KATEDRA INFORMATYKI

Wydział Informatyki, Elektorniki i Telekomunikacji AGH

AGH

Badania Operacyjne

Optymalizacja trasy w trybie online

*M*ichał Bigaj
*M*iłosz Łakomy
*S*ebastian Łakomy
*P*aweł Maślak
*J*akub Nowosiński
*K*rzysztof Trzepla
*J*an Wójcik

23 stycznia 2014

Spis treści

1 Wstęp

Niniejszy projekt ma na celu zbadanie wpływu dynamicznych zmian trasy pojazdu, w zależności od panujących warunków, na sumaryczny czas przejazdu. W pierwszej części opracowania przedstawiony zostanie sposób modelowania ruchu drogowego z wykorzystaniem symulatora *SUMO* oraz wykorzystane w modelowaniu narzędzia, m.in. biblioteka *TraCI*. W drugiej części opracowania opisana zostanie utworzona aplikacja oraz proces symulacji. W trzeciej części znajdzie się opis przeprowadzonych testów wraz z danymi w formie wykresów oraz tabel. W czwartej, ostatniej części opracowania dokonana zostanie analiza otrzymanych wyników wraz z podsumowaniem oraz wnioskami.

Kod źródłowy projektu znajduje się na płycie CD załączonej do dokumentacji. Dodatkowo znajduje się w repozytorium pod adresem: `https://github.com/OR2013/OnlinePathOptimalization`.

2 Modelowanie ruchu drogowego

Istnieje wiele narzędzi umożliwiających symulację ruchu drogowego. W niniejszym projekcie wykorzystano symulator *SUMO* utworzony przez pracowników *Instytutu Transportu Niemieckiej Agencji Kosmicznej* i udostępniony na licencji *GPL*.

2.1 Symulator SUMO

***S**imulation of **U**rban **MO**bility*, w skrócie *SUMO*, to wysoce wyspecjalizowany symulator umożliwiający wytworzenie sieci połączeń oraz badanie zachowania pojazdów w ruchu drogowym.

Cechy symulatora:

- możliwość wygenerowania sieci połączeń oraz ruchu drogowego bez konieczności korzystania z innych aplikacji
- ciągły w czasie i przestrzeni ruch pojazdów
- ruch każdego pojazdu jest modelowany oddzielnie
- duży wybór dostępnych pojazdów (samochody osobowe, samochody ciężarowe, autobusy, itp.)
- duży wybór dostępnych dróg (jednopasmowe, wielopasmowe)
- możliwość wprowadzenia oraz sterowania sygnalizacją świetlną

- możliwość komunikacji z zewnętrznymi aplikacjami, m.in. biblioteka *TraCi*
- konfiguracja symulatora, opis sieci oraz ruchu drogowego z wykorzystaniem plików XML

2.2 Biblioteka TraCI

***Tra**ffic **C**ontrol **I**nterface*, w skrócie *TraCI*, to biblioteka pozwalająca na dynamiczne pobieranie danych na temat obiektów będących częścią symulacji oraz modyfikowanie ich zachowania w danym kroku symulacji. Komunikacja z procesem symulującym ruch drogowy odbywa się z wykorzystaniem socketu TCP. Domyślny numer portu na którym nasłuchuje proces symulujący wynosi 8813. Pełna dokumentacja biblioteki znajduje się pod adresem `http://sumo-sim.org/pydoc/traci.html`.

2.3 Aplikacja Netgenerate

Netgenerate to aplikacja, będąca częścią projektu *SUMO*, umożliwiająca wygenerowanie sieci drogowej, która następnie jest wykorzystywana przez symulator. Szeroka gama parametrów obsługiwanych przez aplikację pozwala na generację różnorodnych sieci drogowych.

W projekcie wkorzystano następujące opcje aplikacji *Netgenerate*:

–**random** - tworzenie sieci drogowej o losowej strukturze

–**rand.iterations** - ilość krawędzi (ulic) w sieci drogowej

–**rand.max-distance** - maksymalna długość krawędzi (ulicy) w sieci drogowej

–**no-turnarounds** - zakaz zawracania na skrzyżowaniach

–**output** - plik w formacie XML stanowiący opis sieci drogowej

2.4 Skrypt randomTrips

Skrypt *randomTrips* umożliwia wytworzenie punktów startowych i docelowych dla poszczególnych pojazdów, a następnie znalezienie trasy łączącej zaproponowane punkty. Powstały ruch drogowy jest zapisywany w formacie XML odpowiednim dla symulatora *SUMO*.

W projekcie wykorzystano następujące opcje skryptu *randomTrips*:

-e - okres trwania ruchu drogowego

-p - częstotliwość określająca pojawianie się nowych samochodów w sieci drogowej

–**fringe-factor** - prawdopodobieństwo, że trasa poszczególnych pojazdów będzie się rozpoczynać i kończyć na obrzeżach sieci drogowej

-n - plik zawierający opis sieci drogowej w formacie XML (wygenerowany przez aplikację *Netgenerate*)

-r - plik w formacie XML stanowiący opis ruchu drogowego

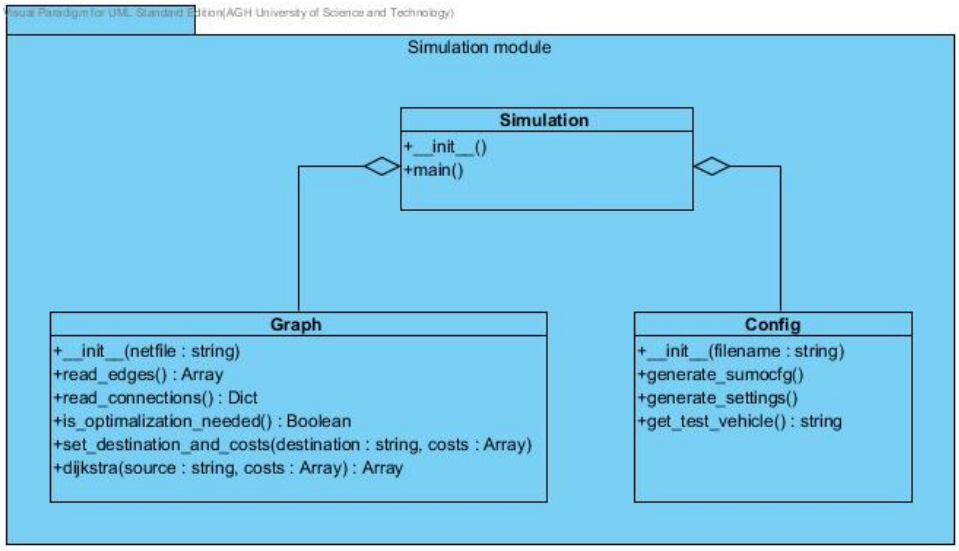
3 Aplikacja

3.1 Opis

W ramach projektu utworzona została aplikacja umożliwiająca badanie wpływu dynamicznych zmian trasy wybranego pojazdu na sumaryczny czas podróży.

3.1.1 Struktura aplikacji

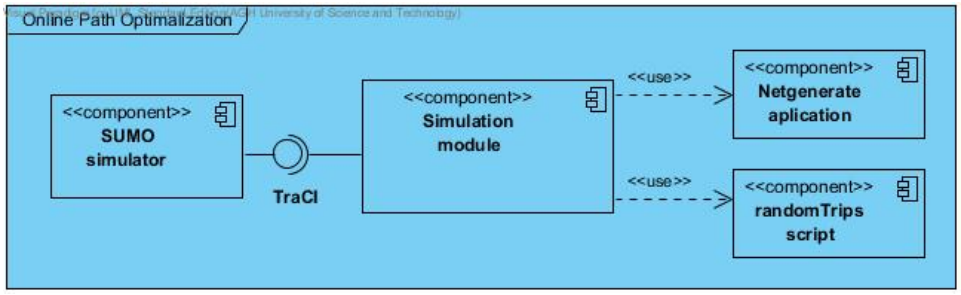
Centralny punkt aplikacji stanowi moduł symulacyjny. Jest on odpowiedzialny za wygenerowanie sieci i ruchu drogowego oraz za przeprowadzenie symulacji (zainicjowanie symulatora, komunikację z symulatorem). Diagram klas modułu symulacyjnego zamieszczony jest poniżej:



Rysunek 1: Diagram klas.

W projekcie wyróżniamy następujące komponenty:

- moduł symulacyjny
- symulator *SUMO*
- biblioteka *TraCI*
- aplikacja *Netgenerate*
- skrypt *randomTrips*



Rysunek 2: Diagram komponentów.

3.1.2 Proces symulacji

Symulacja przebiega w następujących krokach:

1. Wygenerowanie sieci drogowej przy pomocy aplikacji *Netgenerate*.
2. Wygenerowanie ruchu drogowego przy pomocy skryptu *randomTrips*.
3. Uruchomienie symulacja w której podejmuje się próbę optymalizacji trasy wybranego pojazdu (pojawiającego się mniej więcej w połowie czasu trwania symulacji, tak aby w sieci drogowej istniał ruch) na podstawie aktualnego stanu dróg (oczekiwanego czasu przejazdu).
4. Poszukiwanie szybszej trasy dojazdu jest inicjowane gdy spełnione są następujące warunki:
   - pojazd, którego trasa jest optymalizowana, pokonał co najmniej 0.9 odcinka obecnej drogi
   - nastąpiło wydłużenie czasu przejazdu przez któryś z odcinków należących do aktualnej trasy samochodu lub nastąpiło skrócenie czasu przejazdu przez któryś z odcinków nie należących do aktualnej trasy samochodu. W obu przypadkach rozpatrywane są tylko zmiany, których różnica względem początkowego stanu dróg jest większa od parametru *alpha*.

5. Ponowienie symulacji przejazdu wybranego pojazdu dla takiego samego stanu dróg oraz ruchu drogowego, z tą różnicą że tym razem trasa nie jest optymalizowana online, a pojazd porusza się z góry ustaloną (najlepszą w chwili początkowej) trasą.

3.1.3 Parametry aplikacji

Aplikacja udostępnia następujące parametry:

–**nogui** - symulacja bez wizualizacji

–**nogen** - symulacja bez generowania sieci oraz ruchu drogowego

–**prefix** - prefiks nazw tworzonych plików XML

–**size** - rozmiar sieci drogowej

–**alpha** - parametr określający jak duże zmiany czasu przejazdu są uwzględniane

–**rate** - częstotliwość z jaką pojawiają się pojazdy w sieci

–**verbose** - szczegółowe informacje na temat symulacji

–**max-distance** - maksymalna długość krawędzi (ulicy) w generowanej sieci drogowej

3.1.4 Makefile

Do aplikacji dołączony jest plik *Makefile* automatyzujący procesy testowania aplikacji, przeprowadzania eksperymentów oraz generowania dokumentacji.

Makefile udostępnia następujące parametry:

make test - uruchomienie załączonych testów aplikacji

make simulations - przeprowadzenie wszystkich zdefiniowanych eksperymentów

make simulation(nr) - przeprowadzenie eksperymentu numer *nr*, gdzie $1 <= nr <= 7$

make graphs - wygenerowanie wykresów

make docs - wygenerowanie dokumentacji projektu

3.1.5 Testy

W celu sprawdzenia poprawności funkcji udostępnianych przez aplikacje do projektu załączono testy jednostkowe. Można je uruchomić wydając w katalogu głównym projektu polecenie *make test*.

Testy jednostkowe:

test read edges - test sprawdzający poprawność wczytania struktury sieci drogowej

test read connections - test sprawdzający poprawność wczytania połączeń w sieci drogowej

test is optimalization needed - test sprawdzający poprawność warunków w których następuje optymalizacja trasy

test dijkstra simple - test sprawdzający poprawność implementacji algorytmu Dijkstry dla grafu o dziesięciu wierzchołkach

test dijkstra complex - test sprawdzający poprawność implementacji algorytmu Dijkstry dla grafu o tysiącu wierzchołków

3.2 Algorytm Dijkstry

W projekcie poszukiwanie szybszej trasy dojazdu odbywa się z wykorzystaniem algorytmu Dijkstry.

3.2.1 Opis

Przez s oznaczamy wierzchołek źródłowy, $w(i, j)$ to waga krawędzi (i, j) w grafie.

- Stwórz tablicę d odległości od źródła dla wszystkich wierzchołków grafu. Na początku $d[s] = 0$, zaś $d[v] = \infty$ dla wszystkich pozostałych wierzchołków.
- Utwórz kolejkę priorytetową Q wszystkich wierzchołków grafu. Priorytetem kolejki jest aktualnie wyliczona odległość od wierzchołka źródłowego s.
- Dopóki kolejka nie jest pusta:
  - Usuń z kolejki wierzchołek u o najniższym priorytecie (wierzchołek najbliższy źródła, który nie został jeszcze rozważony)
  - Dla każdego sąsiada v wierzchołka u dokonaj *relaksacji* poprzez u: jeśli $d[u] + w(u, v) < d[v]$ (poprzez u da się dojść do v szybciej niż dotychczasową ścieżką), to $d[v] := d[u] + w(u, v)$.

- Na końcu tablica d zawiera najkrótsze odległości do wszystkich wierzchołków.

Dodatkowo możemy w tablicy *poprzednik* przechowywać dla każdego wierzchołka numer jego bezpośredniego poprzednika na najkrótszej ścieżce, co pozwoli na odtworzenie pełnej ścieżki od źródła do każdego wierzchołka - przy każdej relaksacji w ostatnim punkcie u staje się poprzednikiem v.

4 Symulacje

W celu zbadanie wpływu dynamicznych zmian trasy wybranego pojazdu na sumaryczny czas podróży przeprowadzono szereg symulacji, których specyfikacja znajduje się poniżej. Zamieszczone wyniki są wartościami średnimi pochodzącymi z dziesięciu prób.

4.1 Symulacja 1

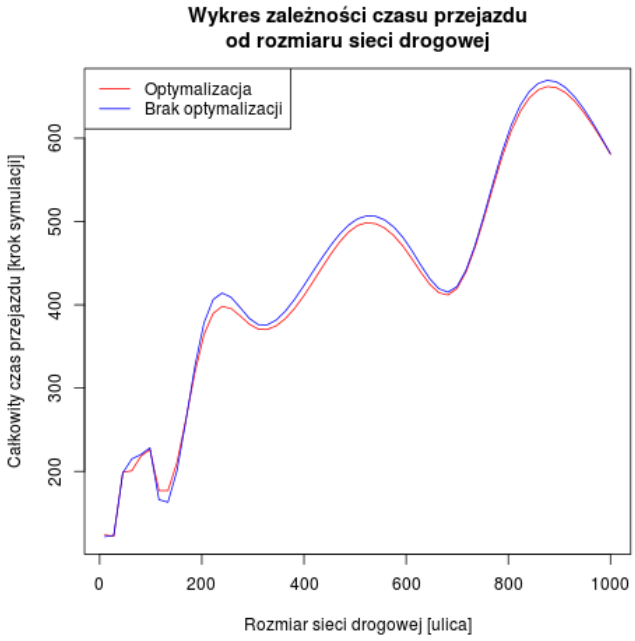
4.1.1 Opis

Pierwsza symulacja miała na celu sprawdzenie jak optymalizacja trasy w trybie online oddziałuje na czas podróży dla różnych wielkości sieci drogowych. Parametr *alpha* wynosił *0.05*, częstotliwość pojawiania się samochodów w sieci drogowej wynosiła *2 kroki symulacji/pojazd* oraz maksymalna długość ulicy wynosiła *500 metrów*.

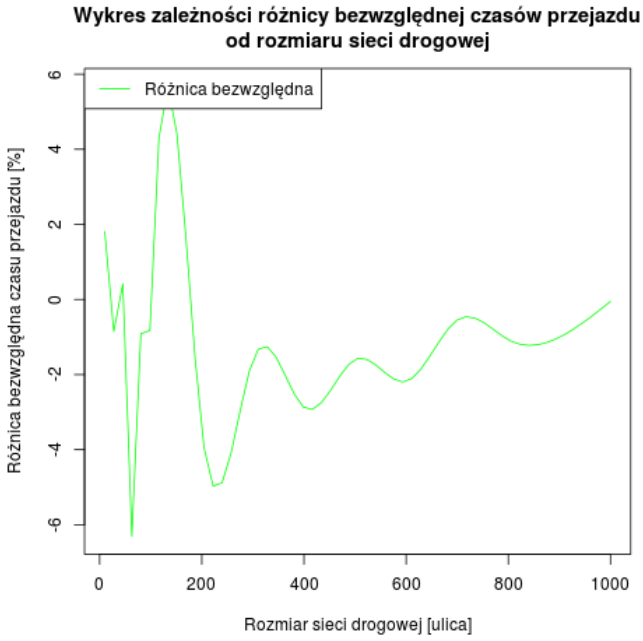
4.1.2 Wyniki

Rozmiar sieci drogowej [ulica]	Optymalizacja [krok symulacji]	Brak optymalizacji [krok symulacji]
10	123.8	121.6
20	137.8	143.7
30	122.5	122.1
40	173.5	171.4
50	213.2	216.4
60	213.9	230.2
70	174.8	178.2
80	214.1	215.7
90	250.4	258.3
100	219.4	219.8
200	355.2	368
300	373.7	379.8
400	411.1	423.3
500	493.6	501.6
600	465.3	475.7
700	420	422.3
800	599.7	606.2
900	659.3	666
1000	581	581.3

Tabela 1: Zależność czasu przejazdu od rozmiaru sieci drogowej.



Rysunek 3: Zależność czasu przejazdu od rozmiaru sieci drogowej.



Rysunek 4: Zależność bezwzględnej różnicy czasów przejazdu od rozmiaru sieci drogowej.

4.1.3 Analiza wyników

Analiza otrzymanych wyników pozwala stwierdzić korzystny wpływ zastosowanej metody optymalizacyjnej na sumaryczny czas przejazdu. W znakomitej większości przypadków czas przejazdu pojazdu, którego trasa była zmienia dynamicznie w zależności od sytuacji panującej w sieci drogowej, okazał się być krótszy od czasu przejazdu dla pojazdu poruszającego się niezmienną trasą. Przewaga wspomnianej metody jest szczególnie widoczna dla średnich grafów o rozmiarze wahającym się od dwustu do trzystu krawędzi i wynosi około 5 procent. Jest to związane z ilością możliwych ścieżek od punktu początkowego do końcowego. Dla małych grafów ilość możliwych ścieżek łączących dwa zadane wierzchołki jest niewielka, a co za tym idzie możliwości wybrania lepszej trasy dojazdu są ograniczone. Z kolei dla dużych grafów liczba ta jest dużo większa, stąd ryzyko podejmowania decyzji nieoptymalnych w skali globalnej jest bardziej prawdopodobne.

4.2 Symulacja 2

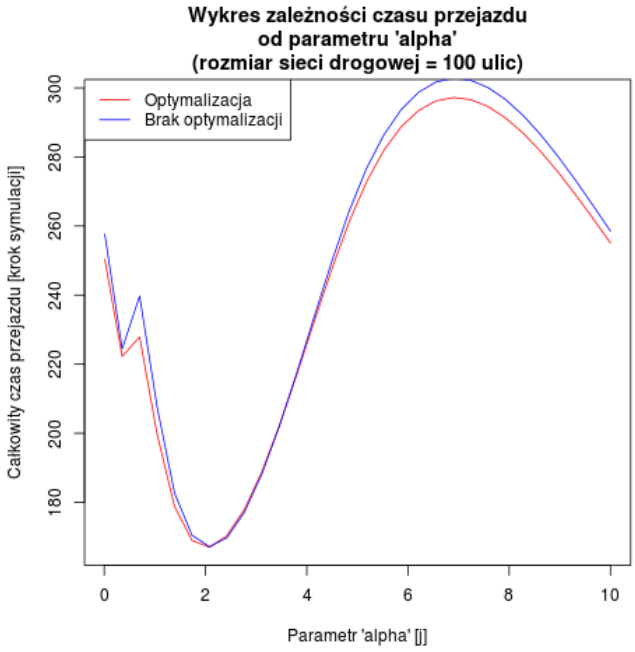
4.2.1 Opis

Druga symulacja polegała na zbadaniu wpływu parametru *alpha* na sumaryczny czas podróży. Rozmiar sieci drogowej wynosił *100 ulic*, częstotliwość pojawiania się samochodów w sieci drogowej wynosiła *2 kroki symulacji/pojazd* oraz maksymalna długość ulicy wynosiła *500 metrów*.

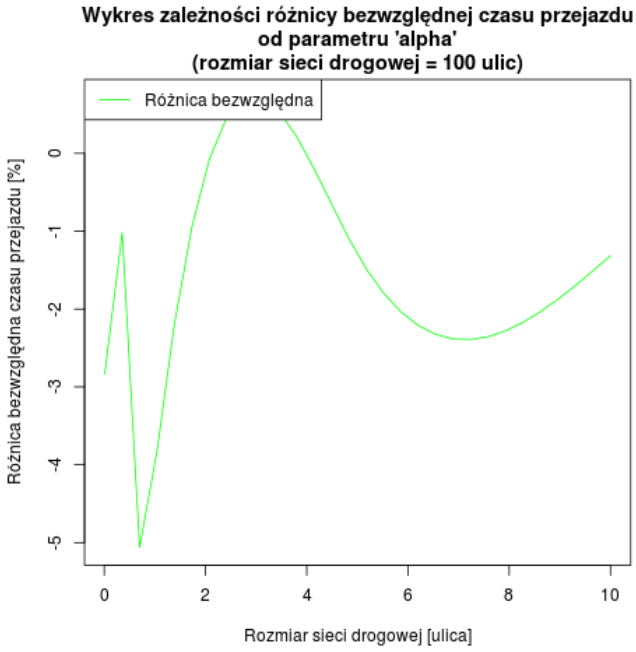
4.2.2 Wyniki

Parametr alpha [j]	Optymalizacja [krok symulacji]	Brak optymalizacji [krok symulacji]
0.01	250.4	257.7
0.02	247.3	256.2
0.05	218.3	222.3
0.1	231.7	234.8
0.2	232.3	233.3
0.5	230.1	238.5
1	203.4	212
2	166.9	167.3
5	267	270.5
10	255.1	258.5

Tabela 2: Zależność czasu przejazdu od parametru *alpha*'.



Rysunek 5: Zależność czasu przejazdu od parametru *alpha*.



Rysunek 6: Zależność bezwzględnej różnicy czasów przejazdu od parametru *alpha*.

4.2.3 Analiza wyników

Parametr *alpha* określał czy zmiana czasu przejazdu przez daną krawędź jest istotna z punktu widzenia metody optymalizacyjnej. Uzyskane wyniki pozwalają stwierdzić, że dla grafów rozmiaru około stu krawędzi średnie czasy przejazdu są najlepsze dla wartości parametru *alpha* wynoszącej około 3 do 4. Oznacza to, że najkorzystniejsze okazało się uwzględnianie zmian, które trzy do czterokrotnie wydłużały czas przejazdu przez daną krawędź. Wynika to z faktu, że analizowanie niewielkich zmian czasu przejazdu może prowadzić do podejmowania częstych i prawdopodobnie nieoptymalnych decyzji zmian trasy. Natomiast dla dużych wartości parametru *aplha* znakomita większość zmian jest odrzucana, a co za tym idzie nie jest dokonywana optymalizacja trasy przejazdu.

4.3 Symulacja 3

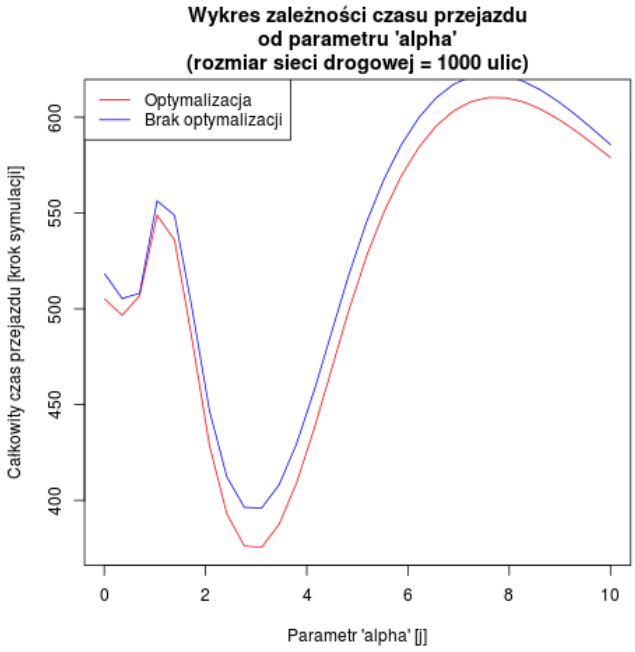
4.3.1 Opis

Trzecia symulacja była powtórzeniem symulacji drugiej, z tą różnica, że rozmiar sieci drogowej wynosił *1000 ulic*.

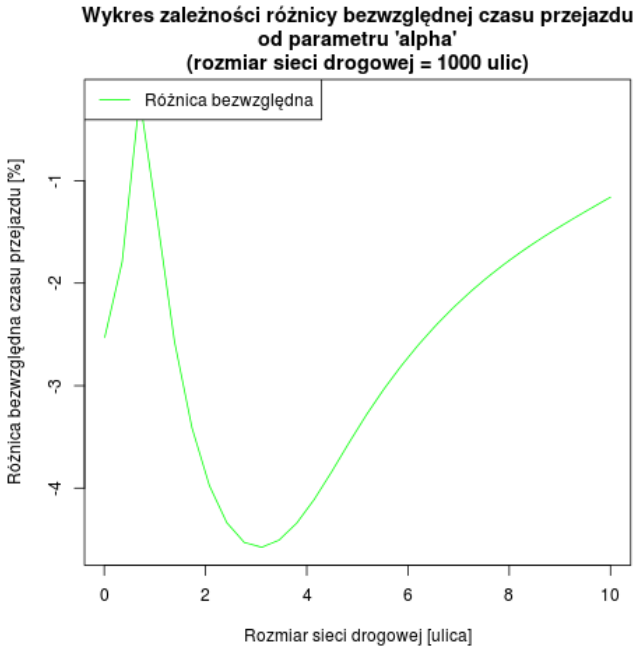
4.3.2 Wyniki

Parametr alpha [j]	Optymalizacja [krok symulacji]	Brak optymalizacji [krok symulacji]
0.01	505.1	518.2
0.02	503.1	502.6
0.05	448.888888888889	455.444444444444
0.1	489.9	490.8
0.2	522.1	523.8
0.5	489.8	493.6
1	546.1	552.6
2	439.9	457.6
5	513.4	531.6
10	578.9	585.7

Tabela 3: Zależność czasu przejazdu od parametru *alpha*'.



Rysunek 7: Zależność czasu przejazdu od parametru *alpha*.



Rysunek 8: Zależność bezwzględnej różnicy czasów przejazdu od parametru *alpha*.

4.3.3 Analiza wyników

Symulacja trzecia podobnie jak symulacja druga sprawdzała wpływ parametru *alpha* na sumaryczny czas podróży. Tym razem okazało się jednak, że najkorzystniejsze wyniki (poprawa czasu przejazdu rzędu 5 procent) są uzyskiwane dla parametru *alpha* wynoszącego od 2 do 3. Pozwala to stwierdzić, że wraz ze wzrostem rozmiaru sieci drogowej należy uwzględniać mniejsze zmiany czasu przejazdu przez krawędzie. Wynika to z kształtowania się ruchu drogowego. Im większa sieć drogowa tym mniejsze prawdopodobieństwo powstawania zatorów i oczekiwania na skrzyżowaniach.

4.4 Symulacja 4

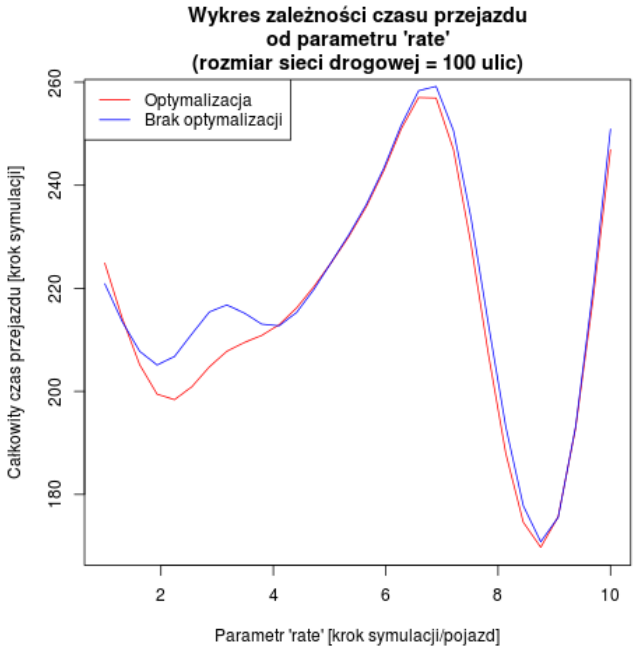
4.4.1 Opis

Czwarta symulacja badała zależność czasu przejazdu od częstotliwości pojawiania się samochodów w sieci drogowej, a co za tym idzie od natężenia ruchu. Rozmiar sieci drogowej wynosi *100 ulic*, parametr *alpha 0.05*, a maksymalna długość krawędzi *500 metrów*.

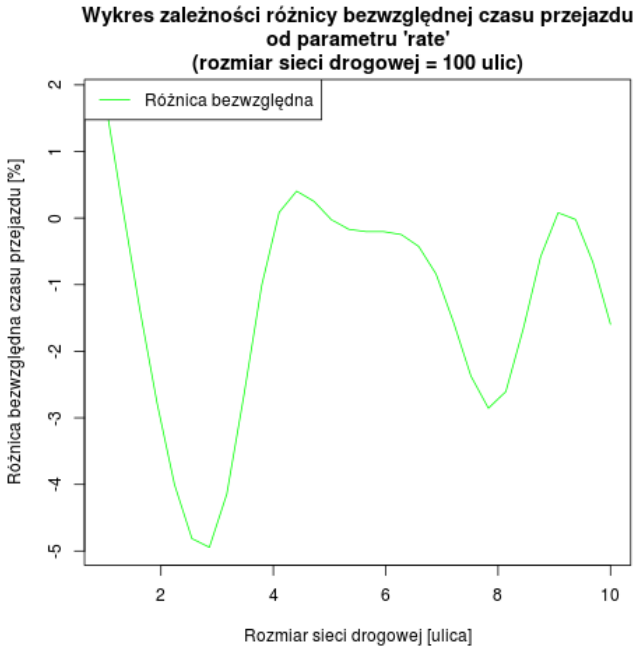
4.4.2 Wyniki

Parametr 'rate' [krok symulacji/pojazd]	Optymalizacja [krok symulacji]	Brak optymalizacji [krok symulacji]
1	224.9	220.9
2	198.8	205.1
3	206.3	216.5
4	212.1	212.5
5	224.5	224.5
6	243.7	244.2
7	254.8	257.5
8	195.9	201.6
9	173.3	173.3
10	246.9	250.9

Tabela 4: Zależność czasu przejazdu od parametru *rate*.



Rysunek 9: Zależność czasu przejazdu od parametru *rate*.



Rysunek 10: Zależność bezwzględnej różnicy czasów przejazdu od parametru *rate*.

4.4.3 Analiza wyników

Parametr *rate* określał częstotliwość pojawiania się nowych pojazdów w sieci drogowej, a co za tym idzie wpływał na natężenie ruchu drogowego w sieci. Uzyskane wyniki pozwalają stwierdzić, że dla grafów rozmiaru około stu krawędzi średnie czasy przejazdu są najlepsze dla wartości parametru *rate* wynoszącej około 2 do 3 kroki symulacji na pojazd. Wynika to z trudności znalezienia lepszej trasy dojazdu dla dużego natężenia ruchu drogowego. Z kolei niewielki ruch w sieci drogowej nie powoduje powstawania zatorów i nie wymaga optymalizacji trasy dojazdu.

4.5 Symulacja 5

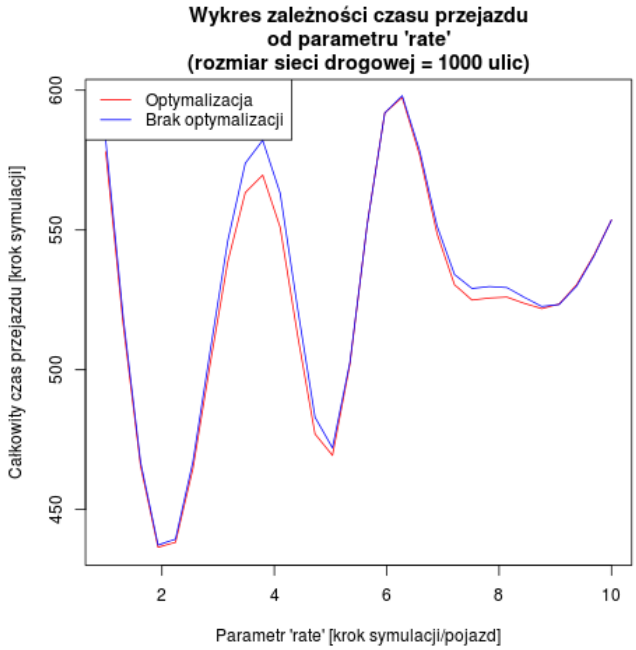
4.5.1 Opis

Piąta symulacja była powtórzeniem symulacji czwartej, z tą różnica, że rozmiar sieci drogowej wynosił *1000 ulic*.

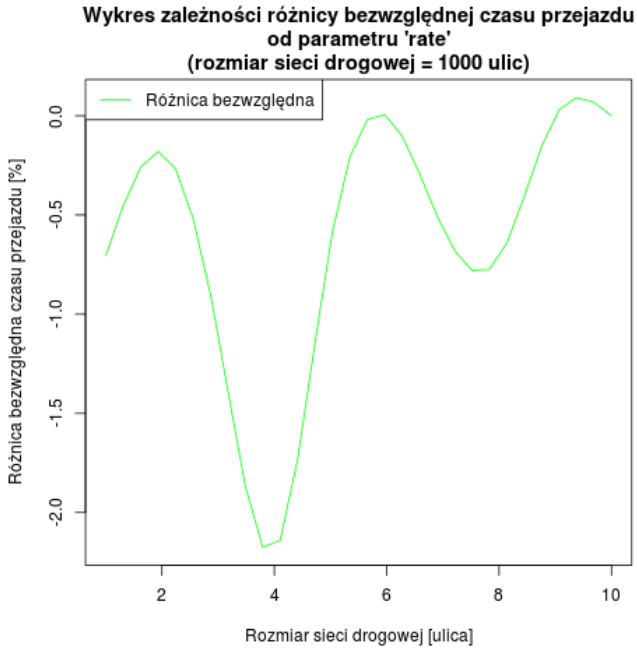
4.5.2 Wyniki

Parametr 'rate' [krok symulacji/pojazd]	Optymalizacja [krok symulacji]	Brak optymalizacji [krok symulacji]
1	578	582.1
2	434.1	434.9
3	519.6	525.4
4	560.3	572.9
5	468.1	471.1
6	594.4	594.4
7	540.8	543.9
8	526.3	530.1
9	522.7	522.7
10	553.6	553.6

Tabela 5: Zależność czasu przejazdu od parametru *rate*.



Rysunek 11: Zależność czasu przejazdu od parametru *rate*.



Rysunek 12: Zależność bezwzględnej różnicy czasów przejazdu od parametru *rate*.

4.5.3 Analiza wyników

Symulacja piąta podobnie jak symulacja czwarta sprawdzała wpływ parametru *rate* na sumaryczny czas podróży. Wyniki otrzymane w tej symulacji potwierdzają rezultaty poprzedniej symulacji. Ponownie okazuje się, że próba optymalizacji trasy w sposób dynamiczny najkorzystniej wpływa na czas przejazdu dla średniego natężenia ruchu drogowego i daje poprawę rzędu trzech procent.

4.6 Symulacja 6

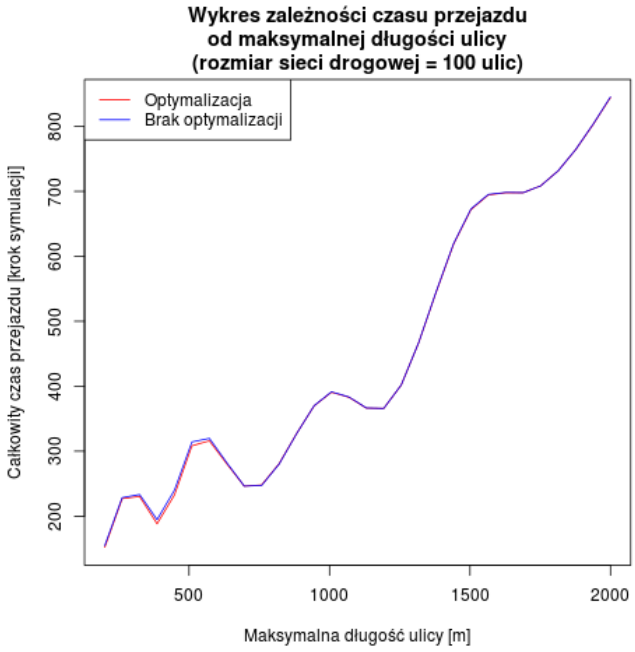
4.6.1 Opis

Szósta symulacja miała na celu zbadanie wpływu długości ulic, a co za tym idzie struktury sieci drogowej, na czas podróży. Rozmiar sieci drogowej wynosił *100 ulic*, parametr *alpha 0.05*, a częstotliwość pojawiania się samochodów w sieci drogowej *2 kroki symulacji/pojazd*.

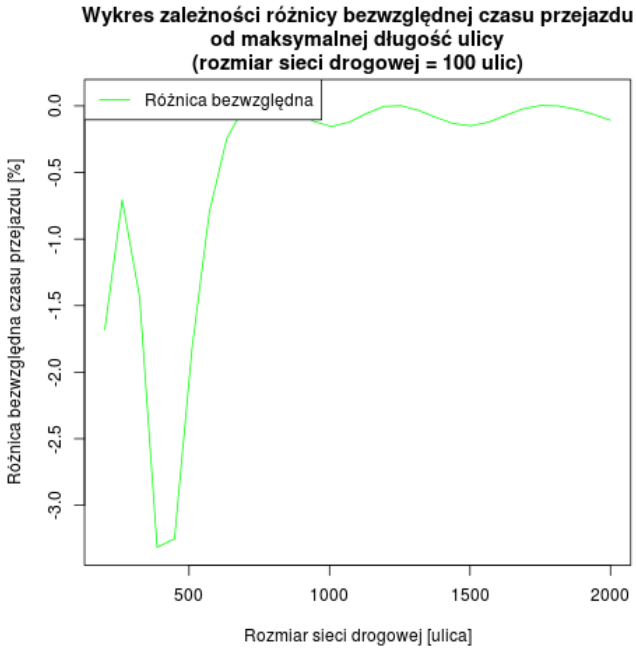
4.6.2 Wyniki

Maksymalna długość ulicy [m]	Optymalizacja [krok symulacji]	Brak optymalizacji [krok symulacji]
200	152.1	154.7
300	241.1	243.2
400	188.2	195.1
500	299.9	306.2
700	245.2	245.2
1000	390	390.6
1200	368	368
1500	669.7	670.7
1700	698.7	698.8
2000	844.5	845.4

Tabela 6: Zależność czasu przejazdu od maksymalnej długości ulicy.



Rysunek 13: Zależność czasu przejazdu od maksymalnej długości ulicy.



Rysunek 14: Zależność bezwzględnej różnicy czasów przejazdu od maksymalnej długości ulicy.

4.6.3 Analiza wyników

Analiza otrzymanych wyników pozwala stwierdzić, że optymalizacja trasy w trybie online wpływa korzystnie na średnie czasy przejazdu. Najlepsze rezultaty dla grafów wielkości około stu krawędzi są uzyskiwane dla stosunkowo krótkich ulic rzędu 300 metrów. Wynika to z faktu, że dla sieci drogowych o zwartej strukturze prawdopodobieństwo powstawania zatorów jest znaczące, a co za tym idzie próby optymalizacji trasy dają korzystniejsze rezultaty.

4.7 Symulacja 7

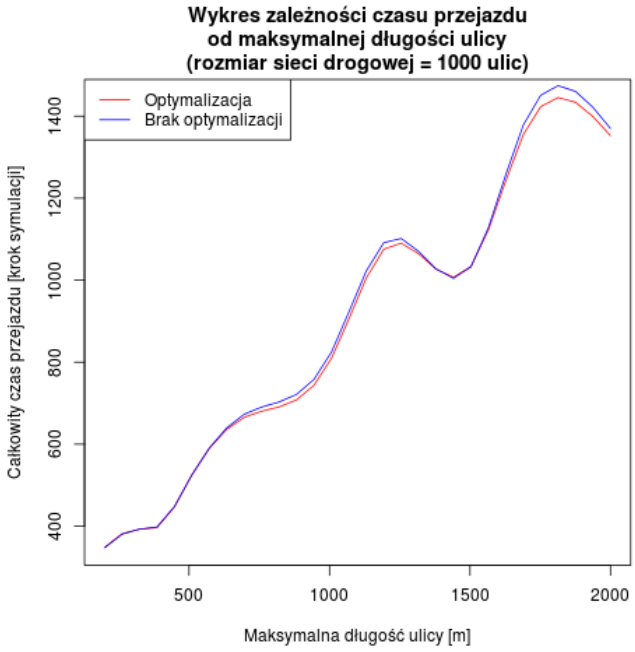
4.7.1 Opis

Siódma symulacja była powtórzeniem symulacji szóstej, z tą różnica, że rozmiar sieci drogowej wynosił *1000 ulic*.

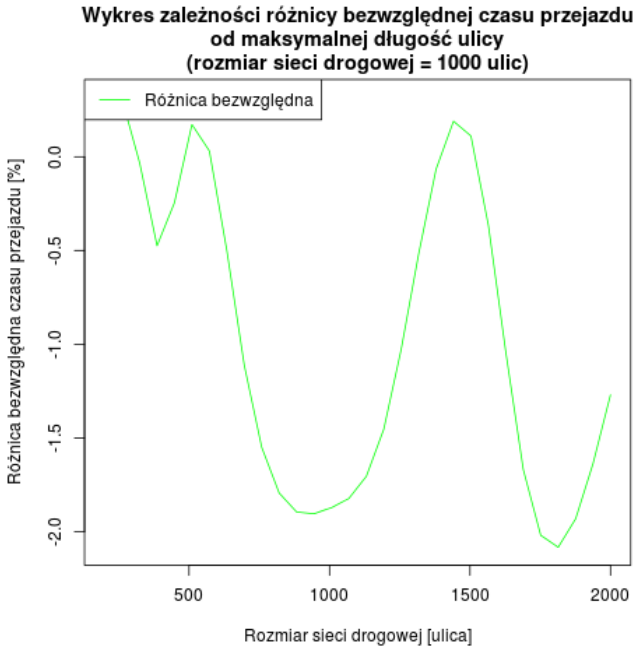
4.7.2 Wyniki

Maksymalna długość ulicy [m]	Optymalizacja [krok symulacji]	Brak optymalizacji [krok symulacji]
200	348.6	347.5
300	392.3	391.7
400	402.6	404.6
500	513.2	512.5
700	666.6	674.3
1000	800.1	815.4
1200	1080.1	1095.6
1500	1030.5	1029.2
1700	1369.9	1394.3
2000	1352.4	1369.8

Tabela 7: Zależność czasu przejazdu od maksymalnej długości ulicy.



Rysunek 15: Zależność czasu przejazdu od maksymalnej długości ulicy.



Rysunek 16: Zależność bezwzględnej różnicy czasów przejazdu od maksymalnej długości ulicy.

4.7.3 Analiza wyników

Symulacja siódma podobnie jak symulacja szósta sprawdzała wpływ długości ulic, a co za tym idzie także struktury sieci drogowej, na średni czas przejazdu. Wyniki otrzymane w tej symulacji potwierdzają rezultaty poprzedniej symulacji. Ponownie okazuje się, że próba optymalizacji trasy w sposób dynamiczny najkorzystniej wpływa na czas przejazdu dla sieci o stosunkowo zwartej strukturze i daje poprawę rzędu dwóch procent.

5 Podsumowanie

Na podstawie serii przeprowadzonych symulacji można stwierdzić, że próby optymalizacji trasy w trybie online dają korzystne wyniki. Średnie czasy przejazdu pojazdu, którego trasa jest dynamicznie zmieniana, są około o dwa do pięciu procent lepsze od czasów przejazdu pojazdu poruszającego się niezmienną trasą. Metoda optymalizacyjna uzyskuje najlepsze rezultaty dla średniego rozmiaru sieci drogowych i średniego natężenia ruchu drogowego.

Należy pamiętać, że przeprowadzone badania miały charakter statystyczny, który wynikał z trudności modelowania nieustannie zmieniającego się ruchu drogowego. Stąd może się okazać, że dla danej sieci i ruchu drogowego metoda optymalizacyjna zakończy się niekorzystnym rezultatem. Jednak dla znakomitej większości przypadków opisana metoda pozwala przebyć zadaną trasę w krótszym czasie, niż gdy nie jest stosowana.

6 Wykorzystane narzędzia i języki programowania

W projekcie wykorzystano następujące narzędzia:

SUMO - symulator ruchu drogowego

Python 2.7 - implementacja modułu symulacyjnego

R - graficzna prezentacja wyników

Visual Paradigm 10.2 - diagramy klas oraz komponentów

LaTeX - formatowanie oraz skład dokumentacji

7 Bibliografia

1. *SUMO* http://sumo-sim.org/
2. *TraCI* http://sumo-sim.org/pydoc/traci.html
3. *Netgenerate* http://sumo-sim.org/userdoc/NETGENERATE.html
4. *randomTrips* http://sumo-sim.org/userdoc/Tools/Trip.html#randomTrips.py
5. *Algorytm Dijkstry* http://pl.wikipedia.org/wiki/Algorytm_Dijkstry