- Po ustaleniu punktów kontrolnych następnym krokiem jest wybór trasy przez agenta.
- Losowo decyduje on o tym czy jego trasa będzie:
  - Możliwie najkrótsza.
  - Możliwie najszybsza.
  - Oparta o trasę zaproponowaną przez algorytm Google'a.

Opis modelu

Przykładowe trasy wybrane za pomocą algorytmów zaimplementowanych w symulacji:

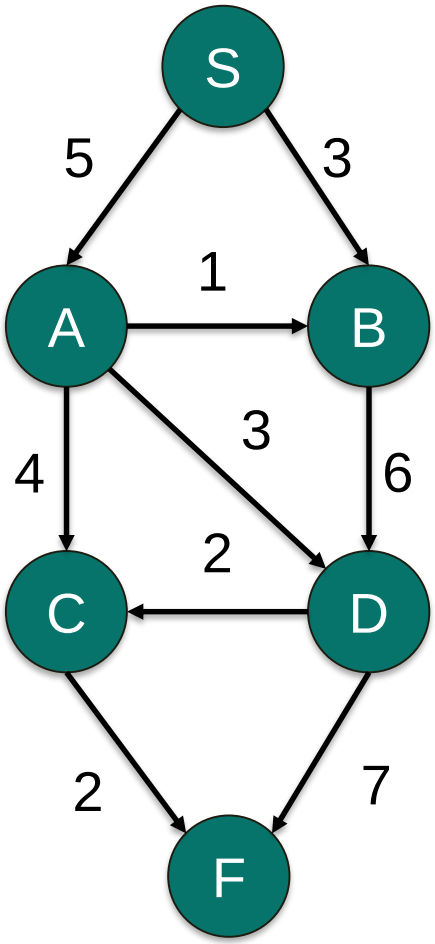
- Najkrótsza trasa
- Najszybsza trasa
- Trasa Google'a



Problem najkrótszej ścieżki

- Problem znalezienia najkrótszej lub najszybszej drogi pomiędzy punktami A i B da się sprowadzić do problemu najkrótszej ścieżki w grafie.
- Istnieje wiele sposobów rozwiązywania tego problemu, najpopularniejszy algorytmy to **algorytm Dijkstry.**

Algorytm Dijkstry



$$Odwiedzone = []$$
$$Nieodwiedzone = [S, A, B, C, D, F]$$

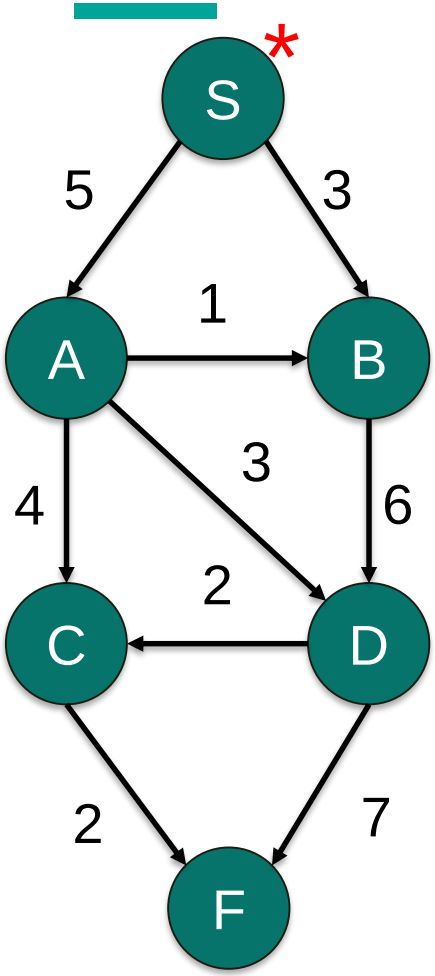
Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	∞	
B	∞	
C	∞	
D	∞	
F	∞	

Wygeneruj dwie listy:

- Odwiedzonych wierzchołków.
- Nieodwiedzonych wierzchołków

Przyjmij, że dla każdego $v \in V \backslash v_o$ odległość od wierzchołka początkowego v_o jest równa ∞, a dla $v_o = 0$.

Algorytm Dijkstry



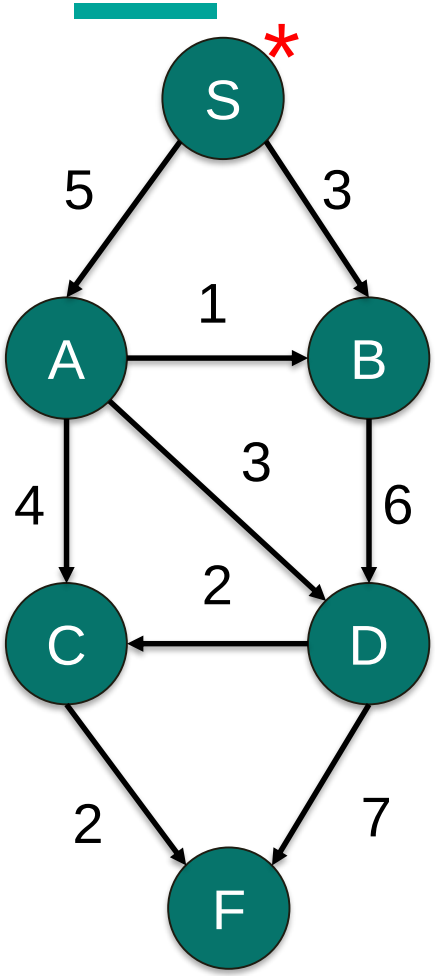
$$Odwiedzone = []$$
$$Nieodwiedzone = [S, A, B, C, D, F]$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	∞	
B	∞	
C	∞	
D	∞	
F	∞	

W każdym następnym kroku:

- Wybierz nieodwiedzony wierzchołek o najkrótszej ścieżce od v_o.

Algorytm Dijkstry



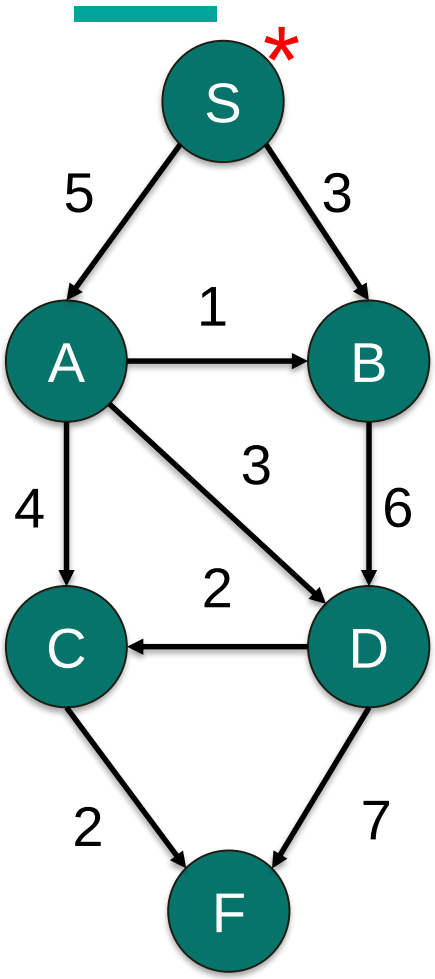
$$Odwiedzone = [S]$$
$$Nieodwiedzone = [A, B, C, D, F]$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	∞	
B	∞	
C	∞	
D	∞	
F	∞	

W każdym następnym kroku:

- Wybierz nieodwiedzony wierzchołek o najkrótszej ścieżce od v_o.
- Oznacz go jako odwiedzony.

Algorytm Dijkstry



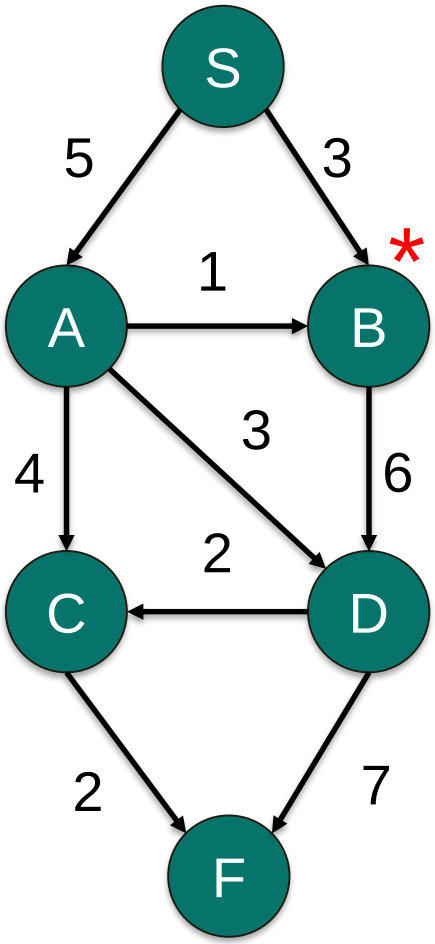
$$Odwiedzone = [S]$$
$$Nieodwiedzone = [A, B, C, D, F]$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	~~∞~~ 5	S
B	~~∞~~ 3	S
C	∞	
D	∞	
F	∞	

W każdym następnym kroku:

- Wybierz nieodwiedzony wierzchołek o najkrótszej ścieżce od v_o.
- Oznacz go jako odwiedzony.
- Wyznacz najkrótszą ścieżkę do sąsiadujących z nim wierzchołków.

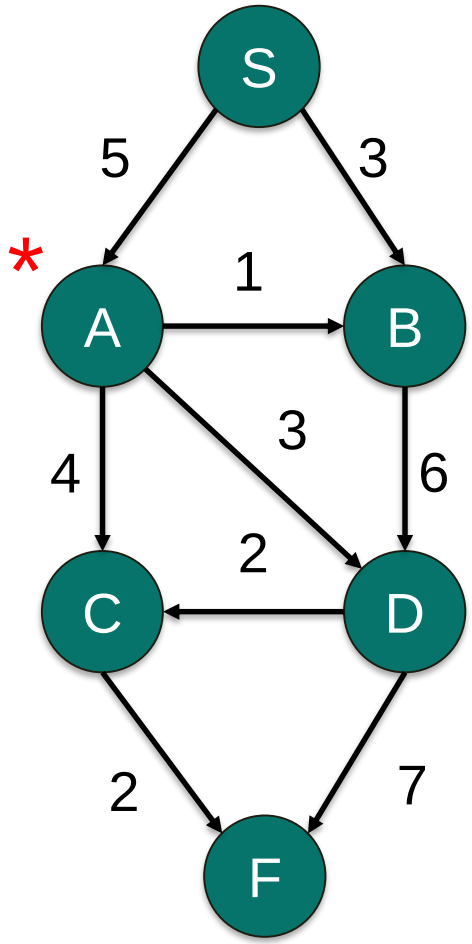
Algorytm Dijkstry



$$Odwiedzone = [S, B]$$
$$Nieodwiedzone = [A, C, D, F]$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	5	S
B	3	S
C	∞	
D	~~∞~~ 9	B
F	∞	

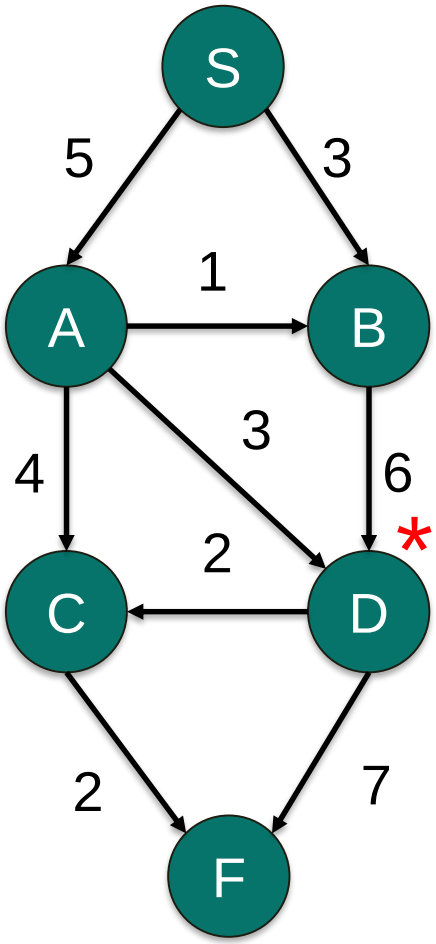
Algorytm Dijkstry



$$Odwiedzone = [S, B, A]$$
$$Nieodwiedzone = [C, D, F]$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	5	S
B	3	S
C	~~∞~~ 9	A
D	~~9~~ 8	~~B~~ A
F	∞	

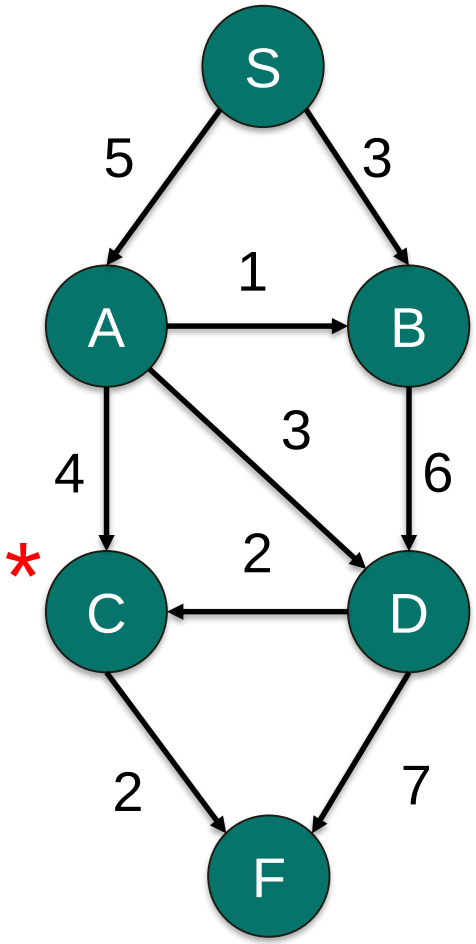
Algorytm Dijkstry



$$Odwiedzone = [S, B, A, D]$$
$$Nieodwiedzone = [C, F]$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	5	S
B	3	S
C	9	A
D	8	A
F	~~∞~~ 15	D

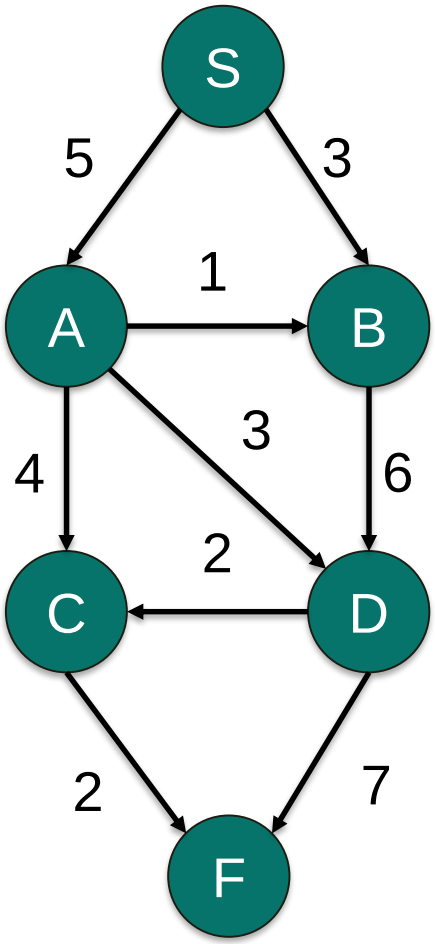
Algorytm Dijkstry



$$Odwiedzone = [S, B, A, D, C]$$
$$Nieodwiedzone = [F]$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	5	S
B	3	S
C	9	A
D	8	A
F	~~15~~ 11	~~D~~ C

Algorytm Dijkstry

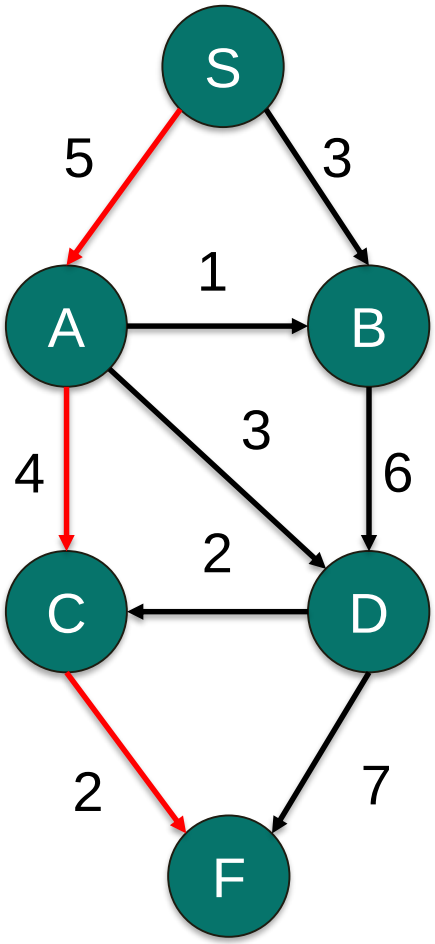


$$Odwiedzone = [S, B, A, D, C, F]$$
$$Nieodwiedzone = []$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	5	S
B	3	S
C	9	A
D	8	A
F	11	C

W ostatnim kroku należy po prostu wyekstrahować odpowiednią ścieżkę na podstawie zebranych informacji.

Algorytm Dijkstry



$$Odwiedzone = [S, B, A, D, C, F]$$
$$Nieodwiedzone = []$$

Wierzchołek	Najkrótsza ścieżka z **S**	Poprzedni wierzchołek
S	0	-
A	5	S
B	3	S
C	9	A
D	8	A
F	11	C

W ostatnim kroku należy po prostu wyekstrahować odpowiednią ścieżkę na podstawie zebranych informacji.

Widzimy, że jest to ciąg wierzchołków $[S, A, C, F]$

o długości ścieżki równej 11.

Algorytm Dijkstry

- Złożoność obliczeniowa algorytmu Dijkstry wynosi $\mathrm{O}(|E| + |V| \log |V|)$.
- Jest on jednym z najefektywniejszych algorytmów rozwiązywania problemów najkrótszej ścieżki.
- Jego użycie nie jest jednak możliwe gdy wagi grafu są ujemne.

https://developers.google.com/maps/documentation/directions/start

Google Directions API

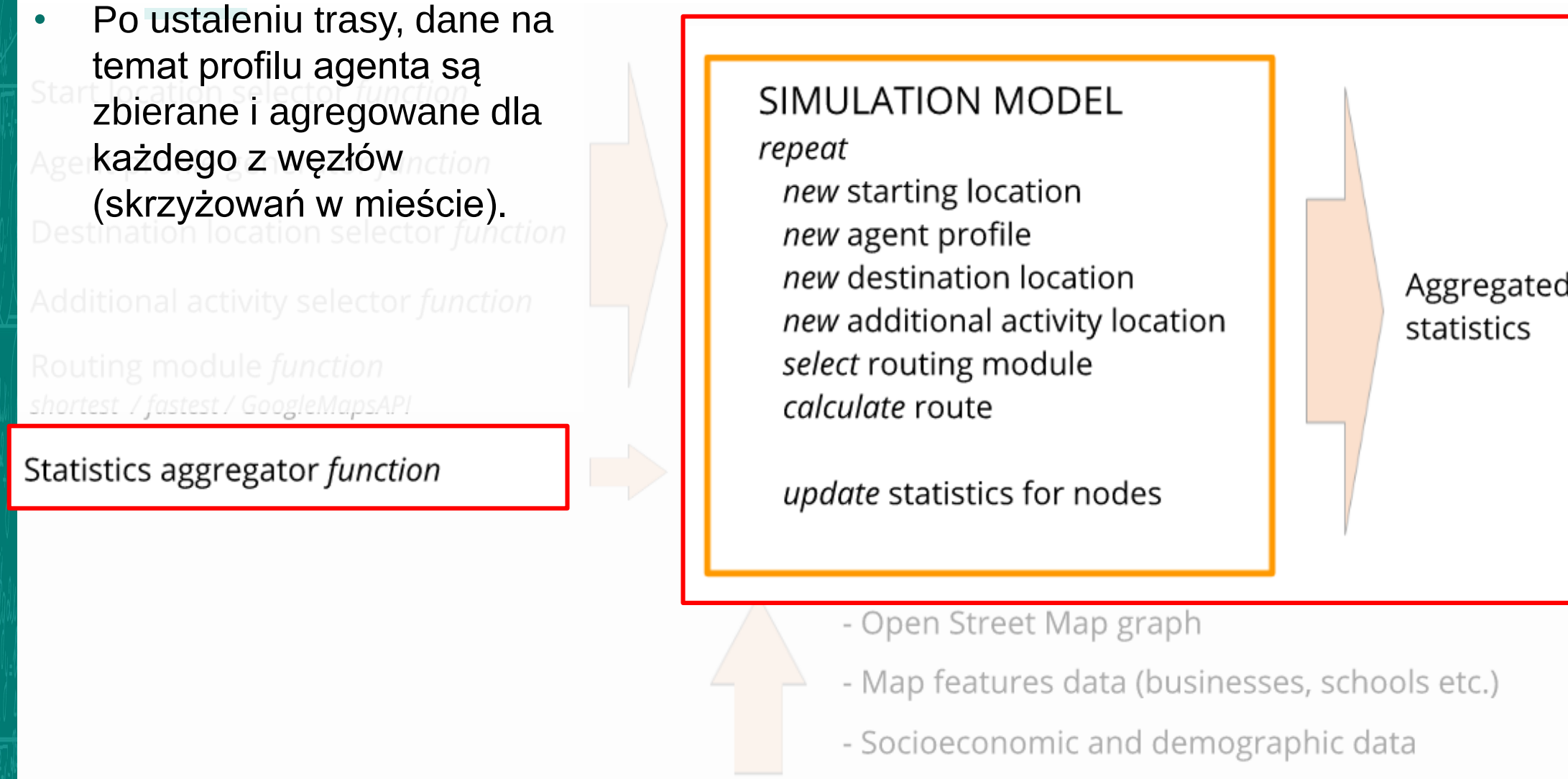
- Platforma Google Maps udostępnia komercyjne narzędzie pozwalające na wyszukanie najszybszej trasy pomiędzy punktami A i B.
- Jego kluczowymi zaletami są:
  - Możliwość szczegółowego dopasowania zapytania do swoich potrzeb.
  - Dynamiczne dostosowywanie czasów przejazdu do zmieniających się warunków na drodze (remontów, korków, etc.).
- Otrzymane wyniki są zwracane w formacie XML lub JSON.

Google Directions API

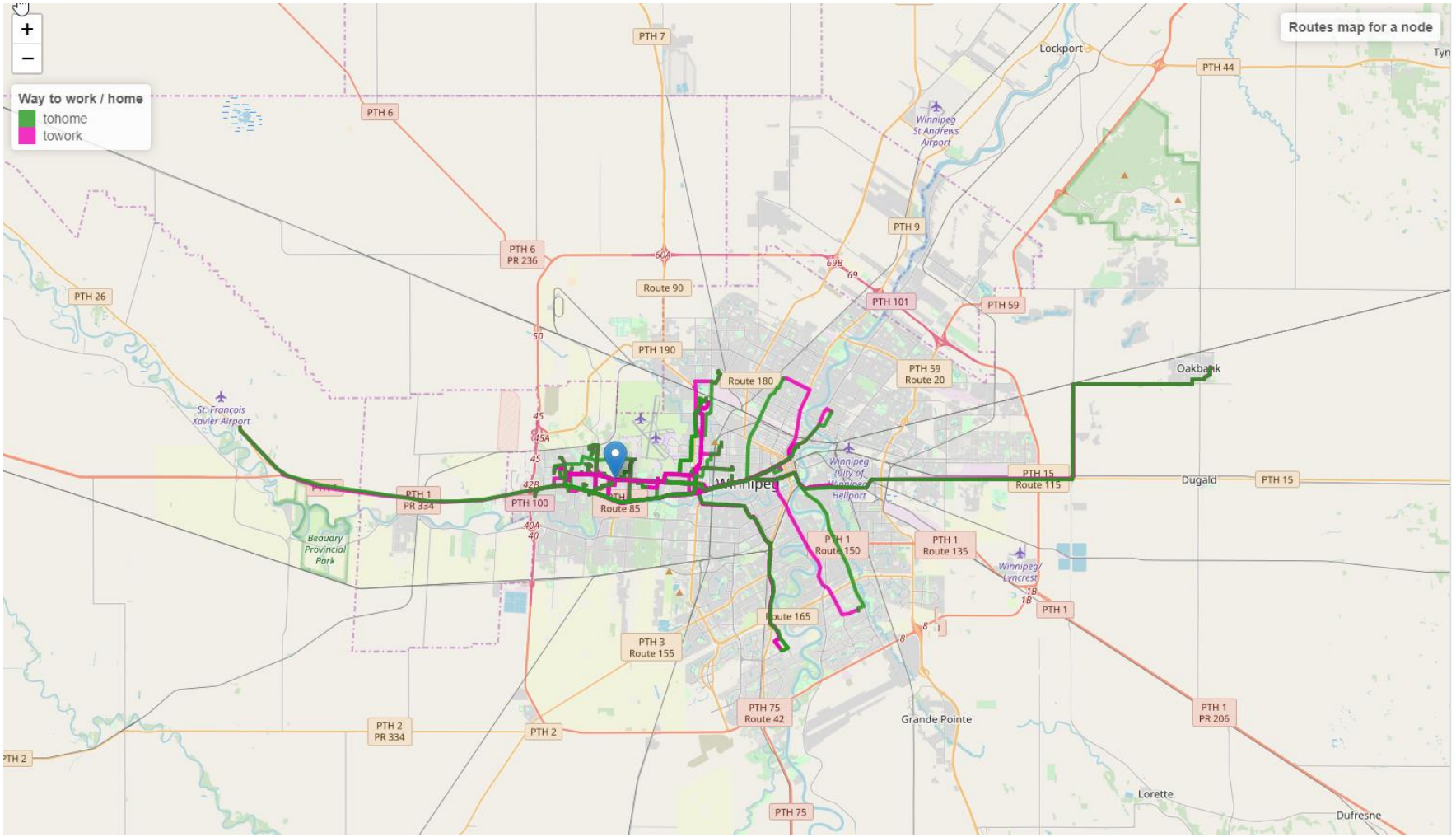
- Mapy Google'a są szczegółowe i przechowują relatywnie dużo informacji na temat otoczenia.
- Aby możliwa była efektywna praca z ich wykorzystaniem konieczne jest odpowiednie zakodowanie tych danych.
- Google wykorzystuje do tego format Encoded Polyline Algorithm.
- Koduje on wszystkie współrzędne trasy do postaci ciągu znaków ASCII, np.: _p~iF~ps|U_ulLnnqC_mqNvxq`@

Opis modelu

- Po ustaleniu trasy, dane na temat profilu agenta są zbierane i agregowane dla każdego z węzłów (skrzyżowań w mieście).



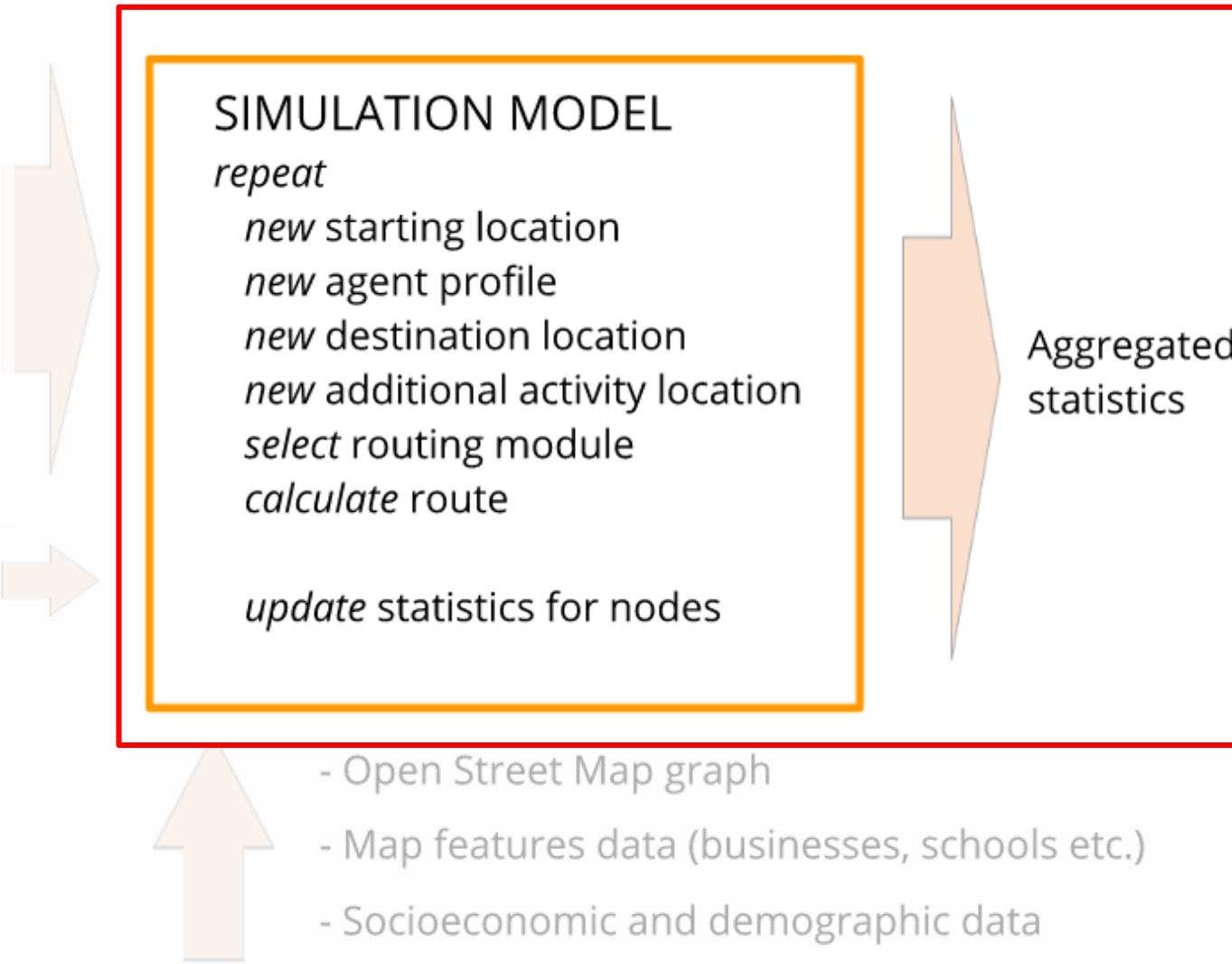
Opis modelu


Routes map for a node
Way to work / home
tohome
towork
PTH 7
PTH 6
PTH 6
PR 236
PTH 26
Route 90
PTH 190
Route 180
PTH 9
PTH 101
PTH 59
PTH 59
Route 20
Lockport
PTH 44
Winnipeg
St Andrews
Airport
Oakbank
St. François
Xavier Airport
PTH 1
PR 334
PTH 100
Route 85
Winnipeg
(City of
Winnipeg)
Heliport
PTH 15
Route 115
Dugald
PTH 15
Beaudry
Provincial
Park
PTH 1
Route 150
PTH 1
Route 135
Winnipeg/
Lyncrest
PTH 1
Route 165
PTH 3
Route 155
PTH 75
Route 42
Grande Pointe
PTH 1
PR 206
PTH 2
PR 334
PTH 2
PTH 2
Lorette
PTH 75
Dufresne

Opis modelu

- Po ustaleniu trasy, dane na temat profilu agenta są zbierane i agregowane dla każdego z węzłów (skrzyżowań w mieście).
- Cała operacja jest powtarzana *N* razy dla kolejnych agentów reprezentujących mieszkańców miasta.

Statistics aggregator *function*

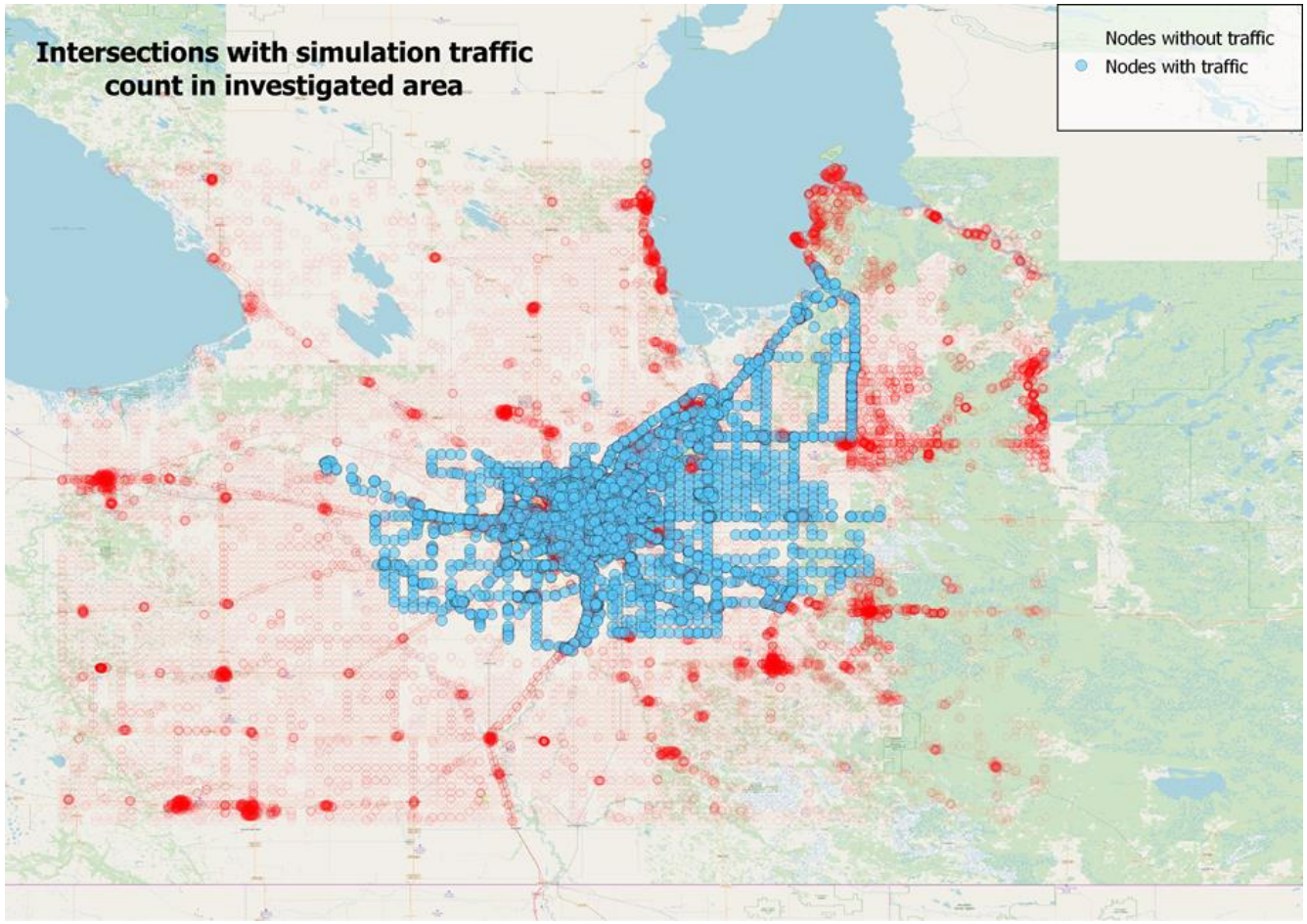


Wyniki

- Symulacja została uruchomiona dla populacji ok. 1 000 000 agentów, co w przybliżeniu odpowiada dziennemu ruchowi drogowemu w Winnipeg.

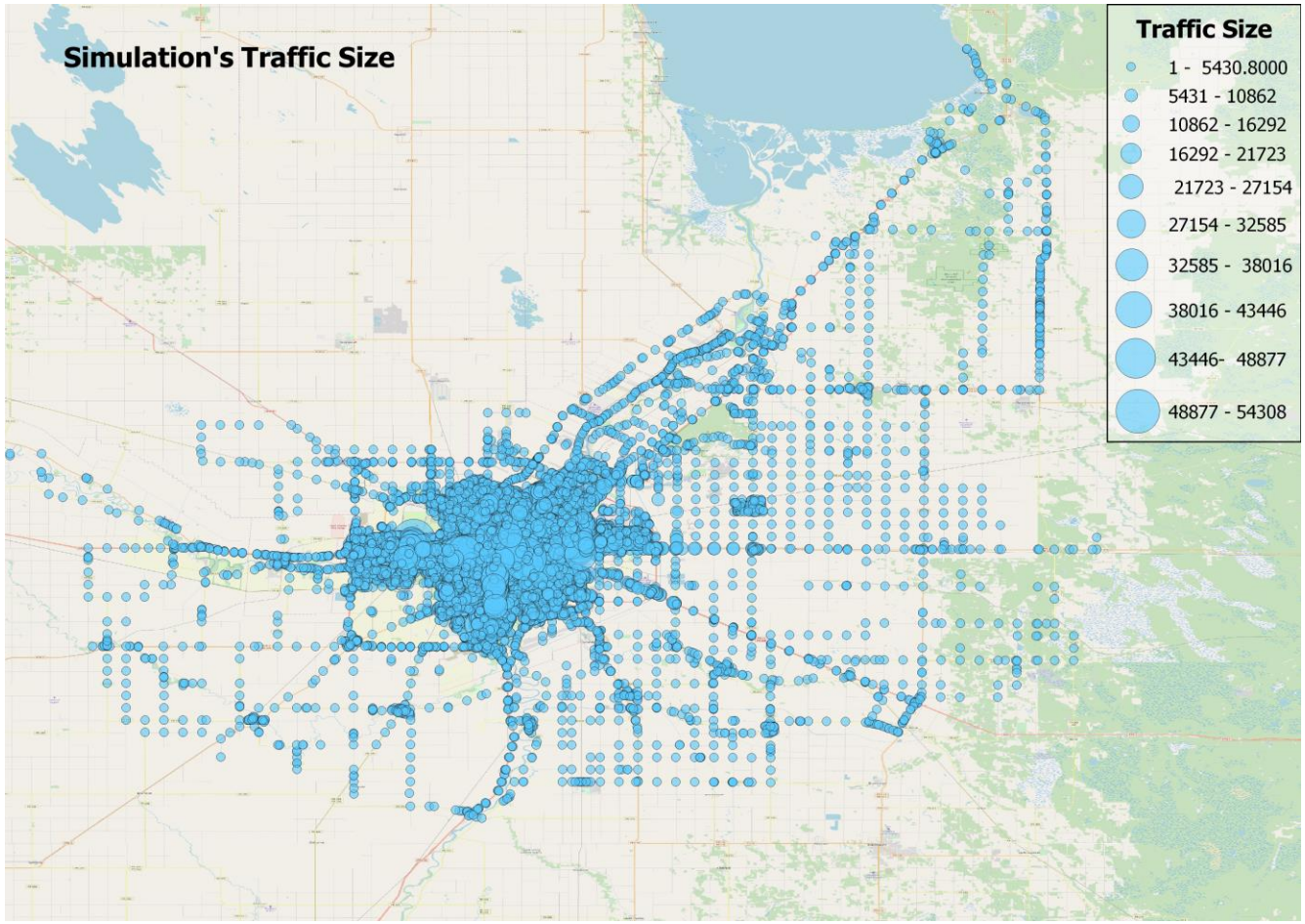
Porównanie skrzyżowań ze względu na ruch drogowy.

Wyniki



Wielkość ruchu drogowego na skrzyżowaniach w przykładowym uruchomieniu symulacji.

Wyniki

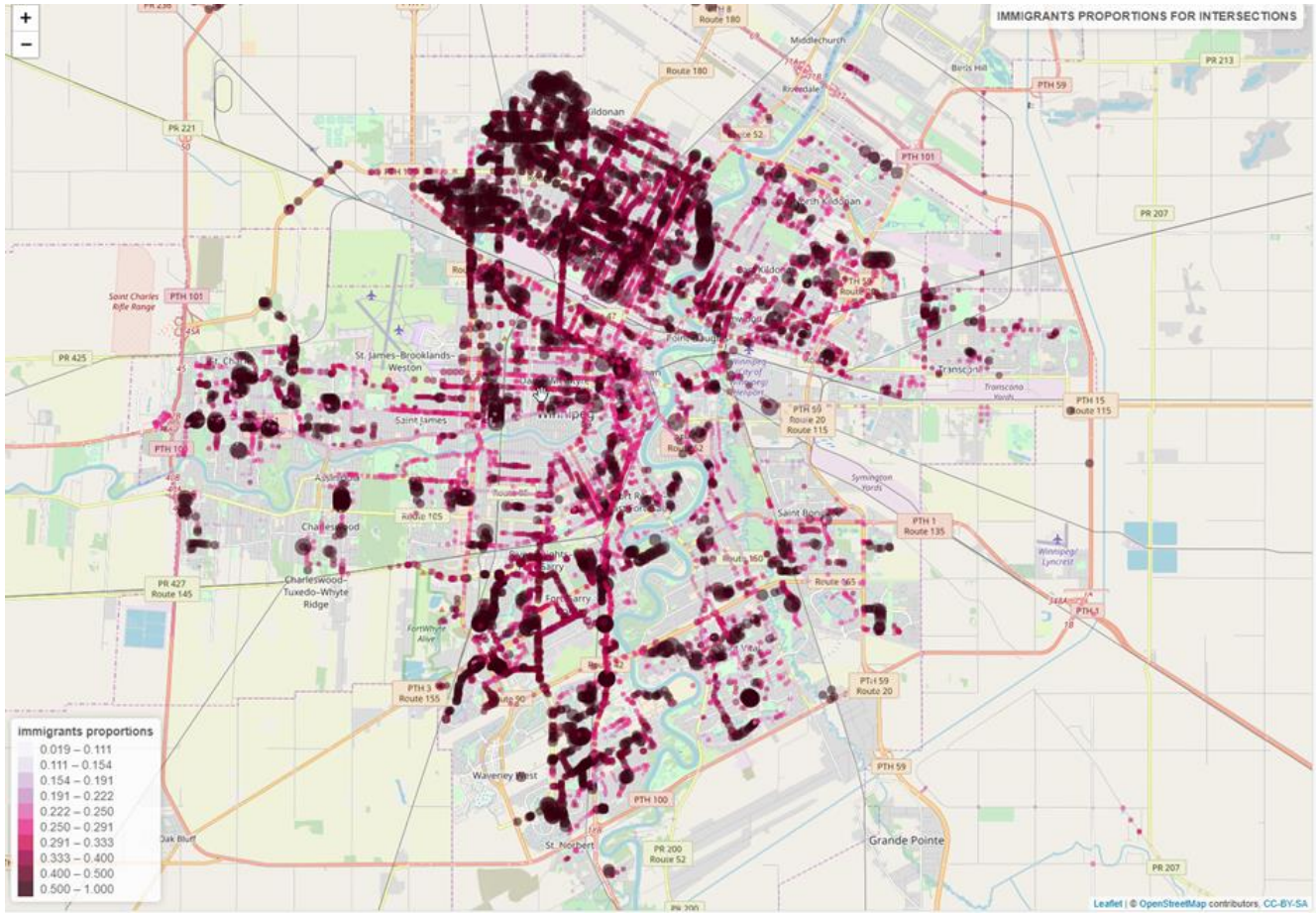


Wielkość ruchu drogowego na skrzyżowaniach w przykładowym uruchomieniu symulacji – tylko centrum miasta.

Wyniki

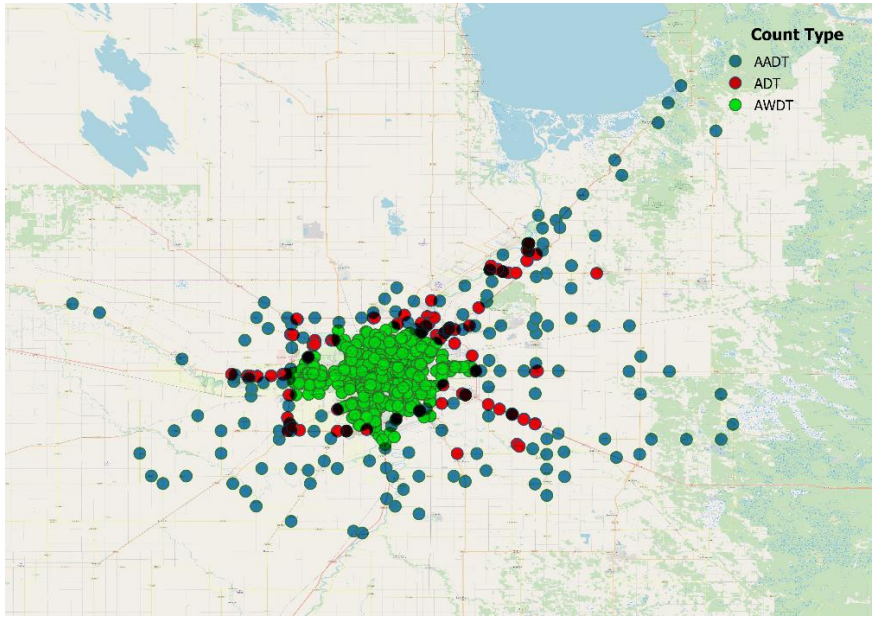
Udział imigrantów wśród osób przejeżdżających przez dane skrzyżowanie.

Wyniki



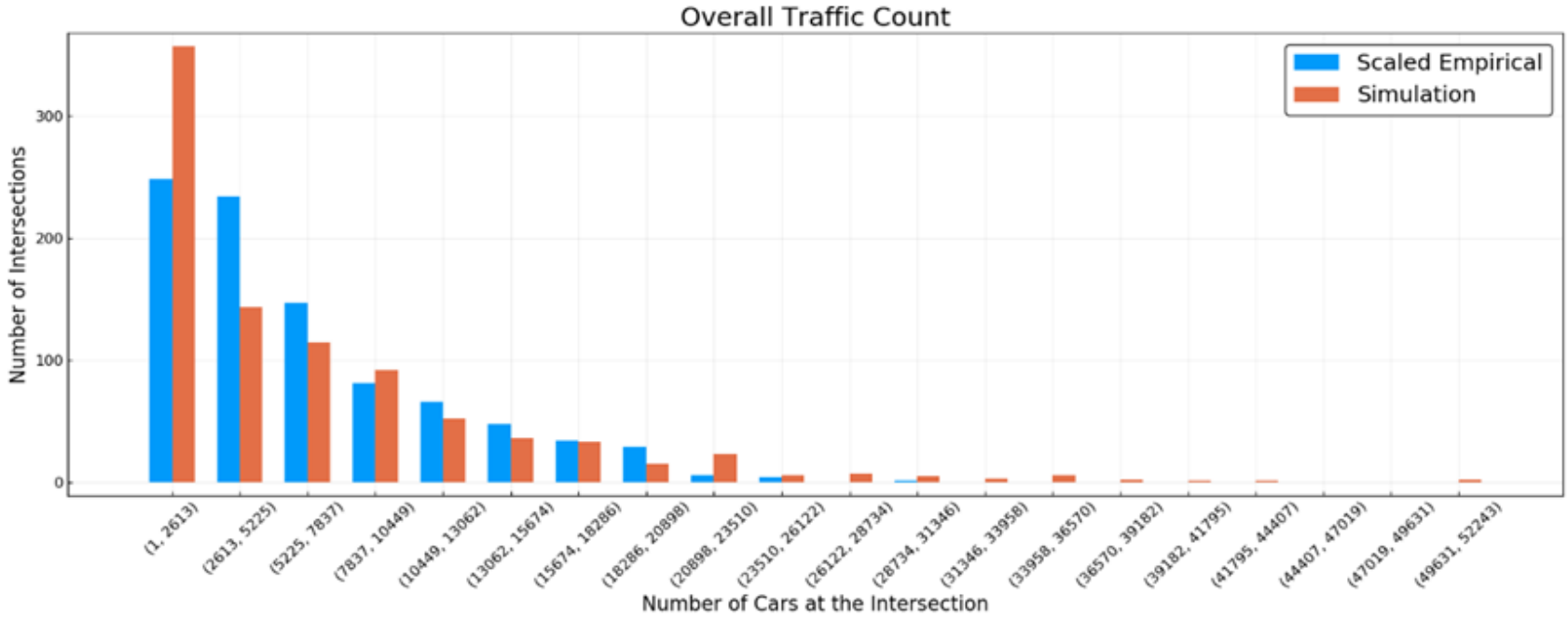
Wyniki

- W celu sprawdzenia poprawności działania symulacji wyniki zostały porównane z rzeczywistymi danymi dotyczącymi natężenia ruchu w Winnipeg.
- Dane dotyczyły ok. 900 punktów pomiarowych (niekoniecznie skrzyżowań).
- Wykres przedstawia ich rozmieszczenie na mapie:



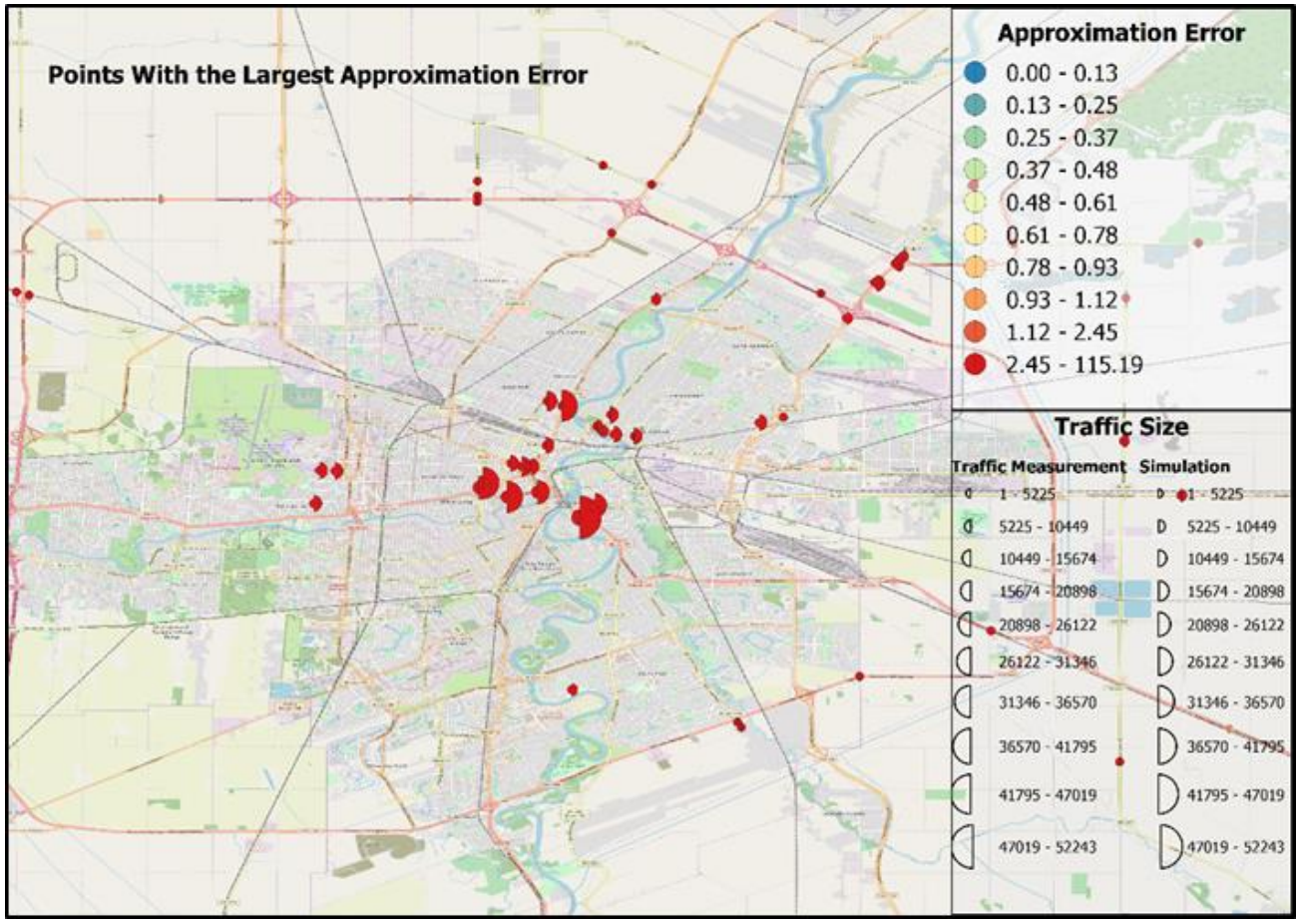
Wyniki

- Rozkłady natężenia ruchu dla danych rzeczywistych pokazane są na poniższym wykresie:



Rozmieszczenie punktów z największym błędem aproksymacji.

Wyniki



Co dalej?

- Zaproponowane narzędzie pozwala na efektywną symulację ruchu drogowego w dużej skali.
- Otrzymane wyniki w znacznym stopniu pokrywają się z rzeczywistymi danymi.
- Kod użyty do napisania symulacji jest wystarczająco elastyczny aby możliwe było dalsze rozwijanie modelu (na przykład poprzez dodanie pamięci i uczenia się do zachowania agentów).
- Ponadto możliwa jest jego modyfikacja w celu rozwiązywania innych problemów związanych z ruchem drogowym w mieście (planowanie remontów, tras komunikacji miejskiej, etc.).

Zadanie dodatkowe

- Punktowane jest każde wykrycie i usunięcie błędu lub modyfikacja poprawiająca działanie prezentowanych bibliotek.
- Propozycję należy wysłać jako issue na repozytorium odpowiedniej biblioteki na GitHubie.