

# Optymalizacja struktury sieci drogowej

Michał Siatkowski

Promotor: dr hab. inż. Aneta Poniszewska - Marańda  
Kopromotor: mgr inż. Łukasz Chomątek

Politechnika Łódzka

Łódź, FTIMS, Informatyka 2014/2015

# Problematyka i zakres pracy

Niniejsza praca obejmuje zagadnienia z zakresu inżynierii oprogramowania i sztucznej inteligencji. Głównym jej celem jest stworzenie aplikacji optymalizującej strukturę sieci drogowej.

# Cele pracy

Celami pracy są:

- ① Zdefiniowanie problematyki optymalizacji struktury sieci drogowej.
- ② Stworzenie aplikacji optymalizującej tę strukturę.
- ③ Analiza i ocena efektywności zastosowanych rozwiązań.

# Metoda badawcza

## Prototypowanie

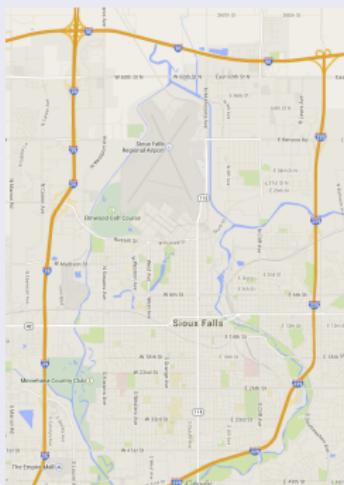
*jest to proces budowy modelu matematycznego i obserwacja czy jego zachowanie może pomóc inżynierom w odkryciu ukrytych wad ich projektu. Z założenia prototypy nie wchodzą w skład ostatecznego systemu.*

# Przegląd literatury w dziedzinie

- ① Wataru Nanya, Hiroshi Kitada, Azusa Hara, Yukiko Wakita, Tatsuhiro Tamaki, and Eisuke Kita  
*Road Network Optimization for Increasing Traffic Flow.*  
Int. Conference on Simulation Technology, JSST 2013.

# Podstawowe definicje

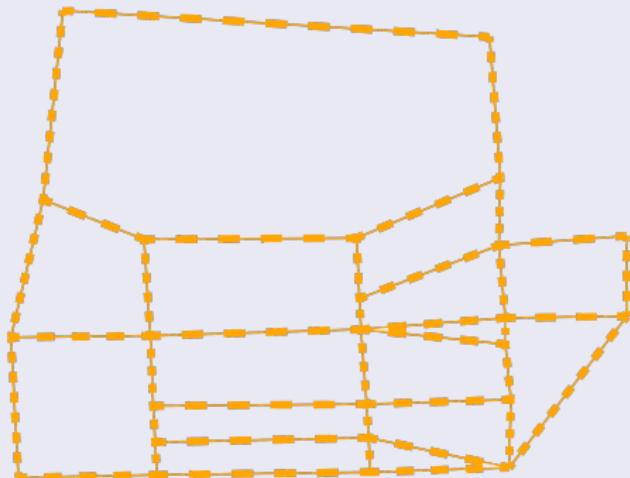
## Sieć drogowa



Rysunek 1 : Fragment sieci drogowej w Sioux Falls, Południowa Dakota.

# Podstawowe definicje

## Sieć drogowa w postaci grafu



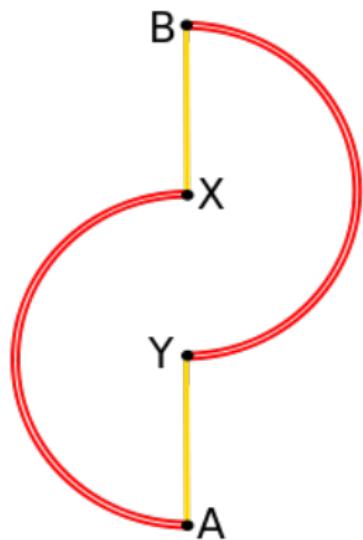
Rysunek 2 : Sieć drogowa miasta Sioux Falls w postaci grafu.

## Podstawowe definicje



Rysunek 3 : Graf z dopasowaną geometrią [13].

## Paradoks Braessa[14]



Rysunek 4 : Wyjściowy układ drogowy

Autostrady:  
 $AX, t_{AX}(p) = 50 + p \text{ min}$   
 $YB, t_{YB}(p) = 50 + p \text{ min}$

Drogi lokalne:  
 $AY, t_{AY}(p) = 10p \text{ min}$   
 $XB, t_{XB}(p) = 10p \text{ min}$

Aut jest 6000 i wszystkie mają za zadanie przejechać trasę z A do B.

## Równowaga Nasha[14]

Równowaga Nasha to taka sytuacja, w której każdy z samochodów spowoduje wydłużenie swojego czasu jazdy, zmieniając decyzję co do wyboru trasy przy niezmienionych decyzjach pozostałych aut.

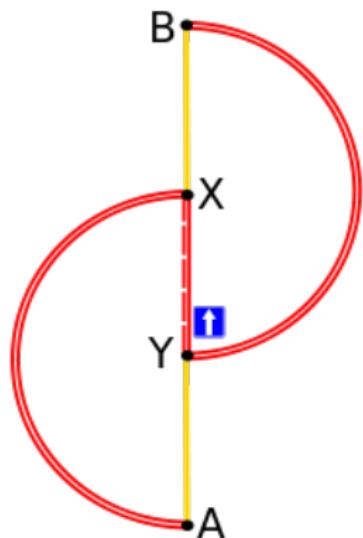
Jeśli  $p$  i  $q$  to liczby aut w tysiącach pokonujących odpowiednio trasy AXB i AYB, otrzymujemy równania:

$$\begin{aligned} p + q &= 6 \\ t_{AX}(p) + t_{XB}(p) &= t_{AY}(q) + t_{YB}(q) \\ 50 + p + 10p &= 10q + 50 + q \end{aligned}$$

rozwiązaniem jest  $p = q = 3$ .

Przy tej gęstości ruchu pokonanie obu dostępnych tras zabiera  $50 + 3 + 30 = 83$  minuty.

## Uzupełniony układ drogowy [14]



Do wyjściowego układu drogowego dodana zostaje autostrada:

$$YX, t_{YX}(p) = 10 + p \text{ min}$$

Aut jest nadal 6000 i wszystkie mają za zadanie przejechać trasę z A do B.

Rysunek 5 : Uzupełniony układ drogowy

## Równowaga Nasha dla uzupełnionego układu[14]

Jeśli  $p$ ,  $q$  i  $r$  to liczby aut w tysiącach pokonujących odpowiednio trasy AXB, AYB i AYXB, otrzymujemy równania:

$$\begin{aligned} p + q + r &= 6 \\ t_{AX}(p) + t_{XB}(p + r) &= t_{AY}(q + r) + t_{YB}(q) = \\ t_{AY}(q + r) + t_{YX}(r) &+ t_{XB}(p + r) \end{aligned}$$

$$50 + p + 10(p + r) = 10(q + r) + 50 + q = 10(q + r) + 10 + r + 10(p + r)$$

rozwiązaniem jest  $p = q = r = 2$ .

Czas przejazdu każdej z tych dróg wynosi wówczas

$$50 + 2 + 10(2 + 2) = 92 \text{ minuty.}$$

## Słabe punkty istniejących rozwiązań

Paradoks Braessa został sformułowany w roku 1970, a od roku 1996 zaczęły pojawiać się prace negujące lub podważające paradoks[2]. Wiele miast jednak brało i bierze pod uwagę paradoks Braessa podczas projektowania swojej przestrzeni:

- Korea, Seul, likwidacja m.in. estakad Cheonggyecheon,
- Niemcy, Stuttgart, likwidacja dróg zbudowanych w latach 60,
- USA, Nowy Jork, czasowe zamknięcie ulicy 42,
- USA, Winnipeg.[15]

## Symulator transportu



Rysunek 6 : Logo symulatora transportu MATSim [5]

## Przestrzeń poszukiwań

Najlepszego rozwiązania będę poszukiwał wykorzystując klasyczny algorytm genetyczny.

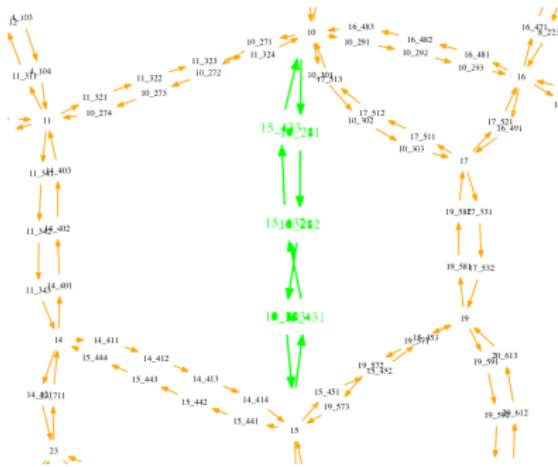


Rysunek 7 : Logo biblioteki Apache Commons Math [6]

## Klasyczny algorytm genetyczny

1	1	0	0	0	0	0
---	---	---	---	---	---	---

### Rysunek 8 : Fragment sieci w postaci tablicy binarnej



### Rysunek 9 : Fragment sieci w postaci grafu

# Technologie i metodologie programistyczne



Rysunek 10 : Logo Java[7]



Rysunek 12 : Logo Python[9]



Rysunek 11 : Logo IDE Eclipse[8]



Rysunek 13 : Logo PyDev[10]

# Dane wejściowe

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <project>
        <name>siouxfalls</name>
        <output-dir>../output/</output-dir>
        <threads>4</threads>
        <log-level>INFO</log-level>
        <python-path>/usr/local/bin/python</python-path>
    </project>
    <scenario>
        <config>../siouxfalls/config.xml</config>
        <network>../siouxfalls/network.xml</network>
        <population>../siouxfalls/population.xml</population>
        <facilities>../siouxfalls/facilities.xml</facilities>
        <iterations>50</iterations>
    </scenario>
    <genetics>
        <population-size>35</population-size>
        <max-generations>150</max-generations>
        <elitism-rate>0.15</elitism-rate>
```

## Wdrożenie

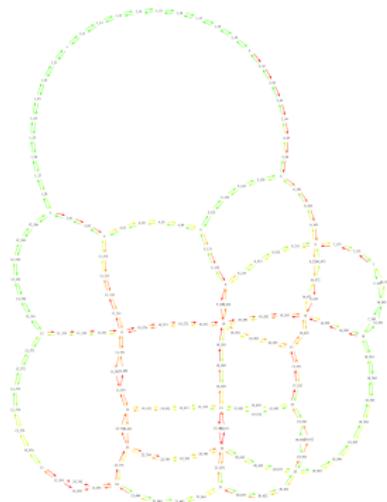


Rysunek 14 : Logo Trisquel[11]

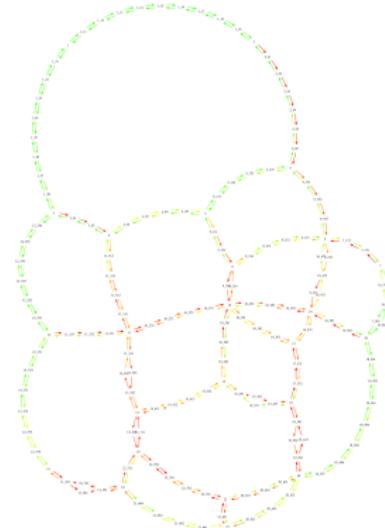
# Przewidywane problemy

- Brak gwarancji lepszego wyniku
- Długi czas obliczeń
- Duże wymagania dotyczące zasobów

# Natężenie ruchu

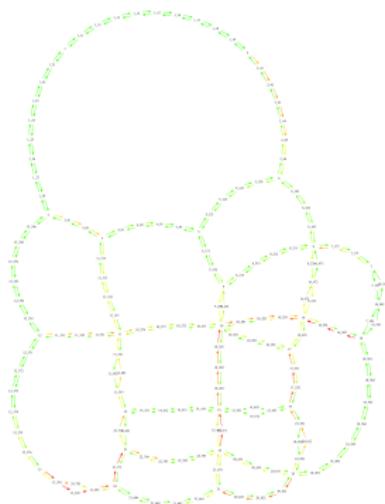


Rysunek 15 : Ruch, 6.00-7.00,  
graf oryginalny

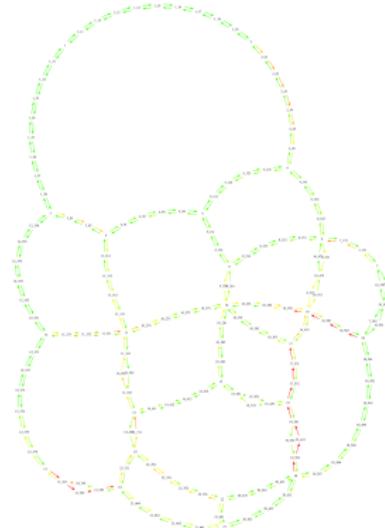


Rysunek 16 : Ruch, 6.00-7.00,  
graf zmodyfikowany

## Natężenie ruchu

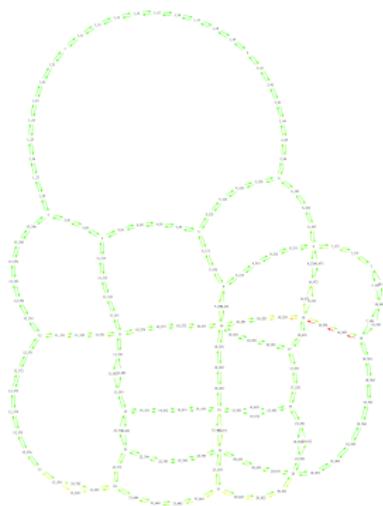


Rysunek 17 : Ruch, 7.00-8.00,  
graf oryginalny

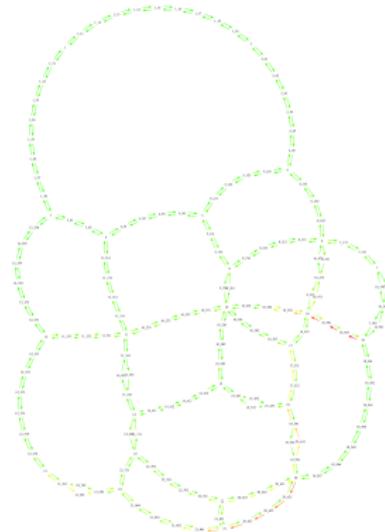


Rysunek 18 : Ruch, 7.00-8.00,  
graf zmodyfikowany

## Natężenie ruchu

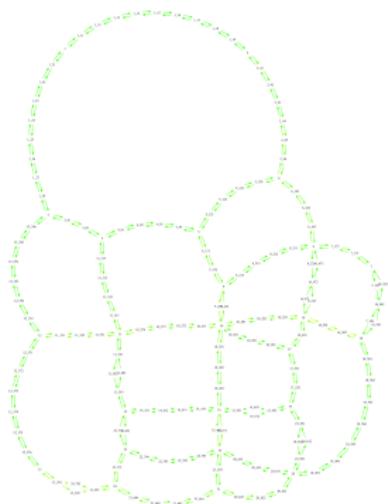


Rysunek 19 : Ruch, 8.00-9.00,  
graf oryginalny

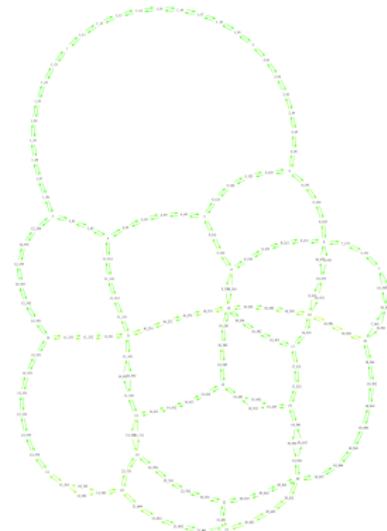


Rysunek 20 : Ruch, 8.00-9.00,  
graf zmodyfikowany

## Natężenie ruchu

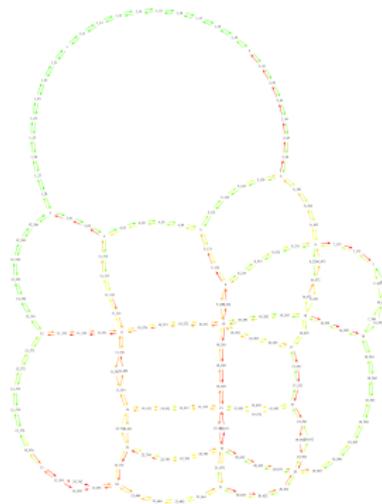


Rysunek 21 : Ruch, 15.00-16.00,  
graf oryginalny

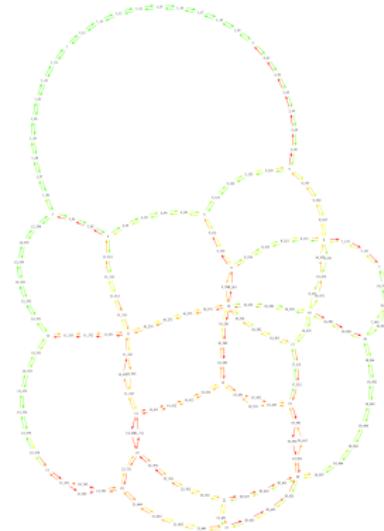


Rysunek 22 : Ruch, 15.00-16.00,  
graf zmodyfikowany

## Natężenie ruchu

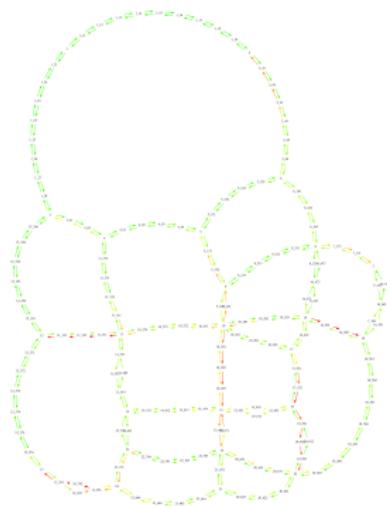


Rysunek 23 : Ruch, 16.00-17.00,  
graf oryginalny

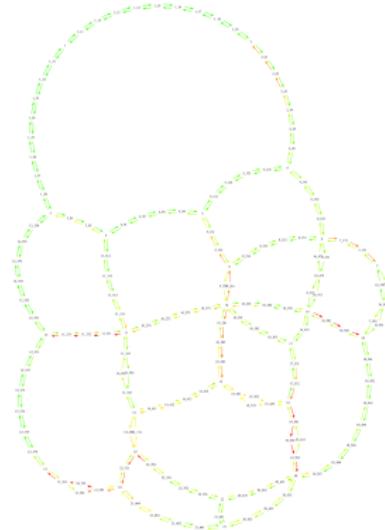


Rysunek 24 : Ruch, 16.00-17.00,  
graf zmodyfikowany

## Natężenie ruchu

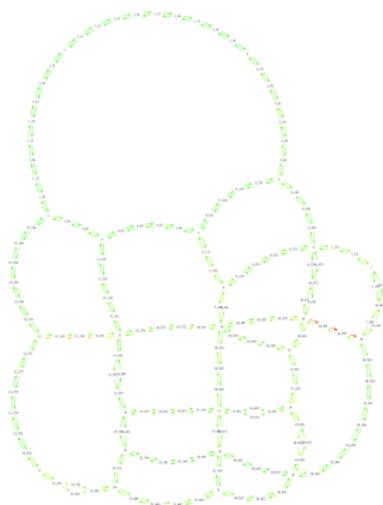


Rysunek 25 : Ruch, 17.00-18.00,  
graf oryginalny

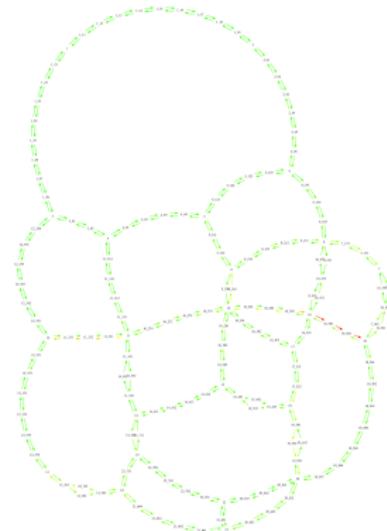


Rysunek 26 : Ruch, 17.00-18.00,  
graf zmodyfikowany

## Natężenie ruchu

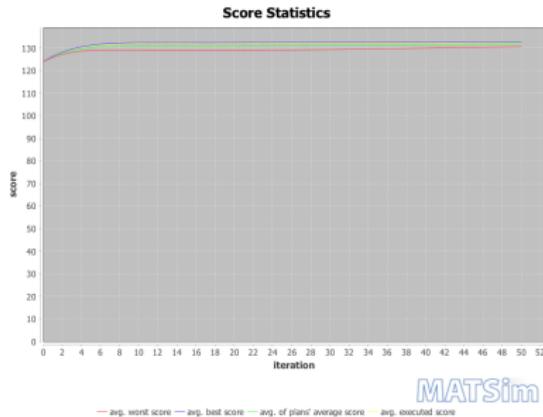


Rysunek 27 : Ruch, 18.00-19.00,  
graf oryginalny

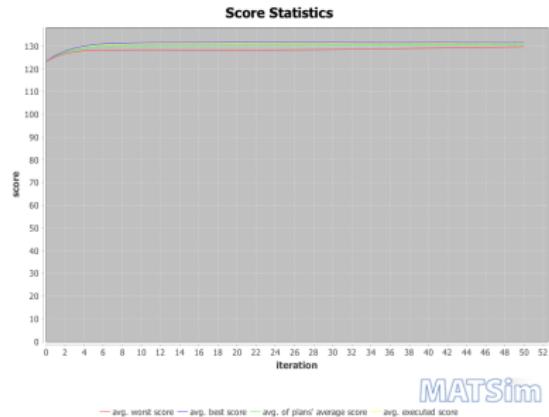


Rysunek 28 : Ruch, 18.00-19.00,  
graf zmodyfikowany

# Wyniki agentów

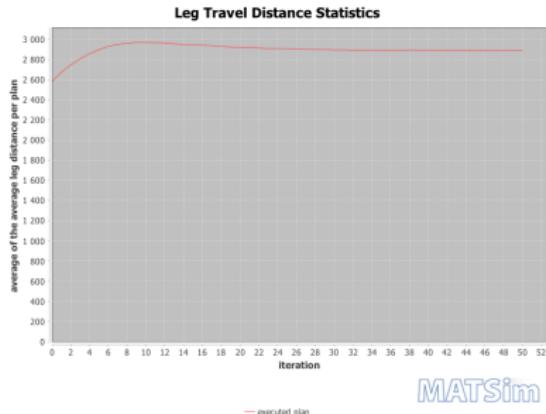


Rysunek 29 : Wyniki agentów,  
graf oryginalny

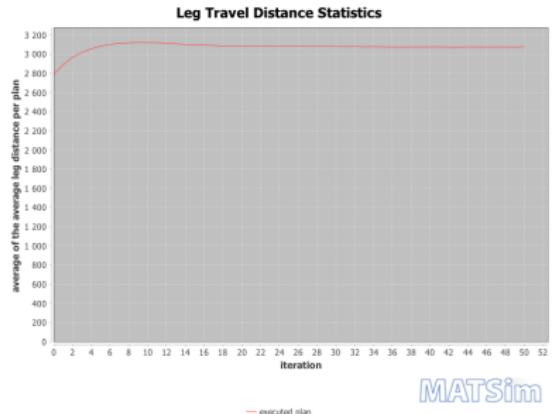


Rysunek 30 : Wyniki agentów,  
graf zmodyfikowany

# Przebyta droga



Rysunek 31 : Przebyta droga, graf oryginalny



Rysunek 32 : Przebyta droga, graf zmodyfikowany

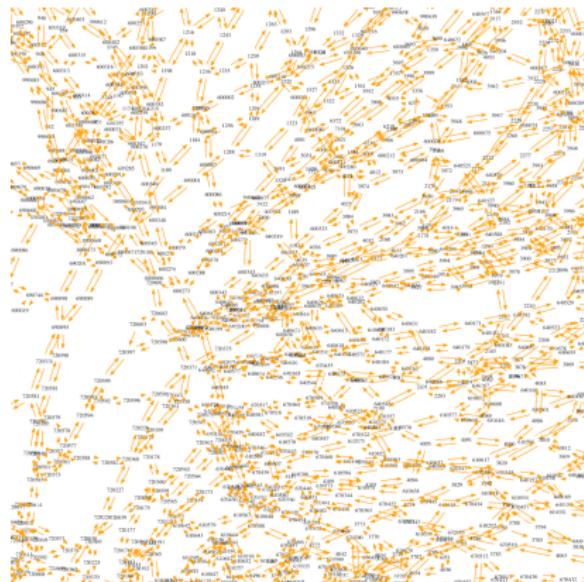
# Średni czas podróży

Graf oryginalny: 1326.6473784330044 [sekund]

Graf zmodyfikowany: 1432.6902865295447 [sekund]

Wynik o  $\approx 8\%$  gorszy.

# Perspektywy dalszych badań w dziedzinie



Rysunek 33 : Część mapy Berlina

# Perspektywy dalszych badań w dziedzinie



Rysunek 34 : Mapa Berlina

# Bibliografia I

-  **Leslie Arthur Keith Bloy,**  
*An investigation into Braess' paradox*, 02/2007
-  **Ric Pas and Shari Principio**  
*Braess' paradox: Some new insights*, April 1996
-  **Wataru Nanya, Hiroshi Kitada, Azusa Hara, Yukiko Wakita, Tatsuhiro Tamaki, and Eisuke Kita**  
*Road Network Optimization for Increasing Traffic Flow*  
*Int. Conference on Simulation Technology, JSST 2013.*
-  **Ana L. C. Bazzan and Franziska Klügl**  
*Reducing the Effects of the Braess Paradox with Information Manipulation*

## Bibliografia II

-  <http://matsim.org>
-  <http://commons.apache.org/proper/commons-math>
-  <http://www.java.com/pl/>
-  <https://eclipse.org>
-  <http://pl.python.org>
-  <http://pydev.org>
-  <https://trisquel.info>

## Bibliografia III

-  M. Rieser, C. Dobler, T. Dubernet, D. Grether, A. Horni, G. Lammel, R. Waraich, M. Zilske, Kay W. Axhausen, Kai Nagel  
*MATSim User Guide*  
updated September 12, 2014
-  A. Chakirov  
*Enriched Sioux Falls Scenario with Dynamic Demand*  
MATSim User Meeting, Zurich/Singapore, June 2013.
-  [http://pl.wikipedia.org/wiki/Paradoks\\_Braessa](http://pl.wikipedia.org/wiki/Paradoks_Braessa)
-  <http://urbnews.pl/paradoks-braessa/>