



Politechnika Łódzka

Instytut Informatyki

PRACA DYPLOMOWA MAGISTERSKA

Optymalizacja struktury sieci drogowej

Optimization of structures of road networks

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Promotor: dr hab. inż. Aneta Poniszewska-Marańda

Kopromotor: mgr inż. Łukasz Chomątek

Dyplomant: inż. Michał Siatkowski

Nr albumu: 186865

Kierunek: Informatyka

Specjalność: Sztuczna Inteligencja i Inżynieria Oprogramowania

Łódź 20.01.2015

Instytut Informatyki

90-924 Łódź, ul. Wólczańska 215, budynek B9

tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.lodz.pl



Spis treści

1 Wstęp.	5
1.1 Problematyka i zakres pracy.	5
1.2 Metoda badawcza i cel pracy.	6
1.3 Przegląd literatury w dziedzinie.	7
1.4 Cel pracy.	8
1.5 Układ pracy.	8
2 Optymalizacja struktury sieci drogowej.	9
2.1 Podstawowe definicje.	9
2.2 Sieć drogowa w postaci grafu	10
2.3 Paradoks Braessa.	12
2.3.1 Przykładowy, wyjściowy układ drogowy.	12
2.3.2 Przykładowy, uzupełniony układ drogowy.	13
2.3.3 Wyjaśnienie intuicyjne.	14
2.4 Graf skierowany silnie spójny.	14
2.5 Klasyczny algorytm genetyczny.	16
2.5.1 Podstawowe pojęcia algorytmów genetycznych	16
2.5.2 Algorytm klasycznego algorytmu genetycznego.	17
2.5.3 Operacje klasyczne, krzyżowanie.	17
2.5.4 Operacje klasyczne, mutacja.	17
2.6 Użycie grafów w algorytmie genetycznym.	18
2.7 Punkty artykulacji grafu.	20
3 Technologie i metody użyte.	21
3.1 Symulator transportu.	21
3.2 Algorytmy genetyczne.	22
3.3 Obsługa grafów.	22
3.4 Technologie i metodologie programistyczne.	22
3.5 Wdrożenie.	23

SPIS TREŚCI

3.6 System obliczeniowy.	24
4 Opis projektu.	25
4.1 Dane wejściowe.	25
4.2 Badane miasto przykładowe: Sioux Falls	27
4.3 Ustawienia symulatora.	31
4.4 Ustawienia projektu.	33
4.5 Wyniki.	34
4.6 Analiza symulacji	45
5 Podsumowanie.	49
5.1 Dyskusja wyników.	49
5.2 Analiza wyników.	50
5.3 Perspektywy dalszych badań w dziedzinie.	52
5.4 Struktura projektu.	53
Spis rysunków	55
Spis tabel	57
Spis listingów	59
Bibliografia	61
Abstract	63

Rozdział 1

Wstęp.

W rozdziale 1 zostały pokrótko opisane motywy wyboru tematu pracy dyplomowej i omówiono najważniejsze aspekty tej dziedziny. Ponadto w rozdziale umieszczone sformułowane cele pracy.

1.1 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.

Problemy komunikacji w dzisiejszych miastach są wszystkim znane. Zatory drogowe i korki w godzinach szczytu są chlebem powszednim. Pomimo wielu prób i sposobów, wciąż nie istnieje metoda jednoznacznie rozwiązująca tę kwestię. Bezspornie, dotyczy to wszystkich miast na świecie. Z teoretycznego punktu widzenia, jedynym rozwiązaniem jest komunikacja publiczna. Oczywistym jest jednak, że nigdy nie doprowadzimy do sytuacji, gdy wszyscy mieszkańcy zrezygnują ze swoich pojazdów. Dodatkowo, wiele osób i usług wymaga oddzielnej formy transportu. W obliczu tych faktów miasta decydują się na rozwój swojej infrastruktury drogowej. Budowa nowych tras oraz poszerzanie starych przynosi nadzieję mniejszych zatorów, a co za tym idzie, szybszego przejazdu do celu. Niestety, historia pokazuje, że takie inwestycje nie zawsze przynoszą oczekiwane korzyści.

Teorii próbujących解释 te zjawiska, jak również dowodów, które je popierają lub obalają jest wiele. Jedną z najpopularniejszych oraz taką która została wykorzystana w niektórych miastach na świecie jest paradoks Braessa [21]. Jest to twierdzenie matematyczne orzekające, że w pewnym modelu ruchu drogowego, czasy podróży pojazdów mogą ulec wydłużeniu po dodaniu do sieci drogowej nowego połączenia. Ma ono również

1.2. METODA BADAWCZA I CEL PRACY.

zastosowanie w przypadku sieci komputerowych oraz istnieją jego analogie dla doświadczeń fizycznych.

Celem pracy jest opracowanie metody, która dla danej struktury sieci drogowej, zmodyfikuje ją wykorzystując prawidłowość z powyższego paradoksu. W efekcie poprzez zamknięcie wybranych ulic w danej sieci drogowej, średni czas podróży powinien ulec skróceniu.

1.2 Metoda badawcza i cel pracy.

Studia literaturowe.

Badania rozpoczęliśmy od poszukiwania źródeł traktujących o opisywanym problemie. Paradoks Braessa został sformułowany w 1970 roku i był od tego czasu wykorzystywany przy planowaniu przestrzeni i infrastruktury wielu miast, np:

- Korea, Seul, likwidacja m.in. estakad Cheonggyecheon,
- Niemcy, Stuttgart, likwidacja dróg zbudowanych w latach 60,
- USA, Nowy Jork, czasowe zamknięcie ulicy 42,
- USA, Winnipeg. [22]

Propozycja rozwiązania problemu.

Oczywistym rozwiązaniem problemu komunikacji mogłoby być stworzenie idealnej sieci odpowiadającej potrzebom danego miasta. Rozbudowa lub modyfikacja tej infrastruktury jest jednak kosztowna i czasochłonna. Z tego powodu postanowiliśmy sprawdzić rozwiązanie zaproponowane przez Braessa. Ponieważ istnieją prace negujące lub podważające paradoks [2] , zdecydowaliśmy by przy potwierdzaniu wyniku optymalizacji nie kierować się wyłącznie założeniami zawartymi w twierdzeniu.

Opis zastosowania algorytmów genetycznych.

Ponieważ nie znaleźliśmy żadnych przesłanek wykazujących jednoznaczna ocenę co do słuszności zamknięcia danej ulicy, zdecydowaliśmy się na losowe przeszukiwanie przestrzeni rozwiązań. Jednym z rozwiązań w przypadku takich poszukiwań są algorytmy genetyczne, które zostały wykorzystane w pracy.

Przedstawienie oceny optymalizacji.

Paradoks Braessa zakłada dość oczywiste potwierdzenie swojej wiarygodności. Dlatego, w celu potwierdzenia jego słuszności, zdecydowaliśmy się na zastosowanie zewnętrznego systemu oceny. Taką rolę spełniają systemy symulacji. System, który został wybrany działa zupełnie oddziennie od metody twierdzenia, symulując rzeczywisty ruch na danej sieci drogowej. Wynik symulacji jest jednoznaczną wartością liczbową, przedstawiającą średni czas przejazdów wszystkich agentów biorących udział w danym scenariuszu. Zakładając stały zestaw agentów dla zmieniających się w wyżej opisany sposób sieci, dążymy oczywiście do minimalizacji średniego czasu przejazdu.

Ocena możliwości wdrożenia proponowanych rozwiązań.

Paradoks Braessa nie jest jedynym twierdzeniem traktującym o problemach komunikacyjnych miast. Wiele teorii jest opartych głównie na socjologicznych lub psychologicznych założeniach¹. Są jednak niemniej ważne. Biorąc pod uwagę złożoność problemu, wynik otrzymany podczas eksperymentu nie może być dowodem, ani decydującym głosem w decyzjach dotyczących ustalania rzeczywistej sieci drogowej miasta.

1.3 Przegląd literatury w dziedzinie.

By przybliżyć temat problemów komunikacyjnych i rozwiązania zaproponowanego przez Braessa polecamy pracę magisterską *Leslie Arthur Keith Bloy* [1]. W pracy zostały opisane również inne twierdzenia dotyczące ruchu drogowego. Publikacja *Reducing the Effects of the Braess Paradox with Information Manipulation* [4] jest bardzo dobrym uzupełnieniem tematu o interesującą nas kwestię symulacji wieloagentowych. Prezentuje ona różnice w wynikach dla przypadku losowego wyboru drogi przez agentów oraz tej wybranej przy użyciu inteligencji kolektywnej².

Zbiorowa praca napisana na potrzeby międzynarodowej konferencji dotyczącej technologii symulacji [4] jest pozycją, która opisuje podobny do podejmowanego przez nas problem. Mianowicie skupia się na modyfikacjach drogi poprzez zmiany dostępności jej składowych.

W zupełnie oddzielnej tematyce, algorytmów genetycznych, polecamy pozycję, która jest wstępem do tej tematyki [6]. Pozycja zawiera bogaty zestaw problemów przykładowych wraz z opisem ich rozwiązania. Ponadto dogłębnie opisuje ona każdy aspekt działania zastosowanych algorytmów.

¹np. paradoks Downsa Thomsona [23] czy prawo Lewisa Mogridge'a [24]

²ang. Collective Intelligence (COIN)

1.4 Cel pracy.

Tematem pracy jest optymalizacja struktury sieci drogowej. Za główny cel przyjęto stworzenie rozwiązania, które przy pomocy algorytmów genetycznych dokona optymalizacji sieci wejściowej. Optymalizacja zostanie przeprowadzona wykorzystując pewne ustalone z góry parametry algorytmu genetycznego.

1.5 Układ pracy.

Rozdział 1 zawiera wstęp i cele pracy. W rozdziale 2 przybliżamy istotę optymalizacji struktur sieci drogowych oraz opisujemy teoretycznie, wykorzystane przez nas rozwiązania. W rozdziale 3 wymieniamy technologie, w których wykonaliśmy aplikację oraz przybliżamy zewnętrzne biblioteki, o które oparliśmy swoje rozwiązanie. Głównym rozdziałem pracy jest rozdział 4, w którym przedstawiamy wykonaną aplikację oraz wyniki optymalizacji przykładowej sieci. W rozdziale podsumowującym 5, zawarliśmy wnioski oraz możliwości rozwoju pracy. Po rozdziale 5, podsumowującym, załączamy spis rysunków, tabel, listingów i bibliografie, w tej kolejności.

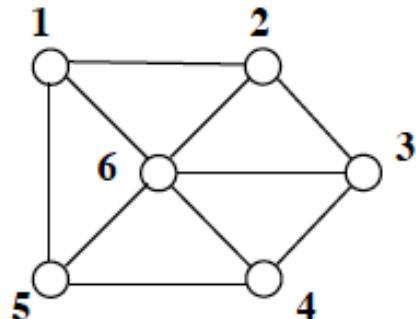
Rozdział 2

Optymalizacja struktury sieci drogowej.

W rozdziale 2 zostały opisane teoretyczne aspekty pracy. Ich źródła opierają się o wybraną literaturę naukową.

2.1 Podstawowe definicje.

Definicja 1 *Grafem (nieskierowanym) nazywamy parę zbiorów (V, E) . Elementy zbioru V nazywają się wierzchołkami, natomiast elementy zbioru E nazywają się krawędziami. Każda krawędź jest parą wierzchołków, tzn. $E \subseteq u, v : u, v \in V$ [9].*



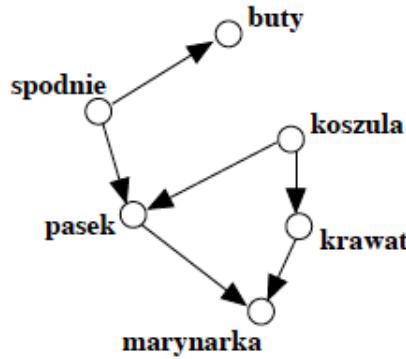
Rysunek 2.1: Przykładowy graf nieskierowany.

Przykładowy graf nieskierowany może zostać opisany przez zbiory:

$$V = \{1, 2, 3, 4, 5, 6\} \quad E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{6, 5\}, \{6, 1\}, \{2, 6\}, \{3, 6\}, \{4, 6\}\}$$

Definicja 2 *Grafem skierowanym nazywamy taki graf, w którym każda krawędź ma zdefiniowany początek i koniec, tzn. żeby pary były uporządkowane. Wtedy $E \subseteq V \times V =$*

$u, v : u, v \in V$. Krawędź (u, v) najłatwiej wyobrazić sobie jako strzałkę od u do v , dlatego często będziemy oznaczać ją przez $u \rightarrow v$ [9].



Rysunek 2.2: Przykładowy graf skierowany.

Przykładowy graf skierowany może zostać opisany przez zbiory:

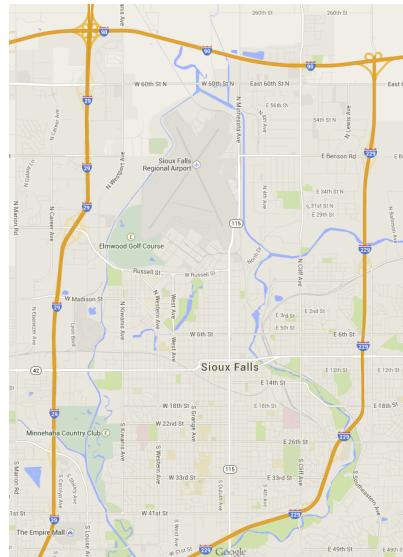
$$V = \{spodnie, buty, pasek, koszula, krawat, marynarka\}$$

$$E = \{\{spodnie \rightarrow buty\}, \{spodnie \rightarrow pasek\}, \{pasek \rightarrow koszula\}, \{koszula \rightarrow krawat\}, \{pasek \rightarrow marynarka\}, \{krawat \rightarrow marynarka\}\}$$

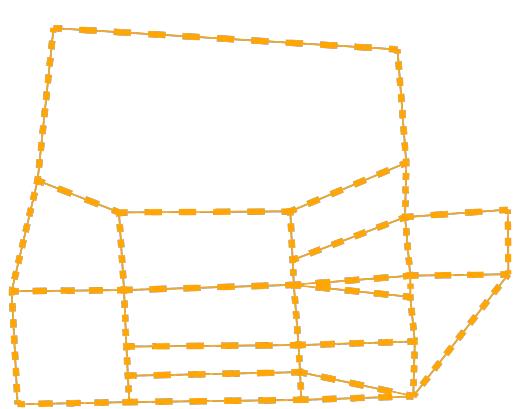
2.2 Sieć drogowa w postaci grafu

W przypadku sieci drogowej mamy oczywiście do czynienia z abstrakcyjną strukturą sieci. Przez sieć drogową rozumiemy bowiem układ dróg lub ulic np. w mieście. Podczas swoich badań posługuję się zawsze pewnym fragmentem większej sieci. Najlepiej te dane zobrazuje poniższy przykład.

Posługując się powyższymi definicjami, tworząc graf z pewnej sieci drogowej, przyjmujemy, że zbiorem V , wierzchołków są skrzyżowania, natomiast zbiór krawędzi E odnosi się do ulic pomiędzy tymi skrzyżowaniami. W pracy posługujemy się zawsze grafem skierowanym. W związku z tym, w przypadku ulic dwukierunkowych tworzone są pary krawędzi z odpowiednimi kierunkami, nawet jeśli ulice nie są rozłączne w rzeczywistości. W celu potwierdzenia wiarygodności powyższego przykładu prezentujemy graf nałożony na mapę miasta [8] na rysunkach 2.4, 2.5.



Rysunek 2.3: Fragment sieci drogowej miasta Sioux Falls, Południowa Dakota.



Rysunek 2.4: Sieć drogowa miasta Sioux Falls w postaci grafu.



Rysunek 2.5: Graf z dopasowaną geometrią.

2.3 Paradoks Braessa.

Jak już wcześniej wspomnieliśmy, jest to twierdzenie matematyczne orzekające, że w pewnym modelu ruchu drogowego czasy podróży pojazdów mogą ulec wydłużeniu po dodaniu do sieci drogowej nowego połączenia. Autorem twierdzenia jest niemiecki matematyk Dietrich Braess [21]. Paradoks działa w oparciu o model ruchu drogowego, który ma następujące cechy:

1. Sieć drogowa składa się ze skończenie wielu węzłów i łączących je odcinków dróg
2. Po sieci porusza się skończenie wiele pojazdów, każdy z nich ma wyznaczony węzeł startowy i węzeł docelowy
3. Odcinki dróg mają przypisane sobie czasy przejazdu, przy czym czasy te mogą zależeć od liczby pokonujących dany odcinek pojazdów.
4. Układ sieci drogowej i czasy przejazdu poszczególnych odcinków są znane pojazdom
5. Celem pojazdów jest przejazd przez sieć z węzłów początkowych do docelowych po trasie złożonej z odcinków drogowych tak, by zminimalizować łączny czas ich pokonania
6. Decyzje o wyborze tras pojazdy podejmują indywidualnie i niezależnie od siebie

Paradoks przedstawiamy w oparciu o przykład z oryginalnego artykułu Dietricha Braessa [5].

2.3.1 Przykładowy, wyjściowy układ drogowy.

Sieć drogowa i auta

Przykład sytuacji, w której ujawnia się paradoks Braessa jest skonstruowany z czterech miast A , B , X i Y . Są one połączone odcinkami drogowymi jak na rysunku i z następującymi czasami przejazdu, przy czym p oznacza gęstość ruchu w tysiącach aut.

Analiza równowagi Nasha

Każde auto musi zdecydować się na wybór trasy albo AXB albo AYB .

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

Autostrady:

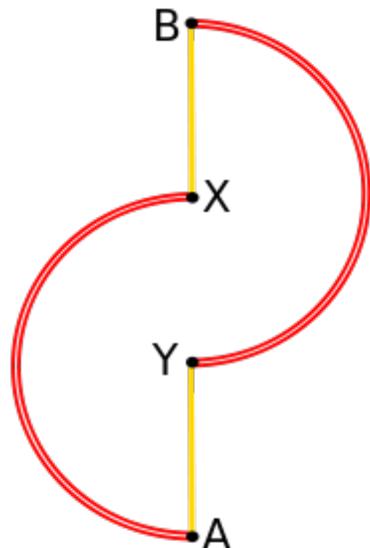
$$AX, t_{AX}(p) = 50 + p \text{ min}$$

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

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.

Rysunek 2.6: Wyjściowy układ drogowy

decyzjach pozostałych aut.

Jeśli p i q to liczby aut w tysiącach pokonujących odpowiednio trasy AXB i AYB , otrzymujmy 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.

2.3.2 Przykładowy, uzupełniony układ drogowy.

Sieć drogowa i auto

Do wyjściowego układu drogowego dodana zostaje autostrada:

Analiza równowagi Nasha

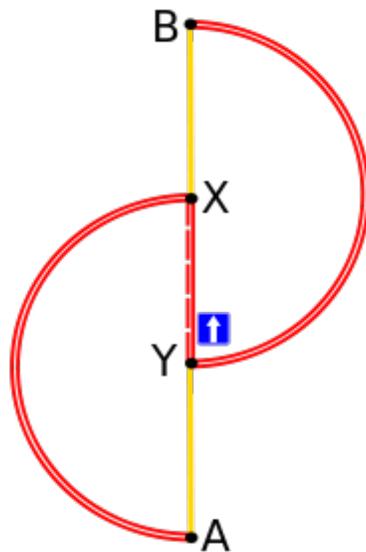
Jeśli p, q i r to liczby aut w tysiącach pokonujących odpowiednio trasy AXB , AYB i $AYXB$, otrzymujmy równania:

2.4. GRAF SKIEROWANY SILNIE SPÓJNY.

Autostrady:

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

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



Rysunek 2.7: Uzupełniony układ drogowy

$$p + q + r = 6$$

$$t_{AX}(p) + t_{XB}(p + r) = t_{AY}(q + r) + t_{YB}(q) = t_{AY}(q + r) + t_{YX}(r) + t_{XB}(p + r)$$

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

rozwiązaniem jest $p = q = r = 2$. Czas przejazdu każdej z tych dróg wynosi wówczas $50 + 2 + 10(2 + 2) = 92$ minuty.

2.3.3 Wyjaśnienie intuicyjne.

Wąskim gardłem systemu są drogi lokalne, na których czas przejazdu bardzo szybko wzrasta wraz z intensywnością ruchu. Po pojawienniu się dodatkowej drogi dostępna staje się nowa trasa, prowadząca oprócz nowego skrótu YX tylko drogami lokalnymi.

Z perspektywy całości systemu nowy odcinek drogowy odciąża ruch na autostradach, gdzie jest to mało odczuwalne, a w zamian jeszcze bardziej zageszcza ruch na drogach lokalnych, powodując wydłużenie czasu podróży.

2.4 Graf skierowany silnie spójny.

Na potrzeby symulatora, sieć drogowa przedstawiona w postaci grafu skierowanego musi spełniać warunek silnej spójności.

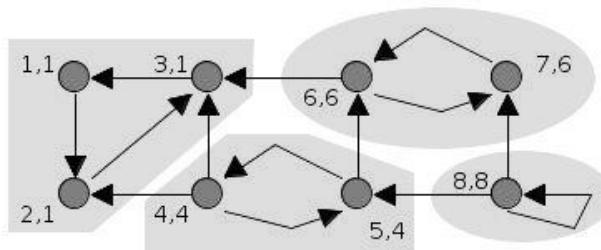
Definicja 3 *Grafem skierowanym silnie spójnym nazywamy graf skierowany w którym jest możliwe dotarcie do każdego wierzchołka zaczynając z dowolnego innego poprzez*

dowolną ilość krawędzi. Wszystkie wierzchołki w grafie skierowanym silnie spójnym muszą zatem posiadać przynajmniej jedną krawędź wchodząjącą i wychodzącą[25].

W sprawdzaniu grafu pod względem spójności, wykorzystujemy algorytm Tarjana do znajdowania składowych silnie spójnych.

Definicja 4 Podstawowym założeniem algorytmu Tarjana jest przeszukiwanie grafu w głąb zaczynając od dowolnego wierzchołka wybranego w sposób arbitralny. Tak jak w przypadku klasycznego przeszukiwania w głąb, każdy sąsiadujący wierzchołek po odwiedzeniu zostaje oznaczony i algorytm nigdy ponownie go nie odwiedza. Dzięki temu, tworzymy kolekcję przeszukanych drzew, która jest drzewem rozpinającym grafu. Składowe silnie spójne są następnie odnajdowane jako poddrzewa, a korzenie tych poddrzew są nazywane korzeniami składowych silnie spójnych. Każdy wierzchołek grafu może być wybrany na korzeń składowej silnie spójnej jeśli zostanie wybrany jako pierwszy wierzchołek podczas przeszukiwania w głąb.

Definicja 5 Składowa silnie spójna (ang. strongly connected component) jest maksymalnym pod grafem, w którym istnieją ścieżki pomiędzy każdym dwoma wierzchołkami. Jeśli pod grafem ten obejmuje wszystkie wierzchołki grafu, to mówimy, że dany graf skierowany jest silnie spójny (ang. strongly connected digraph). W grafach nieskierowanych każdy graf spójny jest również silnie spójny.



Rysunek 2.8: Przykładowy graf z zaznaczonymi składowymi silnie spójnymi.

W efekcie, zawsze przed rozpoczęciem symulacji sprawdzamy, czy graf jest grafem skierowanym silnie spójnym lub dokładniej mówiąc, czy posiada tylko jedną składową silnie spójną.

2.5 Klasyczny algorytm genetyczny.

Idea algorytmu genetycznego została zaczerpnięta z nauk przyrodniczych opisujących zjawiska doboru naturalnego i dziedziczenia. Mechanizmy te polegają na przetrwaniu osobników najlepiej dostosowanych w danym środowisku, podczas gdy osobniki gorzej przystosowane są eliminowane. Z kolei te osobniki, które przetrwają - przekazują informacje genetyczną swoim potomkom. Krzyżowanie informacji genetycznej otrzymanej od „rodziców” prowadzi do sytuacji, w której kolejne pokolenia są przeciętnie coraz lepiej dostosowane do warunków środowiska; mamy tu więc do czynienia ze swoistym procesem optymalizacji.

2.5.1 Podstawowe pojęcia algorytmów genetycznych

Definicja 6 *Populacja* nazywamy zbiór osobników o określonej liczebności.

Definicja 7 *Osobnikami* populacji w algorytmach genetycznych są zakodowane w postaci chromosomów zbiory parametrów zadania, czyli rozwiązania, określone też jako punkty przestrzeni poszukiwań. Osobniki czasami nazywa się organizmami.

Definicja 8 *Chromosomy* to inaczej łańcuchy lub ciągi kodowe, to uporządkowane ciągi genów.

Definicja 9 *Gen* – nazywany też cechą, znakiem, detektorem – stanowi pojedynczy element genotypu, w szczególności chromosomu. *Genotyp*, czyli struktura, to zespół chromosomów danego osobnika. Zatem osobnikami populacji mogą być genotypy albo pojedyncze chromosomy.

Definicja 10 *Fenotyp* jest zestawem wartości odpowiadających danemu genotypowi, czyli zdekodowana struktura, a więc zbiorem parametrów zadania (rozwiązaniem, punkt przestrzeni poszukiwań).

Definicja 11 *Allel* to wartość danego genu, określona jako wartość cechy lub wariant cechy.

Definicja 12 *Locus* to pozycja - wskazuje miejsce położenia danego genu w łańcuchu, czyli chromosomie.

Definicja 13 *Funkcja przystosowania* nazywana też *funkcją dopasowania lub funkcją oceny*. Stanowi ona miarę przystosowania (dopasowania) danego osobnika w populacji.

2.5.2 Algorytm klasycznego algorytmu genetycznego.

Na podstawowy (klasyczny) algorytm genetyczny, nazywany także elementarnym lub prostym algorytmem genetycznym, składają się kroki:

1. inicjacja czyli wybór początkowej populacji chromosomów,
2. ocena przystosowania chromosomów w populacji,
3. sprawdzenie warunku zatrzymania,
4. selekcja chromosomów,
5. zastosowanie operatorów genetycznych,
6. utworzenie nowej populacji,
7. wyprowadzenie najlepszego chromosomu.

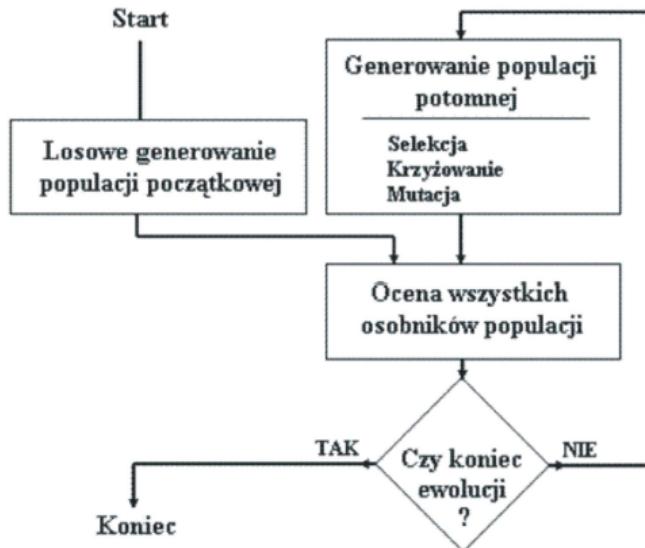
Najłatwiej wyobrazić sobie powyższe kroki analizując je na schemacie:

2.5.3 Operacje klasyczne, krzyżowanie.

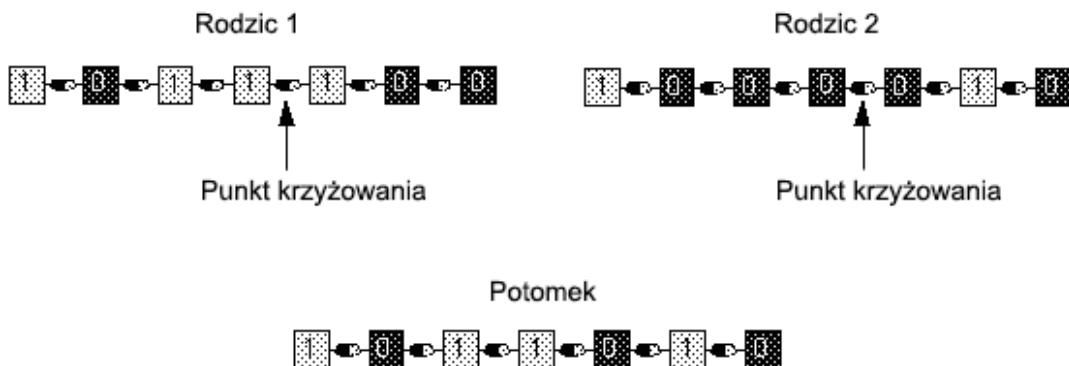
Operacja krzyżowania jest podstawową operacją algorytmów genetycznych służącą do rekombinacji materiału genetycznego. Operacja opiera się na dwóch chromosomach, których części materiału genetycznego zostają wymieszane w celu otworzenia nowego chromosomu. Podstawowa operacja krzyżowania opiera się o jeden punkt krzyżowania i została przedstawiona na poglądowym schemacie 2.10 [11].

2.5.4 Operacje klasyczne, mutacja.

Proces rekombinacji przez krzyżowanie nie byłby w stanie odkryć całej przestrzeni poszukiwań, jeżeli dana kombinacja nie byłaby obecna w sekcjach populacji. To mogłoby prowadzić do błędnego wyniku. Operacja mutacji pozwala na wprowadzenie nowych struktur genetycznych w obecnej populacji. Dokonuje tego poprzez losową zmianę dowolnego genu w chromosomie [11].



Rysunek 2.9: Ogólny schemat algorytmu genetycznego.



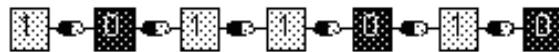
Rysunek 2.10: Ogólny schemat operacji krzyżowania.

2.6 Użycie grafów w algorytmie genetycznym.

W swojej pracy posługuję się przede wszystkim grafami. W tym wypadku klasyczna odmiana algorytmu genetycznego musiałaby zostać zmodyfikowana na potrzeby wykorzystania grafów jako chromosomów. Dodatkowo wymagałoby to zastosowania nowych sposobów krzyżowania i mutacji jednostek.

Zdecydowaliśmy, że zamiast przystosowywać algorytm genetyczny do nowej struktury, przystosujemy strukturę do algorytmu. Ponieważ naszym zadaniem jest zdecydowanie o zamknięciu lub nie, danej ulicy (krawędzi grafu) postanowiłem opisać tę strukturę jako listę opisującą ten stan. Zgodnie z tą myślą, sieć drogowa (graf) jest tłumaczona na tablicę elementów przyjmujących wartości:

Przed mutacją:



Po mutacji:



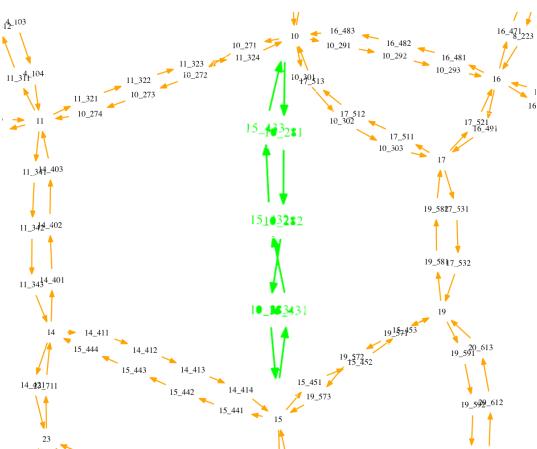
Rysunek 2.11: Ogólny schemat operacji mutacji.

- 1 (prawda) - dla ulicy (węzła), który jest przejezdny,
- 0 (fałsz) - dla ulicy (węzła) zamkniętego dla ruchu.

Tablica tworzona w ten sposób może być modyfikowana przez algorytm genetyczny, jak również bez problemu możemy z niej odtworzyć stan obecny sieci drogowej. Za przykład podajemy sytuację przedstawioną na rysunkach 2.12 i 2.13.

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

Rysunek 2.12: Fragment sieci w postaci tablicy binarnej

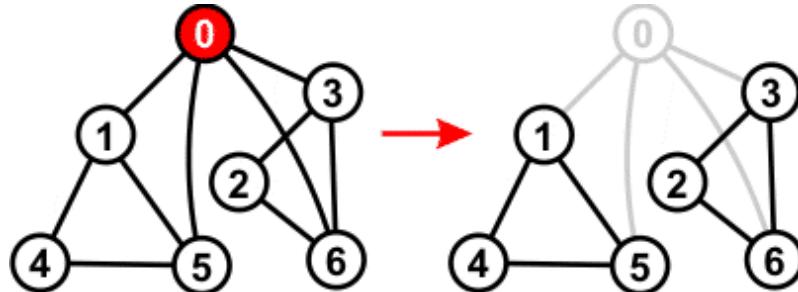


Rysunek 2.13: Fragment sieci w postaci grafu

Przedstawiony na rysunku 2.12 fragment sieci zawiera kolejne elementy zerowe, które w tłumaczeniu na przykładowy graf 2.13, są zaznaczone kolorem zielonym. W tym przypadku krawędzie te zostaną wyłączone z ruchu.

2.7 Punkty artykulacji grafu.

Punktem artykulacji¹ jest wierzchołek, którego usunięcie z grafu spowoduje zwiększenie liczby spójnych składowych. Na rysunku 2.14, punktem artykulacji jest wierzchołek o numerze 0.



Rysunek 2.14: Przykład grafu z zaznaczonym punktem artykulacji.

Algorytm poszukiwania punktów artykulacji wykorzystuje przeszukiwanie grafu w głąb. Podczas nierekurencyjnego przejścia, wierzchołek zostaje oznaczony jako punkt artykulacji, wtedy i tylko wtedy, jeśli jest on korzeniem posiada więcej niż dwóch synów. A wierzchołek v nie będący korzeniem drzewa DFS jest punktem artykulacji wtedy i tylko wtedy, gdy przynajmniej dla jednego jego syna nie istnieje krawędź wtórna $\{u, w\}$, taka że wierzchołek u jest potomkiem v i w jest przodkiem v .

Punkty te są wykorzystywane, by podczas losowych mutacji i krzyżowań w algorytmie genetycznym nie doszło do stworzenia grafu innego niż grafu skierowanego silnie spójnego. Krok sprawdzenia jest wykonywany podczas każdorazowego zamazywania drogi, zatem za każdym razem gdy zostaje wprowadzona nowa wartość „0” w tablicy.

¹ang. articulation point lub cut vertex

Rozdział 3

Technologie i metody użyte.

W rozdziale 3 znajdują się przedstawione technologie wykorzystane w projekcie, jak również krótki opis ich wykorzystania. Załączamy także opis systemu, na którym zostały wykonane obliczenia.

3.1 Symulator transportu.

MATSim jest środowiskiem¹ do implementacji szerokiej skali symulacji transportu opartych na agentach. Składa się z wielu modułów, które mogą zostać połączone lub używane oddzielnie. Moduły można zastępować własnymi implementacjami w celu przetestowania pojedynczych aspektów pracy [12].



Rysunek 3.1: Logo symulatora transportu MATSim

Oczywiście MATSim nie jest jedynym dostępnym symulatorem transportu. Podobnych rozwiązań jest wiele. Wybór motywowały jednak przede wszystkim dostępnością materiałów szkoleniowych i przykładowymi scenariuszami. MATSim jest żywym projektem oparty o licencję open source². Dysponuje szerokim zasobem przykładow i gotowych scenariuszy, w dodatku udostępniając prace wielu osób na swoim repozytorium.

¹ang. framework, pl. szkielet do budowy aplikacji

²pol. otwarte oprogramowanie

3.2 Algorytmy genetyczne.

Projekt The Apache Commons jest tworzony przez Apache Software Foundation. Głównym celem projektu jest stworzenie wolnego oprogramowania do wielokrotnego użytku w języku Java. Projekt podzielony jest na trzy główne części: proper, sandbox, i dormant [13].

W przypadku głównej części Apache Commons Proper wyróżniamy wiele modułów różniących się funkcjonalnością i celem. Apache Commons Math jest modułem zapewniającym rozwiązywania problemów głównie matematycznych i statystycznych. Znajduje się w nim implementacja podstawowej formy algorytmu genetycznego, którą rozszerzamy w pracy.



Rysunek 3.2: Logo biblioteki Apache Commons Math

3.3 Obsługa grafów.

Ponieważ moduł do wizualizacji rozwiązań MATSim nie spełniał naszych oczekiwania zarówno pod względem wydajności jak i stabilności zdecydowaliśmy się na stworzenie własnej implementacji. W tym celu wykorzystaliśmy dojrzały projekt NetworkX. Jest to biblioteka wykonana w języku Python stworzona z myślą o grafach i sieciach. Jest ona dostarczana jako darmowe oprogramowanie na licencji BSD-new [14].



Rysunek 3.3: Logo biblioteki NetworkX

3.4 Technologie i metodologie programistyczne.

Użyte przez nas technologie są poniekąd wymuszone przez języki w jakich zostały stworzone wykorzystywane pomocnicze rozwiązania. W przypadku symulatora MATSim jest to Java. Zdecydowaliśmy się natomiast na wykorzystanie Pythona głównie ze względu na przeważającą przewagę biblioteki NetworkX nad bibliotekami obsługującymi

grafy w języku Java. Nie jest to wyjątek, gdyż ogólnie rozwiązania dostarczane dla Python'a, szczególnie w przypadku problemów akademickich, są dużo lepsze. Poniżej zestaw narzędzi, które wykorzystaliśmy w procesie tworzenia aplikacji.



Rysunek 3.4: Logo Java.



Rysunek 3.5: Logo IDE Eclipse.



Rysunek 3.6: Logo Python.



Rysunek 3.7: Logo PyDev.

Logo pochodzą ze stron dostawców [15, 16, 17, 18].

3.5 Wdrożenie.

Ze względu na duże wymagania sprzętowe obliczeń, a zarazem ograniczoną przenośność rozwiązania z powodu skorzystania z Pythona, zdecydowaliśmy się na usprawnienie rozwiązania. Wykorzystując system Linux Ubuntu stworzyliśmy maszynę wirtualną spełniającą wszystkie wymagania do uruchomienia aplikacji. Dzięki temu stało się możliwe wykorzystanie innych komputerów oprócz tego, na którym zostało stworzone rozwiązanie. Pomijamy również dzięki temu problemy przystosowania czy instalowania dużej ilości zewnętrznego oprogramowania i bibliotek [19].



Rysunek 3.8: Logo systemu Linux Ubuntu.

W katalogach pracy znajduje się instrukcja wraz z wymaganymi pakietami do uruchomienia projektu. Podczas badań wykorzystywany był system Linux Ubuntu 12.04 LTS.

3.6 System obliczeniowy.

Korzystając z powyższego rozwiązania, dzięki któremu projekt jest możliwy do zainstalowania na innych maszynach, do obliczeń wykorzystaliśmy zewnętrznego dostawcę. Przy wyborze decydującym czynnikiem była cena rozwiązania, co w przypadku chmury Microsoft Windows [20], pozwoliło na darmowe rozwiązanie³.



Rysunek 3.9: Logo chmury Microsoft Windows Azure.

³dzięki darmowemu okresowi próbному

Rozdział 4

Opis projektu.

W niniejszym rozdziale opisujemy strukturę projektu z opisem jego uruchomienia. Ponieważ przeprowadzane przez projekt obliczenia wymagają wiele czasu, zdecydowaliśmy się na pominięcie interfejsu użytkownika przy projektowaniu aplikacji. Całość jest obsługiwana przez plik konfiguracyjny, który zostaje wczytany na początku operacji i za jego pomocą kontrolujemy przebieg obliczeń. Pomimo dwóch osobnych technologii, w których został wykonany projekt, są one od siebie zależne. Całość jest obsługiwana przez projekt w Java, skrypty w Pythonie jedynie wspierają niektóre procesy.

4.1 Dane wejściowe.

Przykładowy plik konfiguracyjny dla projektu z opisem jego funkcji przedstawiono na listingu 4.1.

Listing 4.1: Plik konfiguracyjny projektu

```
<?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>
        <python-main>../python/ms/call_center.py</python-main>
        <java-path>/usr/bin/java</java-path>
        <matsim-jar>../matsim/matsim-r31927mod.jar</matsim-jar>
        <matsim-xmx>2g</matsim-xmx>
    </project>
    <scenario>
        <config>../scenarios/siouxfalls/config.xml</config>
        <network>../scenarios/siouxfalls/network.xml</network>
        <population>../scenarios/siouxfalls/population.xml</population>
        <facilities>../scenarios/siouxfalls/facilities.xml</facilities>
        <iterations>9</iterations>
```

4.1. DANE WEJŚCIOWE.

```
</scenario>
<genetics>
    <population-size>24</population-size>
    <max-generations>200</max-generations>
    <elitism-rate>2.0</elitism-rate>
    <crossover-rate>1.0</crossover-rate>
    <mutation-rate>0.8</mutation-rate>
    <tournament-arity>4</tournament-arity>
</genetics>
</config>
```

Jak widać konfiguracja dzieli się na trzy podstawowe grupy:

- project,
- scenario,
- genetics.

Grupa *project* odpowiada za główne ustawienia całego projektu, w *scenario* znajdziemy ustawienia dotyczące symulacji przeprowadzanej przez MATSim, natomiast sekcja *genetics* zawiera ustawienia dotyczące algorytmu genetycznego.

Wyjaśnienie opcji *project*:

- name - nazwa projektu, używana jako katalog wyjściowy,
- output dir - katalog, gdzie zapisujemy wyniki,
- threads - ilość wątków, które mają być użyte podczas obliczeń,
- log level - poziom logowania Log4J¹,
- python path - ścieżka instalacji Pythona,
- python main - folder ze skryptami pomocniczymi,
- java-path - ścieżka instalacji Javy,
- matsim-jar - ścieżka do biblioteki MATSim,
- matsim-xmx - maksymalna pamięć RAM dostępna dla symulatora MATSim.

Wyjaśnienie opcji *scenario*:

¹zewnętrzna biblioteka do logowania, dostępne poziomy: DEBUG, INFO, WARN, ERROR i FATAL

- config - scieżka dostępu pliku konfiguracyjnego scenariusz MATSim,
- network - scieżka dostępu pliku z siecią wejściową scenariusza,
- population - scieżka dostępu pliku z populacją scenariusza,
- facilities - scieżka dostępu pliku z budynkami scenariusza,
- iterations - ilość iteracji symulacji MATSima.

Wyjaśnienie opcji *genetics*:

- population size - rozmiar populacji,
- max generations - ilość testowanych generacji (warunek stopu),
- elitism rate - ilość najlepszych chromosomów biorących udział w kolejnej iteracji,
- crossover rate - szansa na krzyżowanie osobników,
- mutation rate - szansa na mutacje osobników,
- tournament rate - ilość osobników biorących udział w turnieju.

4.2 Badane miasto przykładowe: Sioux Falls

Główną zaletą korzystania z MATSima, o której już wcześniej wspominaliśmy, jest dość pokaźny zbiór danych przykładowych. Jednym z nich jest materiał zaprezentowany przez twórców aplikacji, który udostępnili w 2013 roku na zgromadzeniu użytkowników platformy [8]. Przykład ten dotyczy miasta Sioux Falls w Południowej Dakocie. Domyslnie scenariusz składa się z:

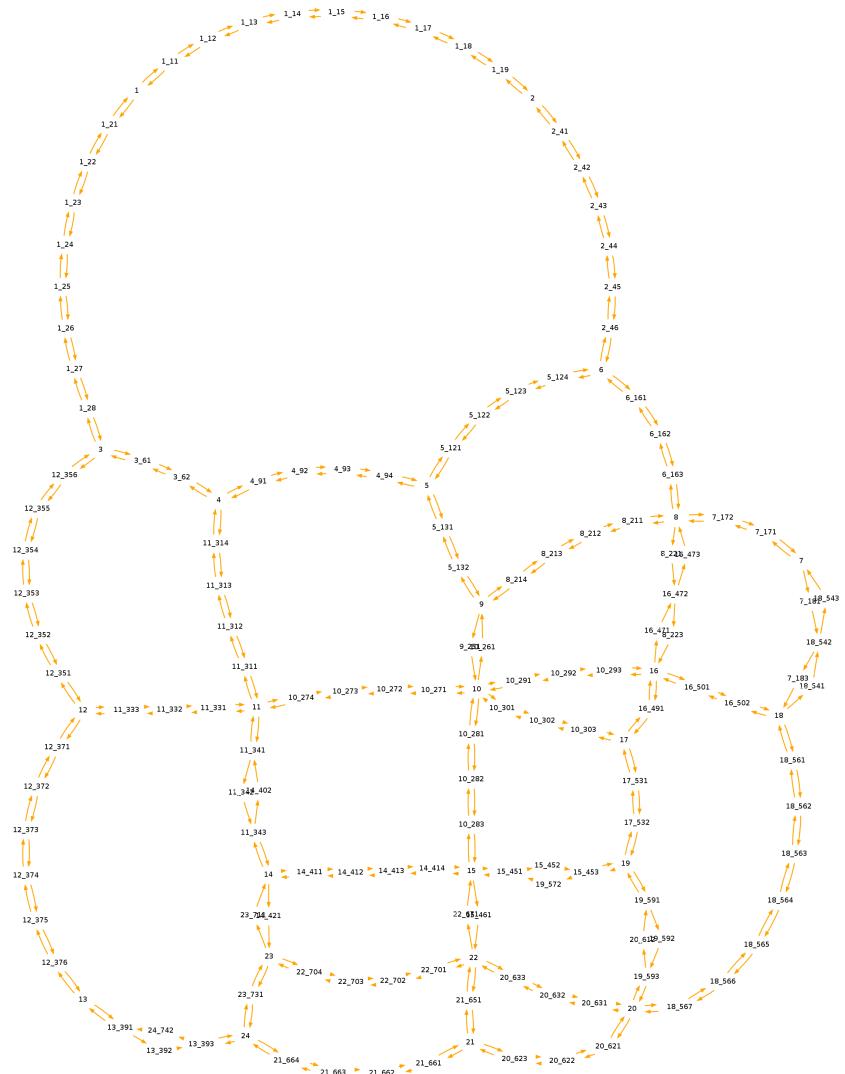
- dwóch grup zapotrzebowania bez charakterystyk socjodemograficznych:
 - 68094 agentów z samochodem oraz korzystających z transportu publicznego,
 - 40877 agentów posiadających samochód.
- dostosowanej sieci drogowej miasta Sioux Falls,
- transportu publicznego razem z rozkładem jazdy,
- przykładowych miejsc zamieszkania, pracy i rozrywki.

4.2. BADANE MIASTO PRZYKŁADOWE: SIOUX FALLS

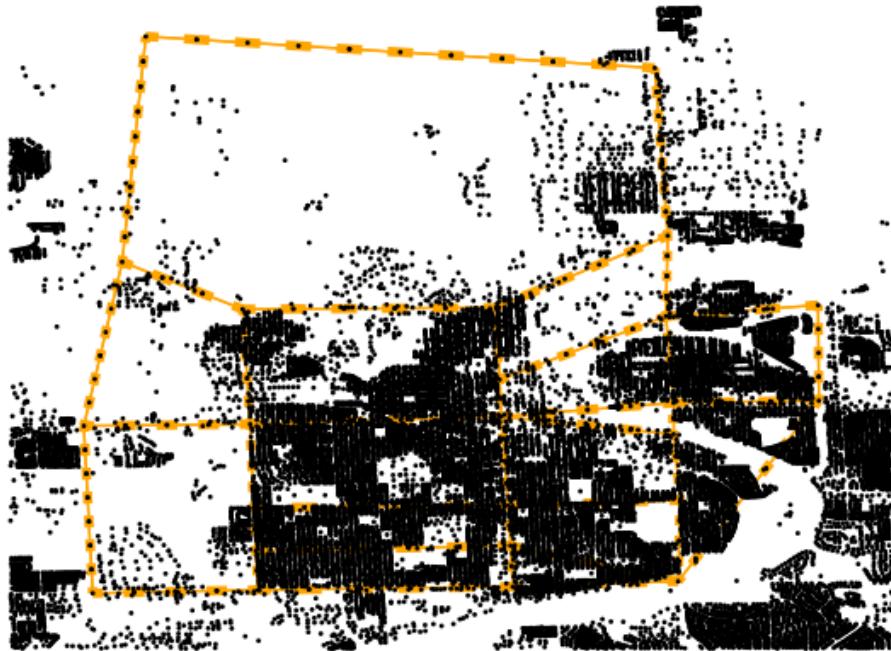
Po zapoznaniu się z przykładem, dostarcza on nawet za dużo danych, które by nas interesowały. Dostosowaliśmy go zatem do swoich potrzeb poprzez:

- usunięcie transportu publicznego,
 - wyposażenie każdego agenta w swój samochód.

Powyższe modyfikacje znacznie wzmogły ruch w mieście (co było dla nas pozytywnym efektem) i skróciły obliczenia związane z symulacjami. Na rysunku 4.1 przedstawiamy sieć miasta odwzorowaną w postaci grafu. Natomiast na rysunku 4.2 prezentujemy rozkład miejsc pracy, domostw i innych zakładów nałożonych na graf sieci drogowej miasta.

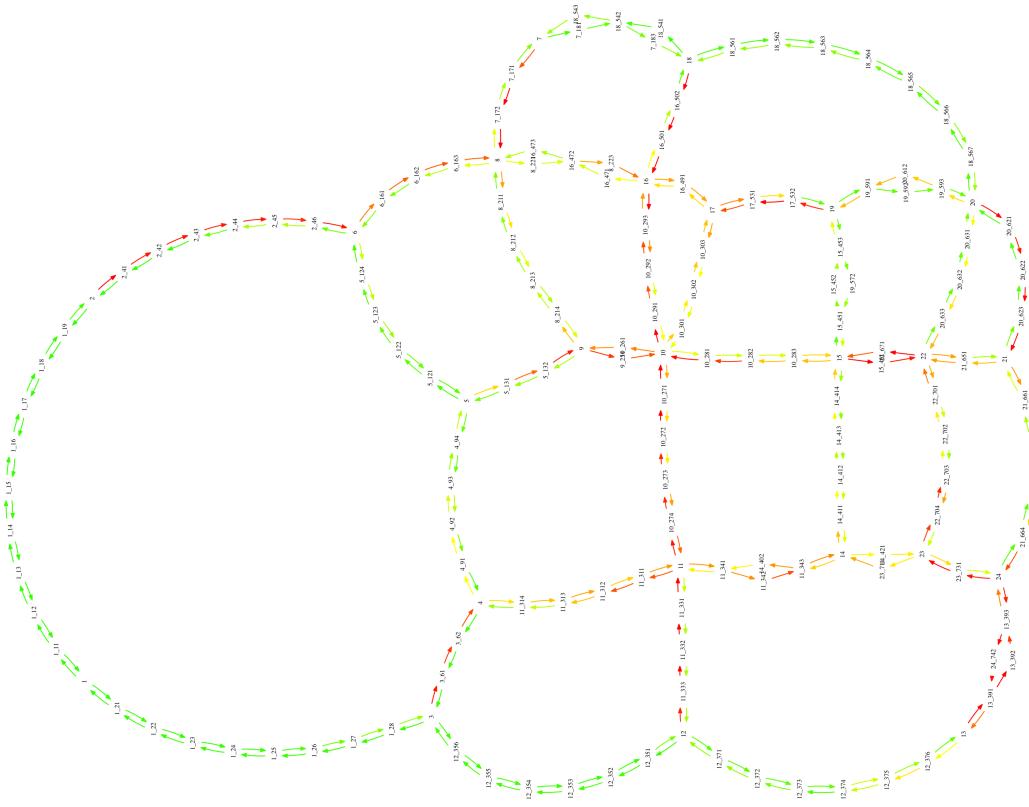


Rysunek 4.1: Graf sieci miasta Sioux Falls wykorzystany w badaniach.

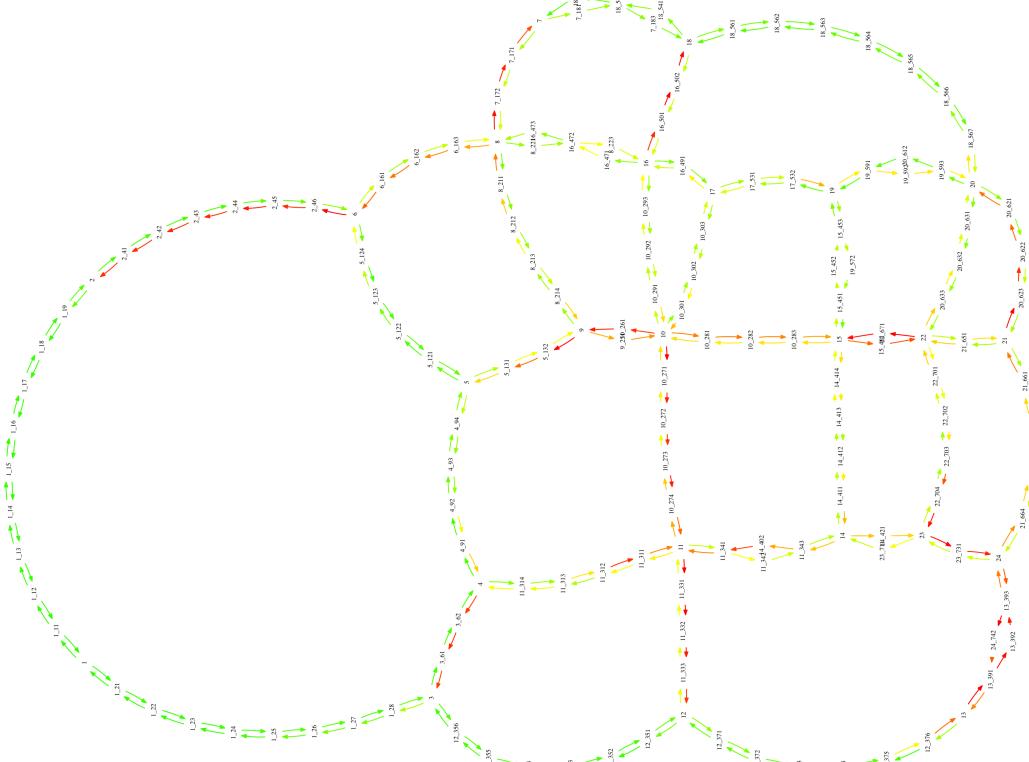


Rysunek 4.2: Rozkład budynków na grafie miasta Sioux Falls.

Na rysunkach 4.3 i 4.4 prezentujemy ruch drogowy w najbardziej intensywnych godzinach dnia, tj. godzinach szczytu. Ruch jest przedstawiony jako natężenie ruchu na każdej krawędzi (ulicy), poprzez odniesienie go do możliwości przepustowości każdego węzła. Kolor zielony oznacza małe obciążenie, żółty średnie i czerwony - wysokie natężenie ruchu w tym miejscu. Rysunki są obrócone o 90° w lewo, dla zwiększenia ich czytelności i rozmiaru.



Rysunek 4.3: Natężenie ruchu w mieście Sioux Falls w godzinach 6.00-7.00.

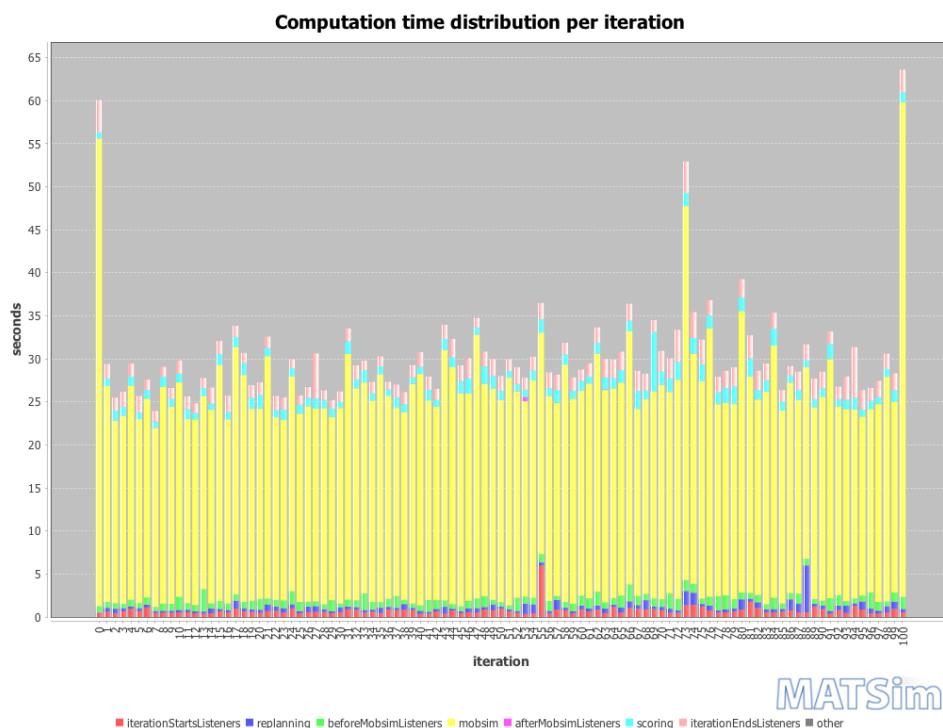


Rysunek 4.4: Natężenie ruchu w mieście Sioux Falls w godzinach 16.00-17.00.

4.3 Ustawienia symulatora.

Ostatnią ważną kwestią przed realizacją samego zadania optymalizacji jest dobór jego ustawień. Część z nich jest oczywiście stała lub nie ma wpływu bezpośrednio na obliczenia. Kluczową rolę dla ostatecznego wyniku odgrywają jednak parametry samej symulacji.

Większość ustawień została niezmieniona, pochodzi więc od twórców przykładu Sioux Falls. Symulator zawiera jednak ważny parametr ilości iteracji. Zgodnie z założeniami symulatora, podczas każdej kolejnej iteracji, celem agenta jest zwiększenie jego średniego wyniku. Dąży on zatem do optymalizacji swoich planów dnia względem warunków na drodze. Przekłada się to bezpośrednio również na średni czas podróży każdego agenta. Twórcy zalecają stosowanie wysokich liczb iteracji, dla uzyskania jak najlepszych wyników. Operacje te są jednak czasochłonne. Na rysunku 4.5 przedstawiony został graf czasu uruchomienia symulacji jednej sieci dla stu iteracji. Podczas symulacji wykorzystywana była domyślna sieć miasta.



Rysunek 4.5: Graf czasu trwania symulacji MATSim dla stu iteracji.

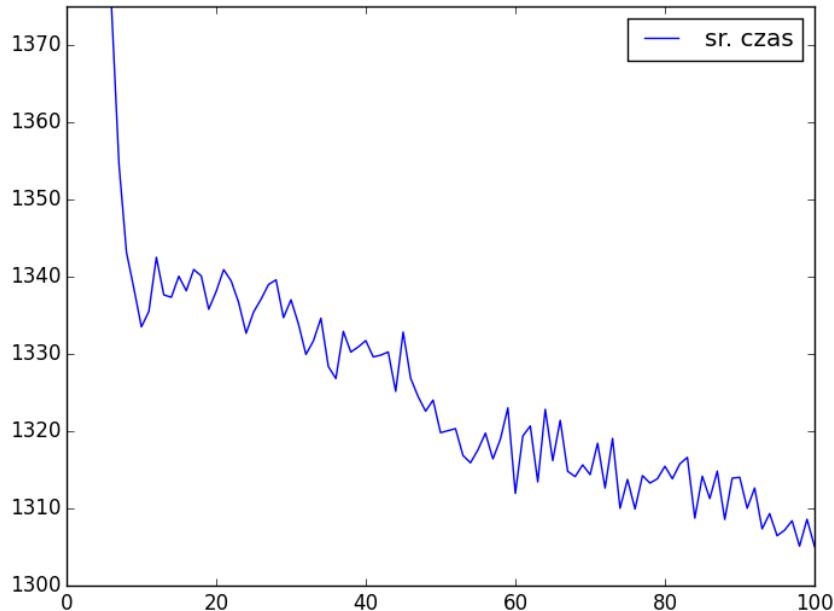
Warto również omówić kolejne kroki każdej iteracji, w nawiasie podaję nazwy procesów z wykresu:

1. initial demand (iterationStartListeners),
2. execution (beforeMobSimListeners, mobSim, afterMobSimListeners),

3. scoring (scoring),
4. replanning (replanning),
5. analyses (iterationEndsListeners).

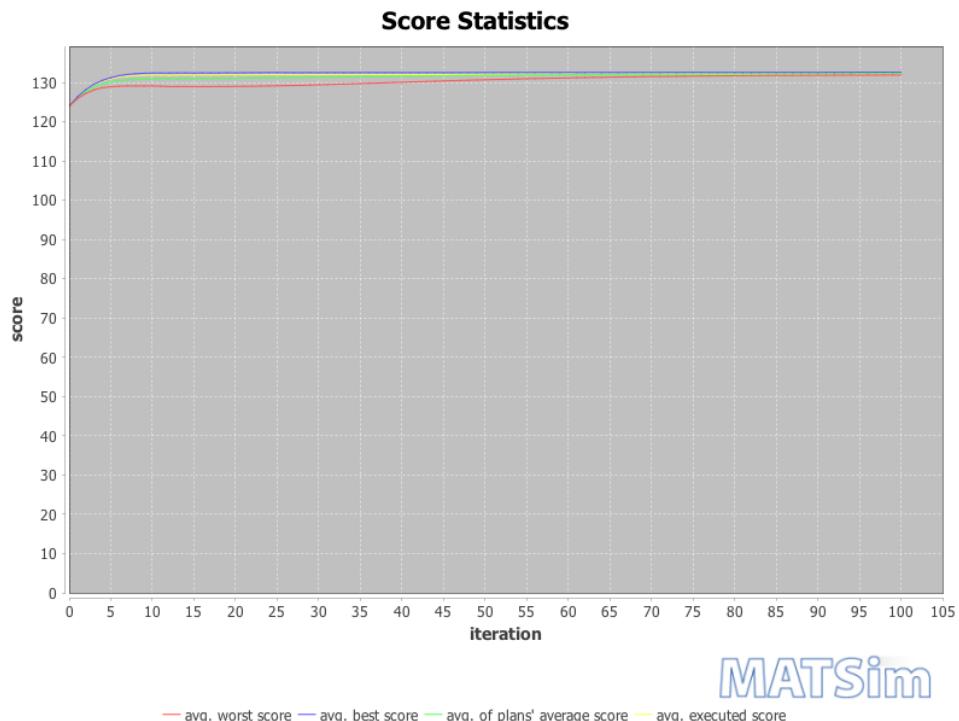
W pierwszym punkcie iteracji, agenci przygotowują się do swoich dni. Znając swoje zajęcia, planują oni drogę oraz dokładne godziny swoich akcji. Podczas *execution* następuje sprawdzenie tych planów w akcji, czyli zachodzi właściwa symulacja. Podczas *scoring* agenci zostają ocenieni na podstawie swojego dnia. Generalnie rzecz biorąc wyższy wynik uzyskują agenci, którzy jak najmniej czasu spędzili w podróży a najwięcej podczas zaplanowanych zajęć. Najważniejszym punktem symulacji jest jednak *replanning*. W tym punkcie agenci „wyciągają wnioski” ze swojego dnia i starają się nie popełnić tych samych błędów, wciąż dążąc do lepszego wyniku. Ostatni punkt iteracji dotyczy stworzenia analizy iteracji dostępnej dla użytkownika.

Analizując wykres czasu trwania kolejnych iteracji, na rysunku 4.5, możemy również odczytać czas całej symulacji, który wyniósł dokładnie 52min. 43sek. W przypadku analizy wielu (setek) sieci, czas więc jest wyraźnie zbyt długi. Przeanalizowaliśmy wpływ ilości iteracji na średni czas przejazdu, rysunek 4.6. Średni wynik² uzyskany przez agentów został natomiast przedstawiony na rysunku 4.7.



Rysunek 4.6: Graf średniego czasu przejazdów dla stu iteracji.

²ang. score



Rysunek 4.7: Graf wyników agentów dla stu iteracji.

Na wykresach 4.6 i 4.7, widać, że wpływ iteracji, ma największe znaczenie w początkowych stadiach. W późniejszych iteracjach zmiana następuje dużo wolniej. Wiąże się to z wykorzystanym algorytmem występującym w fazie *replanning*.

Po analizie wykresów prezentowanych w sekcji 4.3, zdecydowaliśmy w projekcie na użycie 10 iteracji podczas analizy każdej sieci. Wartość ta optymalnie łączy zalety szybkiego spadku wartości średnich czasów podróży i czasu potrzebnego na obliczenia.

4.4 Ustawienia projektu.

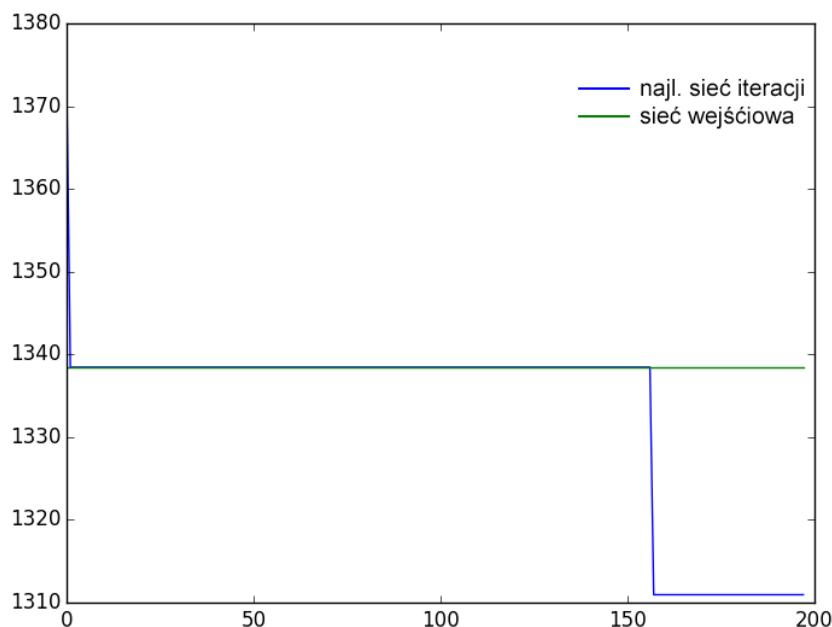
Podczas badań wykorzystaliśmy ustawienia algorytmu genetycznego zgodne z listingu 4.2. Wartości zostały dobrane na podstawie doświadczenia nabytego w trakcie studiów literaturowych.

Listing 4.2: Ustawienia algorytmu genetycznego podczas badań.

```
<genetics>
  <population-size>24</population-size>
  <max-generations>200</max-generations>
  <elitism-rate>2.0</elitism-rate>
  <crossover-rate>1.0</crossover-rate>
  <mutation-rate>0.8</mutation-rate>
  <tournament-arity>4</tournament-arity>
</genetics>
```

4.5 Wyniki.

Wykres przedstawiający zestawienie wyniku najlepszej sieci wyłonionej w danej iteracji, do wyniku sieci wejściowej przedstawiamy na rysunku 4.8.



Rysunek 4.8: Graf wyników algorytmu genetycznego.

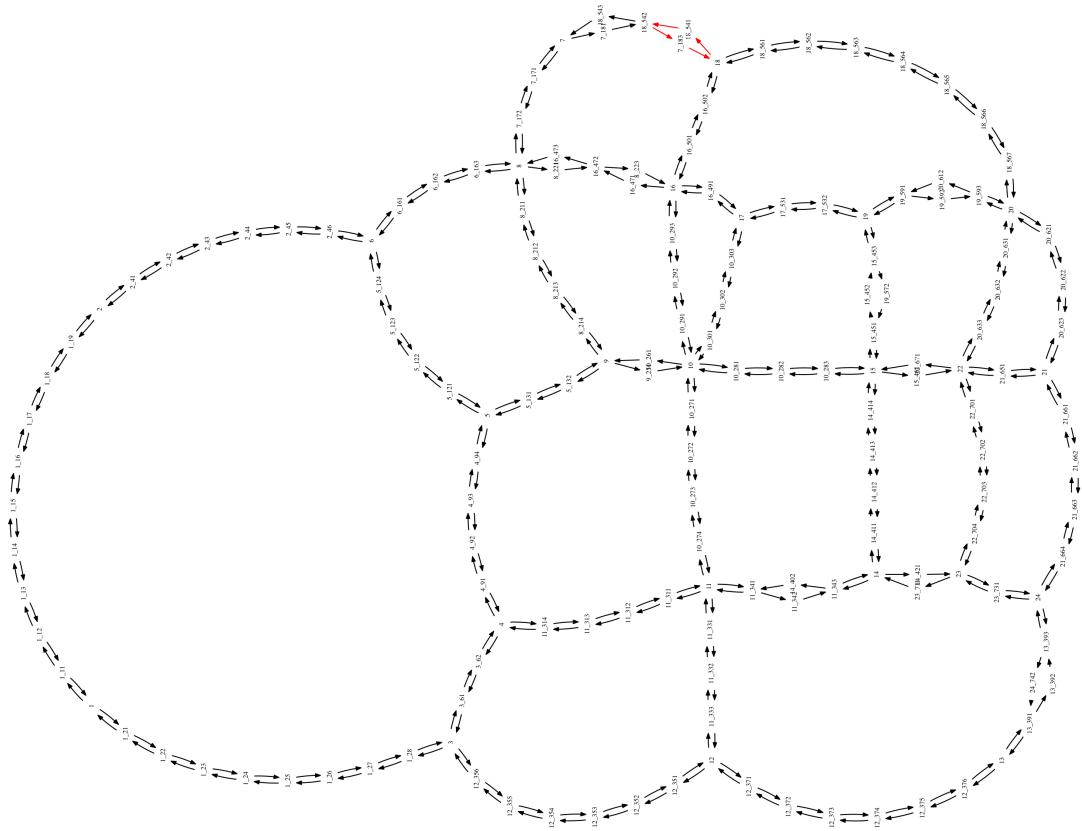
Ponieważ w projekcie wykorzystany został elitystyczny model populacji algorytmu genetycznego, powyższy wykres nie przedstawia praktycznie żadnych zmian w populacji, która została ulepszona. Wyniki wszystkich chromosomów były jednak zachowywane w bazie. Dzięki temu możemy przedstawić więcej sieci spełniających założenia projektowe.

Najlepsze wyniki symulacji przedstawiamy w tabeli 4.1. Przedstawione są tutaj wszystkie sieci, które uzyskały wynik lepszy od sieci wejściowej, tj. 1338.4505528474617. Od teraz, będziemy odnosić się do tych sieci używając ich numeru identyfikacyjnego (pierwsza kolumna).

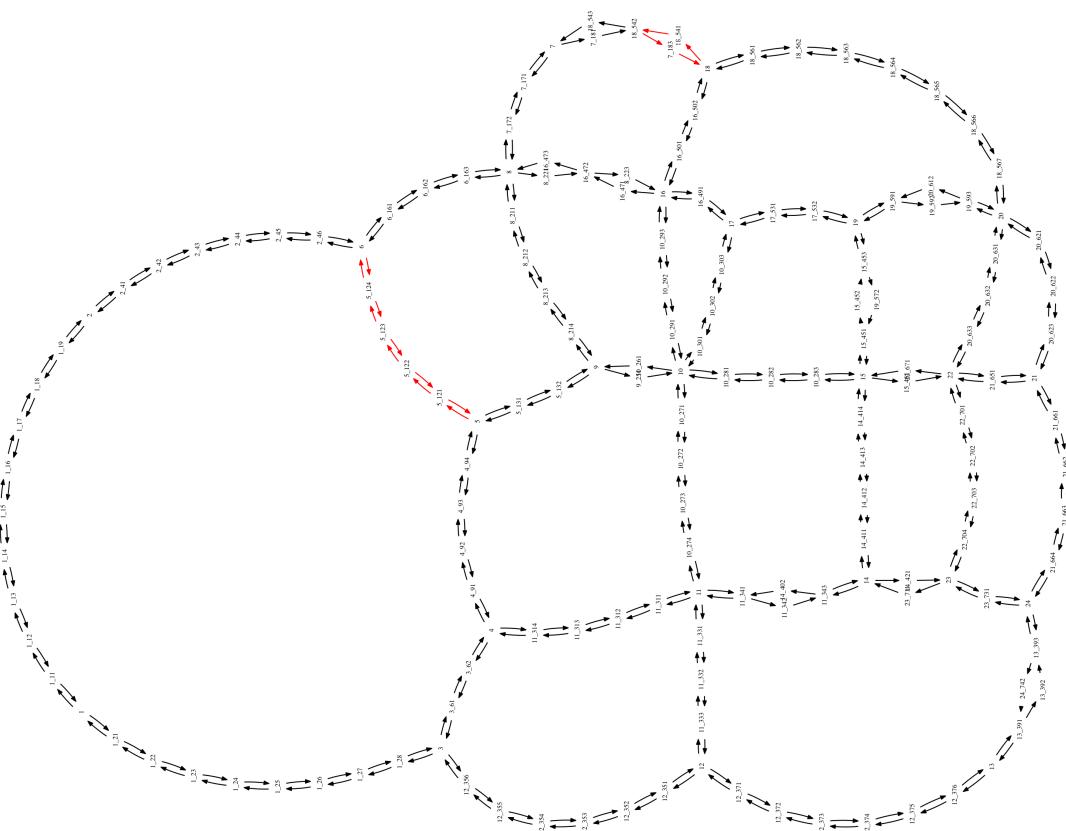
Tablica 4.1: Zbiór najlepszych otrzymanych sieci.

ID	Reprezentacja binarna	Wynik sieci
1	0111 11	1310.95255617644
2	0111 11	1312.64143383664
3	0111 11	1312.73299250981
4	0111 11	1313.75345975508
5	0111 11	1318.77194744977
6	0111 11	1320.54353227916
7	0111 11	1322.64708120319
8	0111 11	1323.53190464867
9	0111 11	1325.07095470218
10	0111 11	1327.94727737487
11	0111 11	1329.46298894305
12	0111 11	1332.87335631911
13	0111 11	1334.28138152419
14	0111 11	1334.48972773749
15	0111 11	1335.74516109856
16	0111 11	1335.96306027821
17	0111 11	1336.65051123529

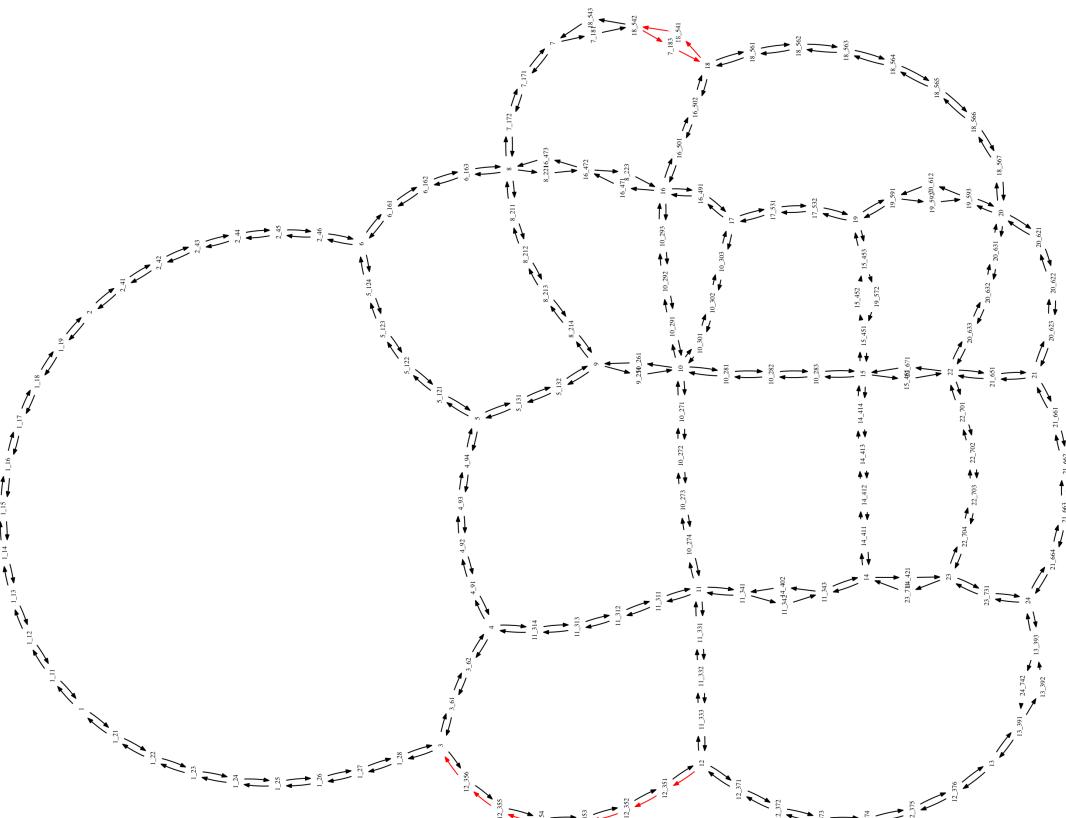
Na rysunkach 4.9 - 4.25 prezentujemy grafy uzyskanych sieci, dla wizualizacji rozwiązań. **W celu uproszczenia odnajdowania usuniętych węzłów, są one zaznaczone kolorem czerwonym.** Tak, jak w poprzednim przypadku, sieci zostały obrócone o 90° w lewo. W niektórych przypadkach wycięte węzły są trudno widoczne, z powodu nikiej długości strzałki. Dotyczy to przede wszystkim przykładów: 4.13, 4.15, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.24. W tych sieciach zostały wycięte drogi w pobliżu węzła o numerze 15.



Rysunek 4.9: Sieć miasta Sioux Falls, rozwiązanie nr. 1.

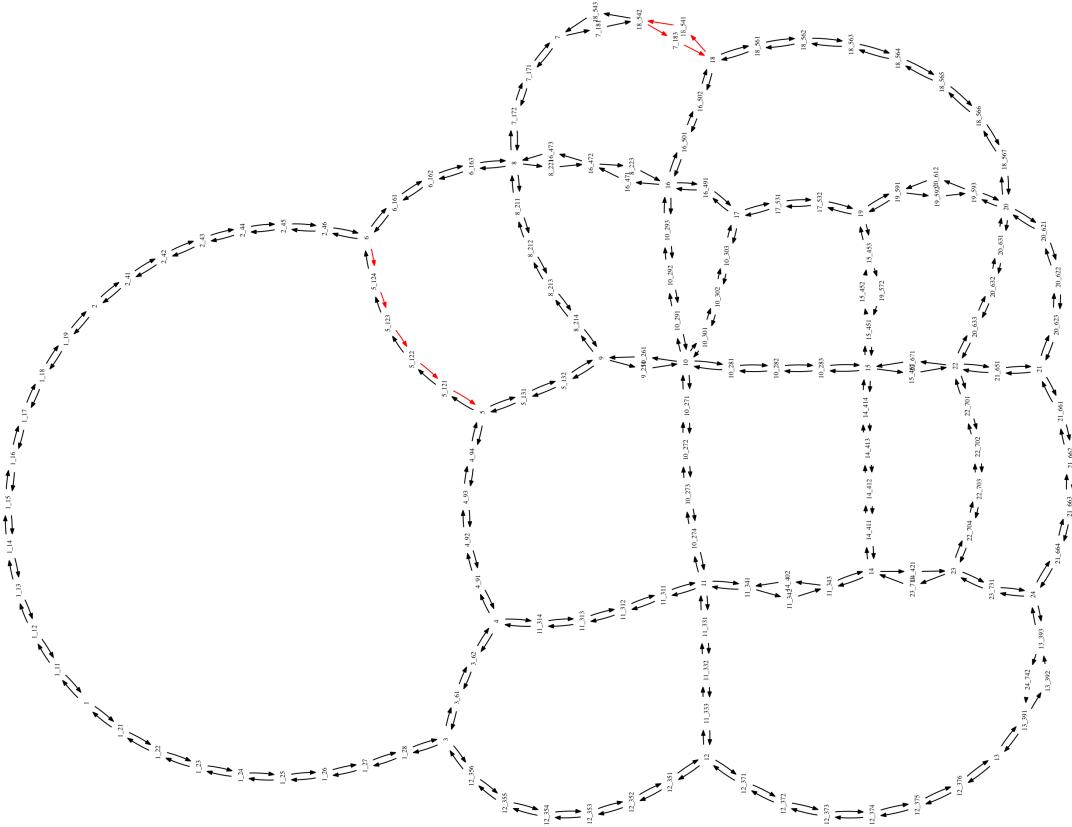


Rysunek 4.10: Sieć miasta Sioux Falls, rozwiązanie nr. 2.

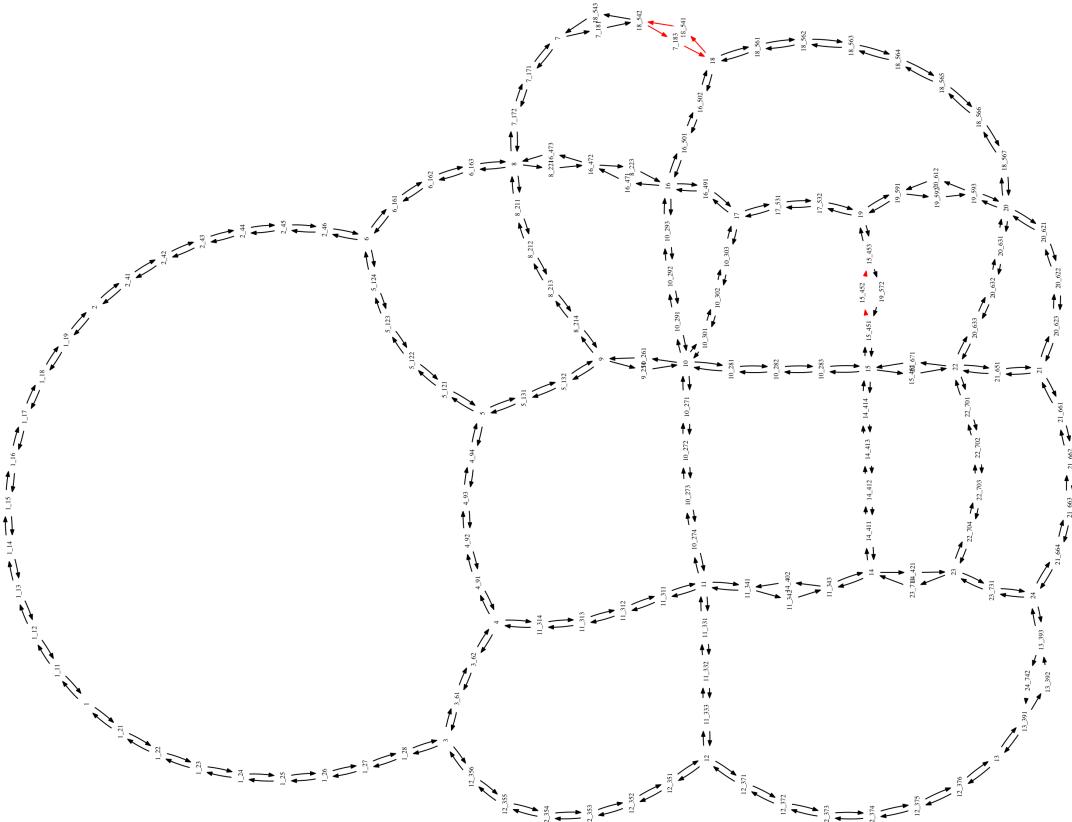


Rysunek 4.11: Sieć miasta Sioux Falls, rozwiązańe nr. 3.

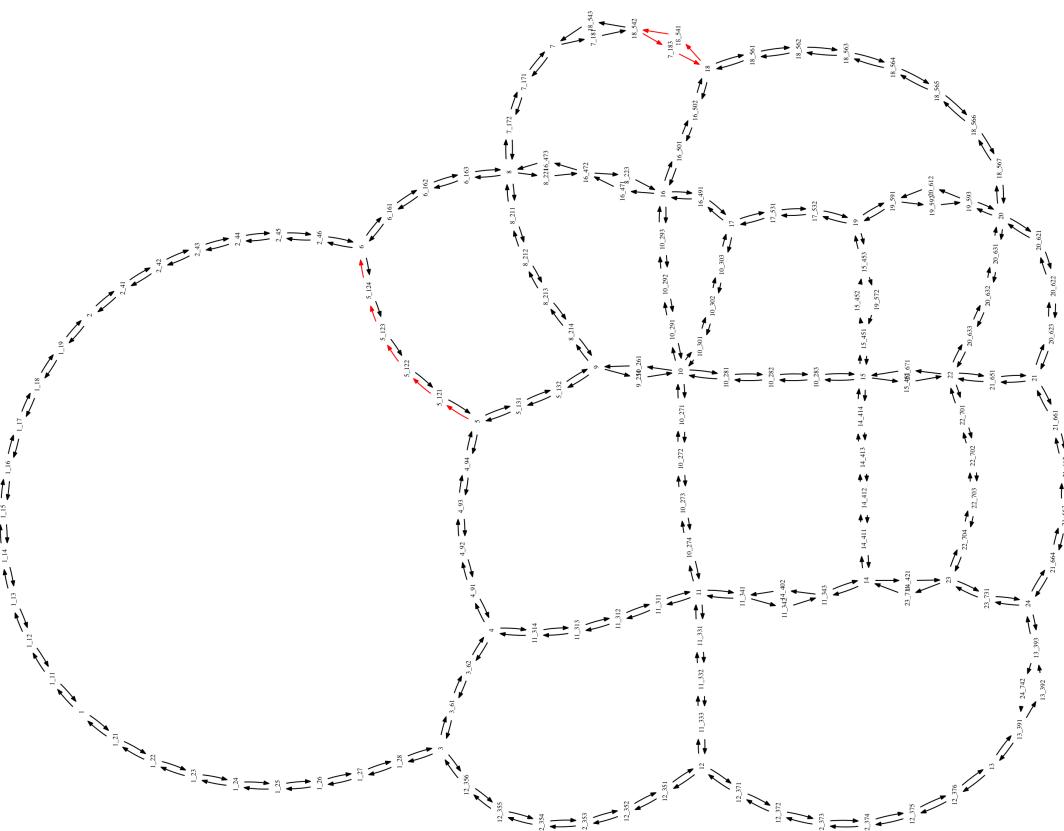
4.5. WYNIKI.



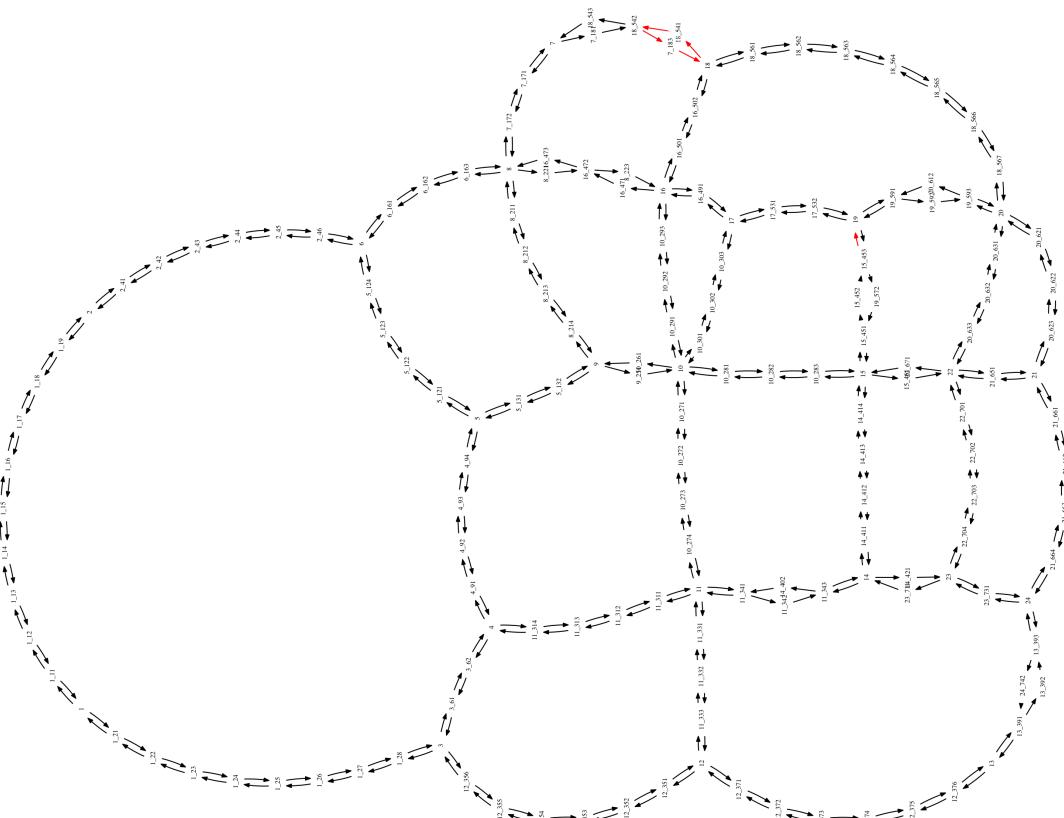
Rysunek 4.12: Sieć miasta Sioux Falls, rozwiàzanie nr. 4.



Rysunek 4.13: Sieć miasta Sioux Falls, rozwiàzanie nr. 5.

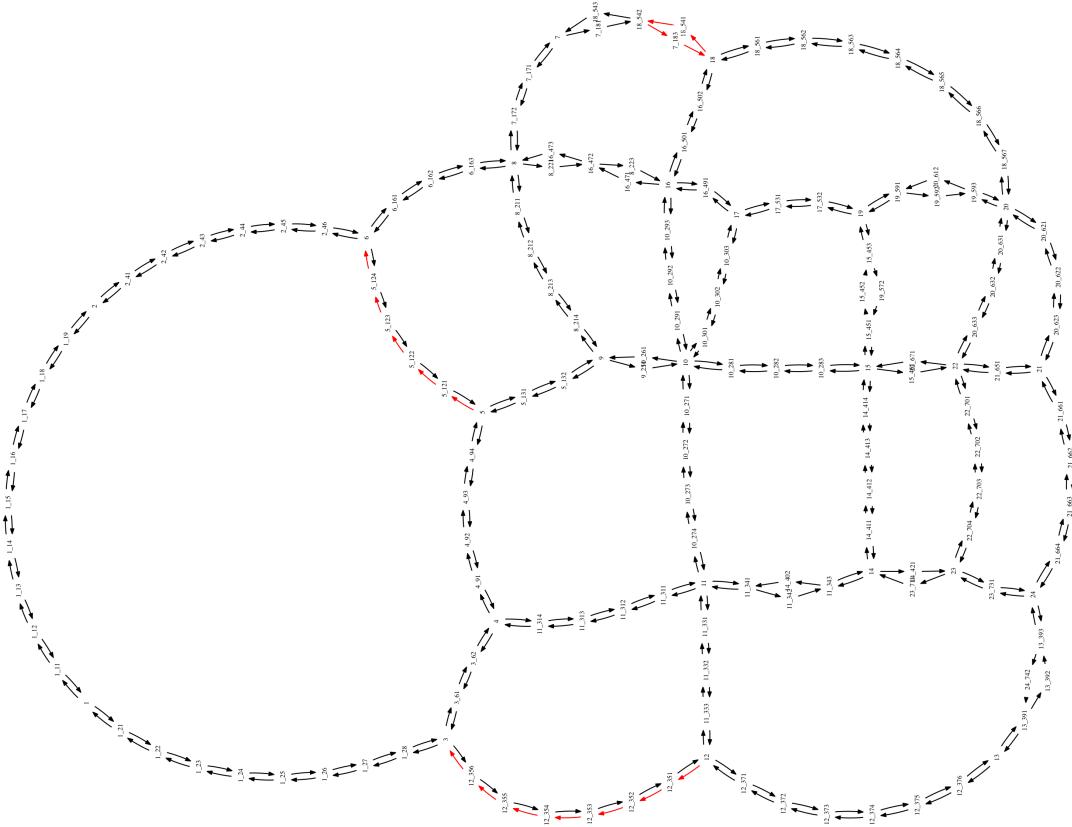


Rysunek 4.14: Sieć miasta Sioux Falls, rozwiàzanie nr. 6.

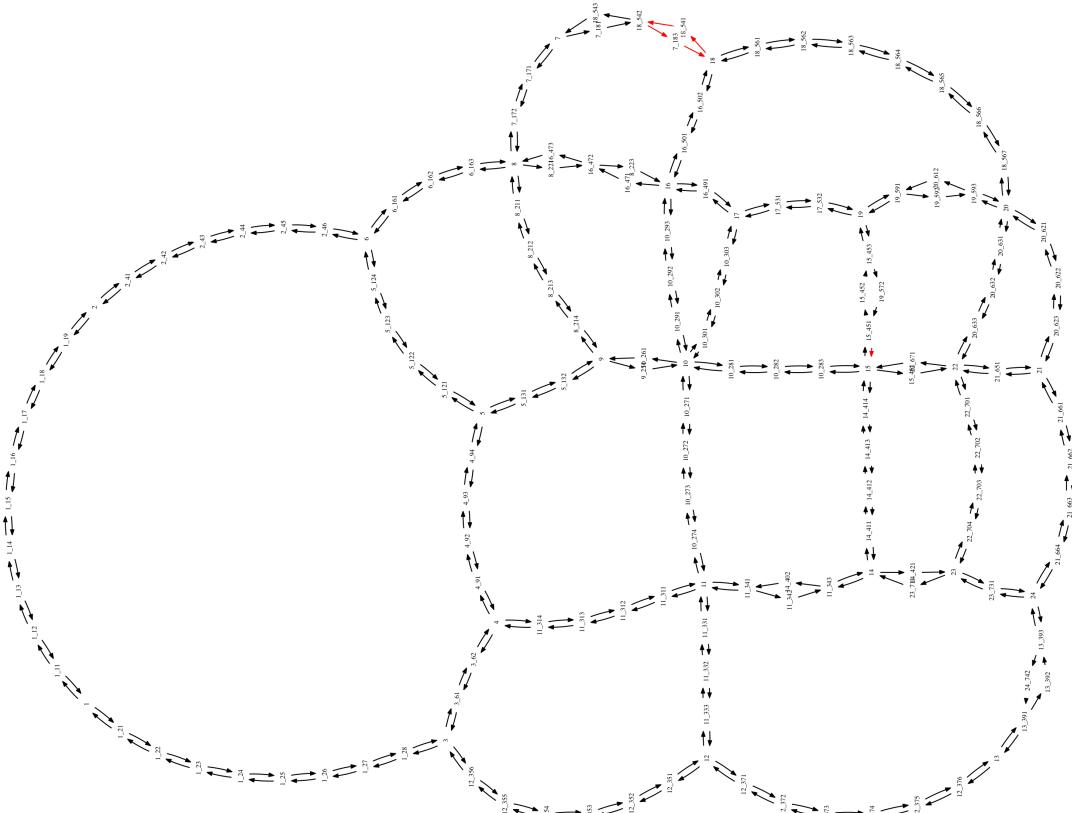


Rysunek 4.15: Sieć miasta Sioux Falls, rozwiązańe nr. 7.

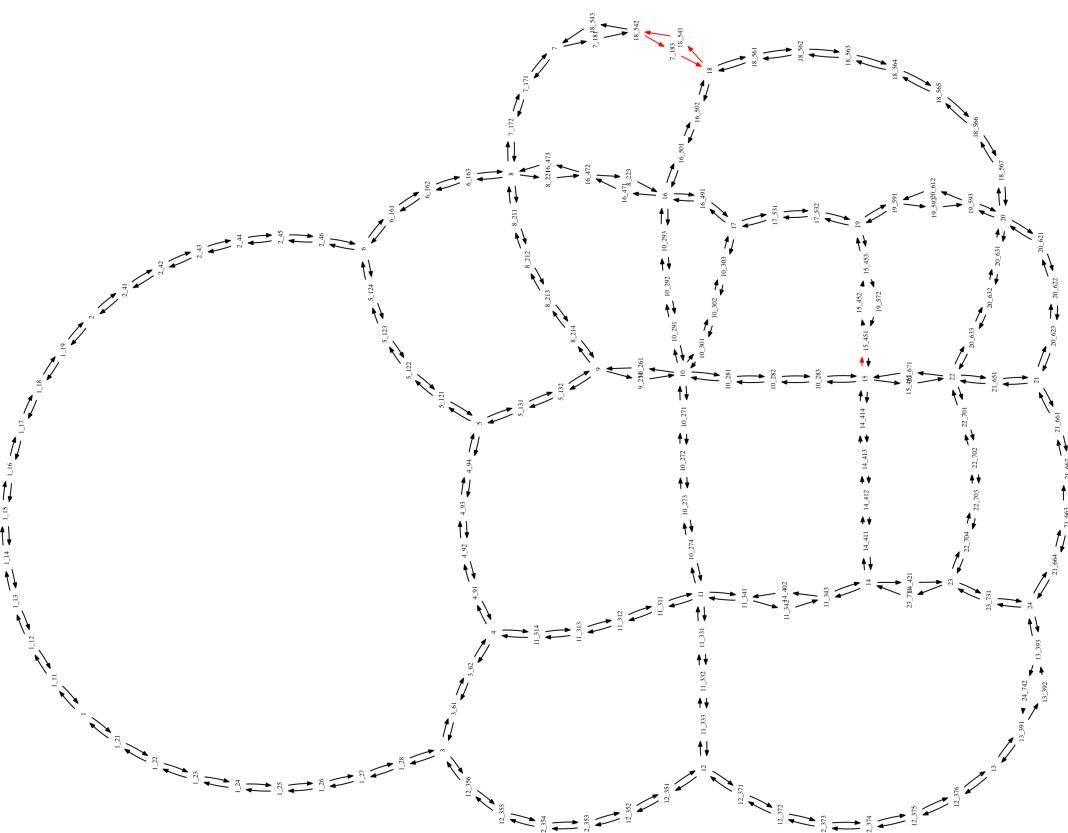
4.5. WYNIKI.



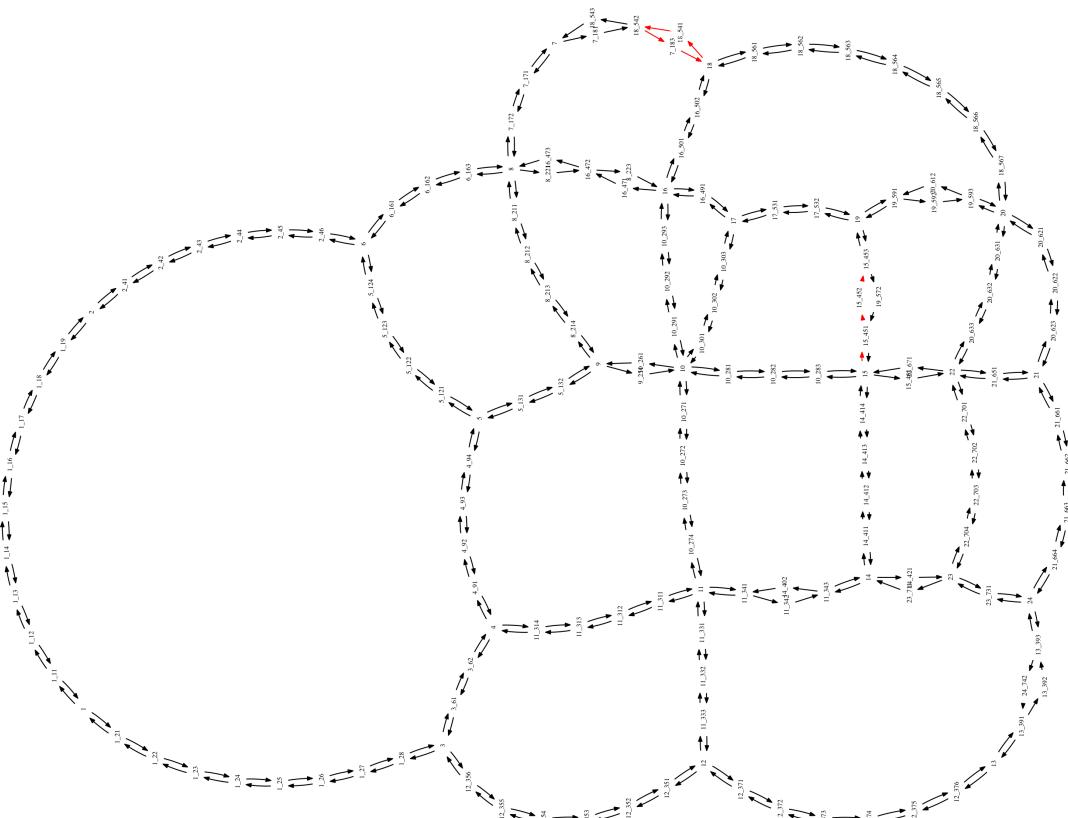
Rysunek 4.16: Sieć miasta Sioux Falls, rozwiàzanie nr. 8.



Rysunek 4.17: Sieć miasta Sioux Falls, rozwiązańe nr. 9.

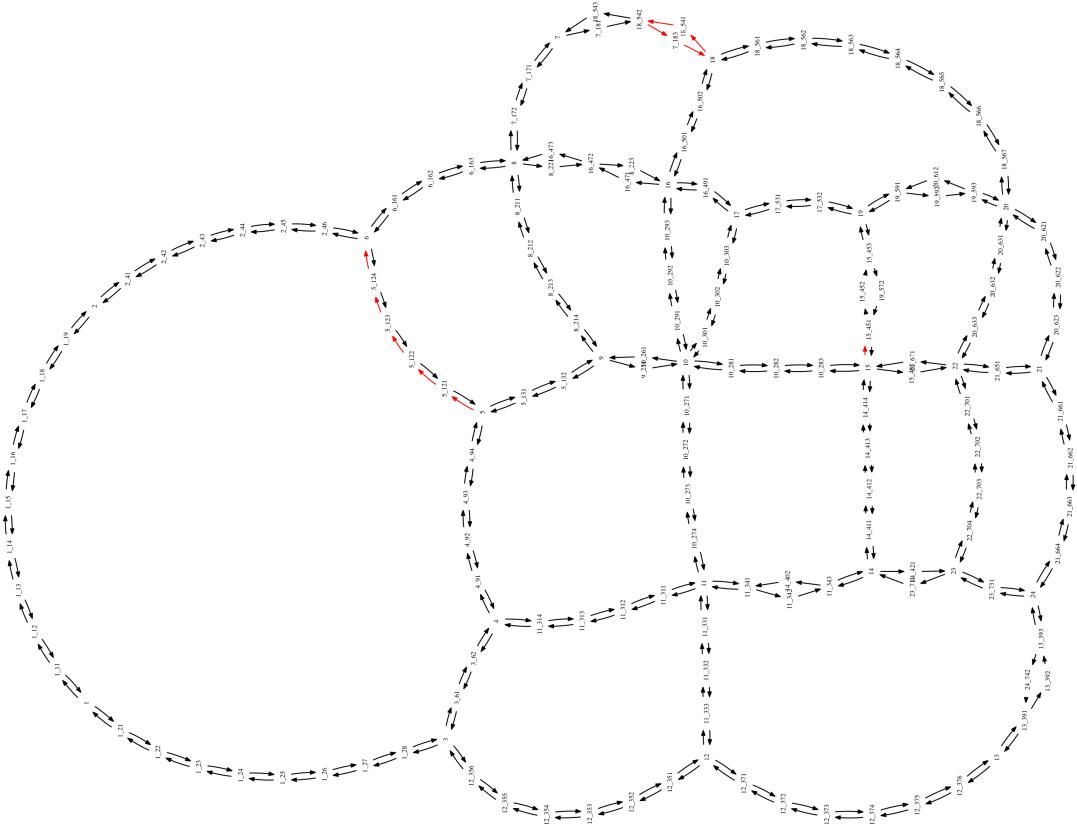


Rysunek 4.18: Sieć miasta Sioux Falls, rozwiązańe nr. 10.

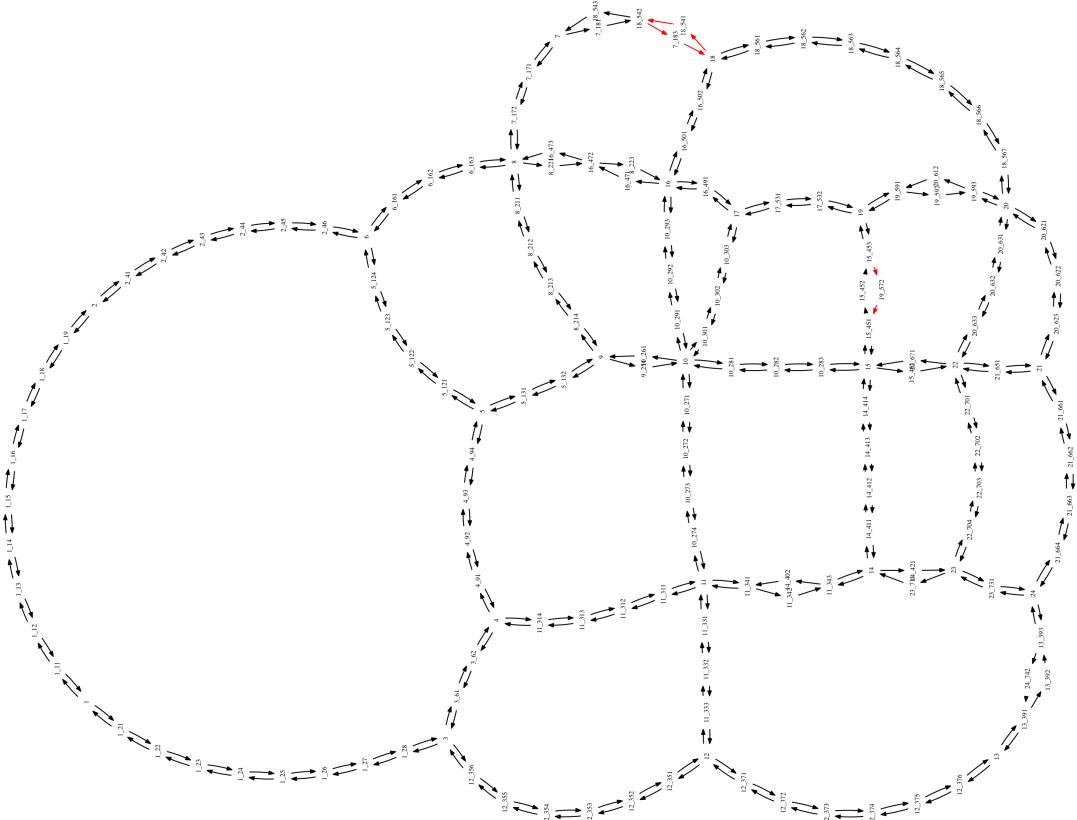


Rysunek 4.19: Sieć miasta Sioux Falls, rozwiążanie nr. 11.

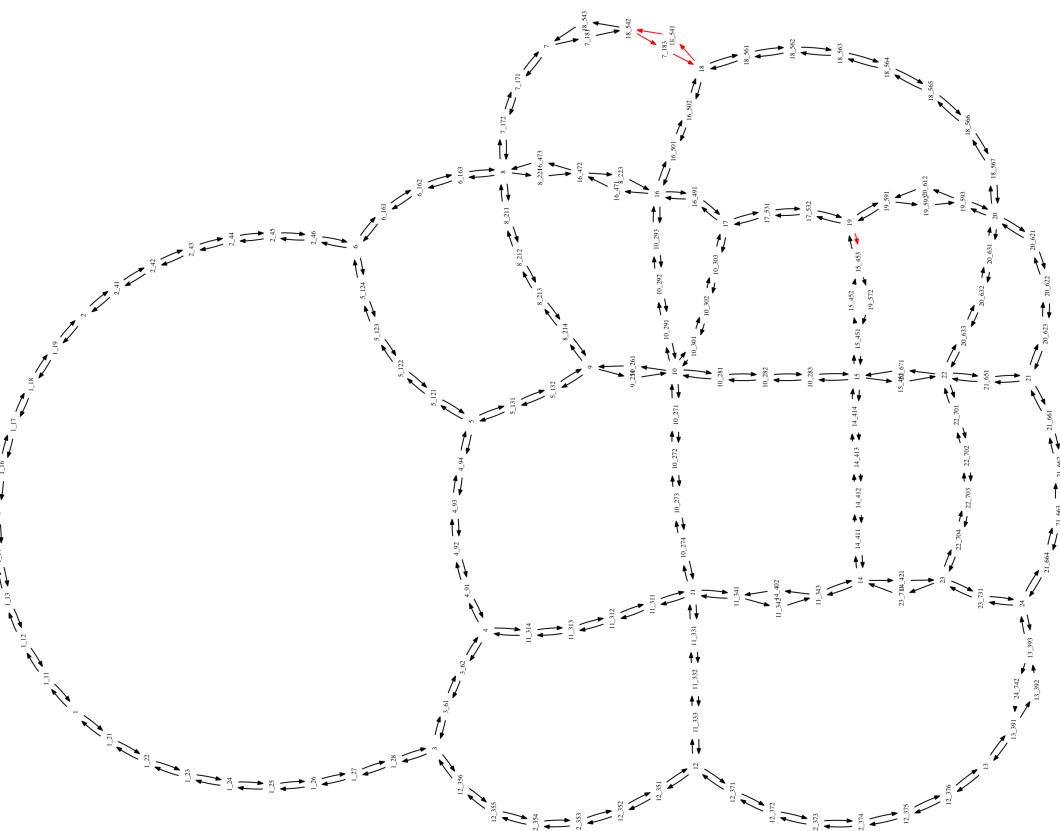
4.5. WYNIKI.



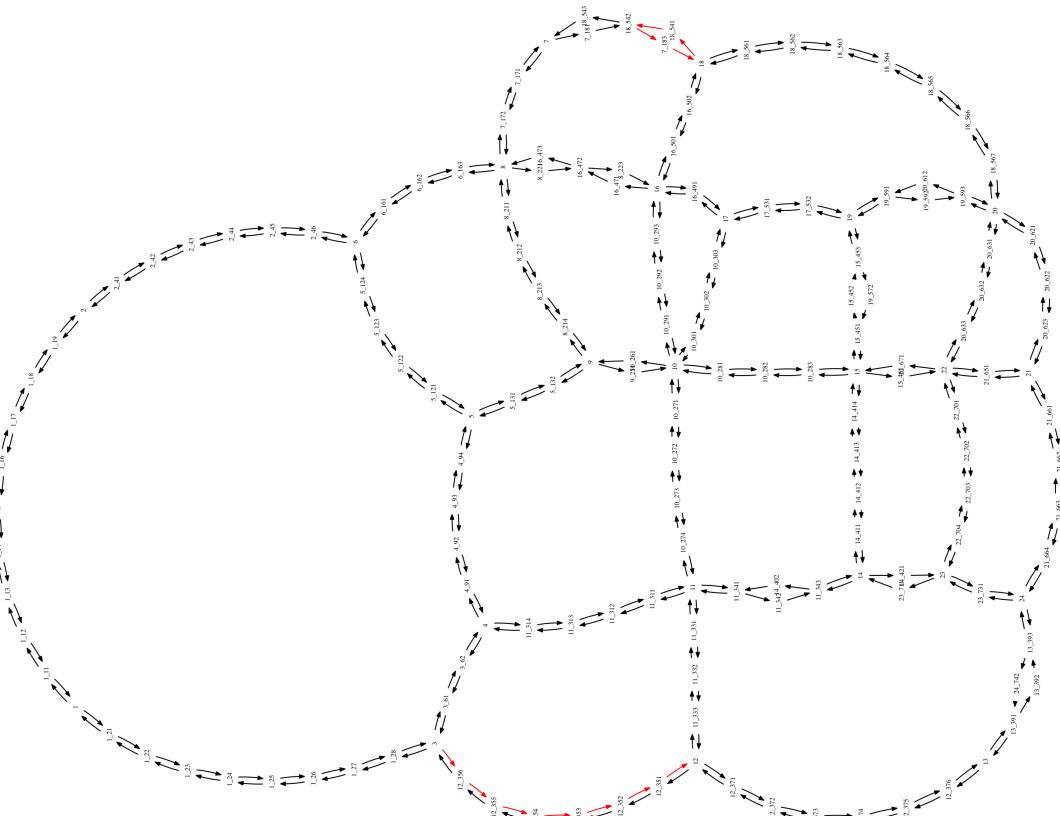
Rysunek 4.20: Sieć miasta Sioux Falls, rozwiązańe nr. 12.



Rysunek 4.21: Sieć miasta Sioux Falls, rozwiążanie nr. 13.

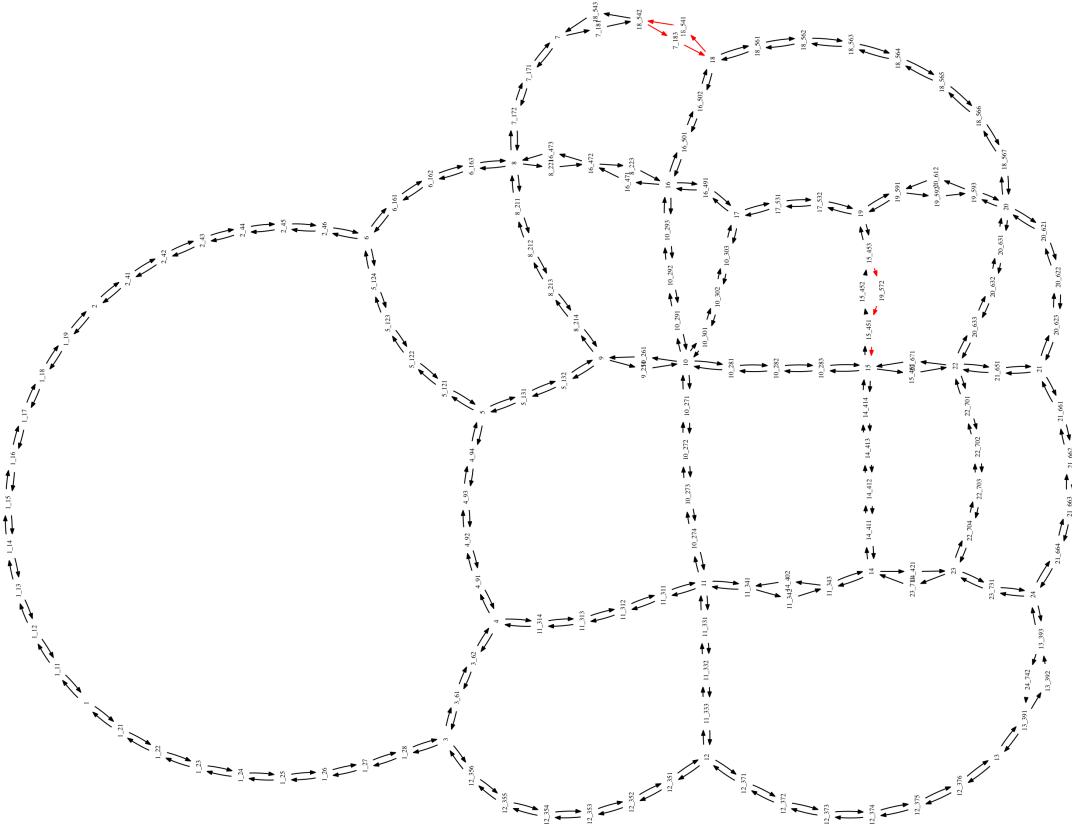


Rysunek 4.22: Sieć miasta Sioux Falls, rozwiązańe nr. 14.

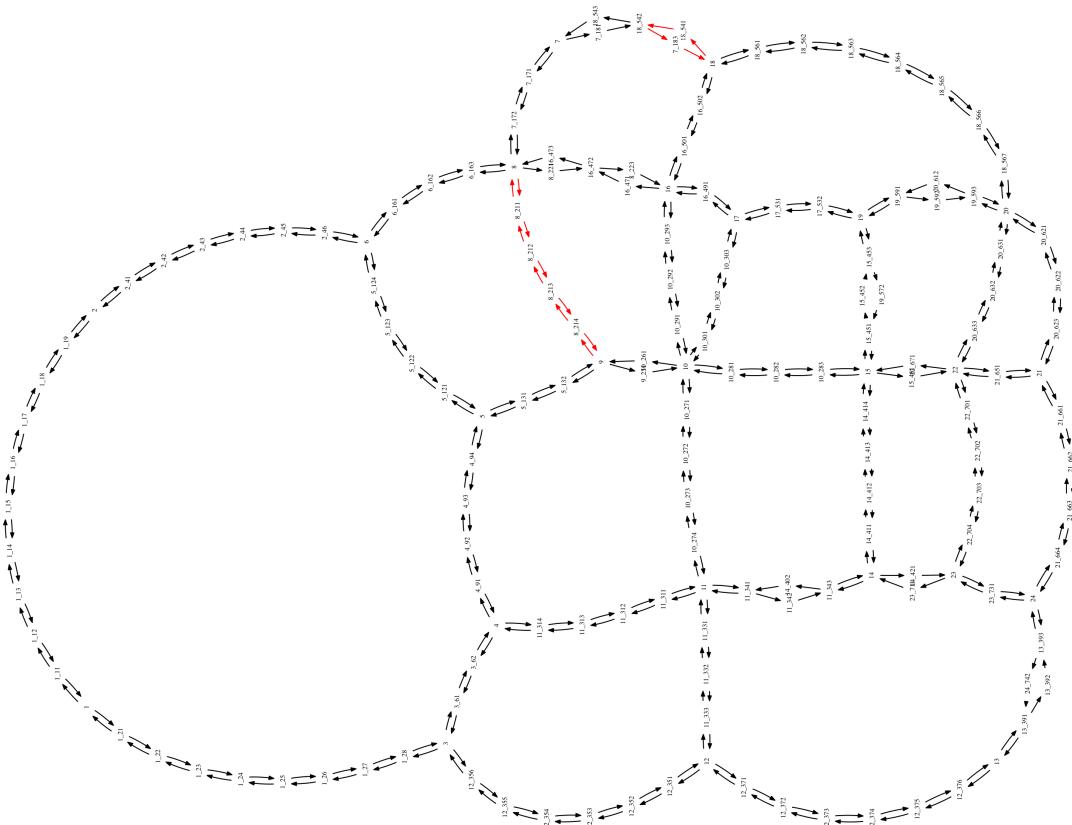


Rysunek 4.23: Sieć miasta Sioux Falls, rozwiążanie nr. 15.

4.5. WYNIKI.



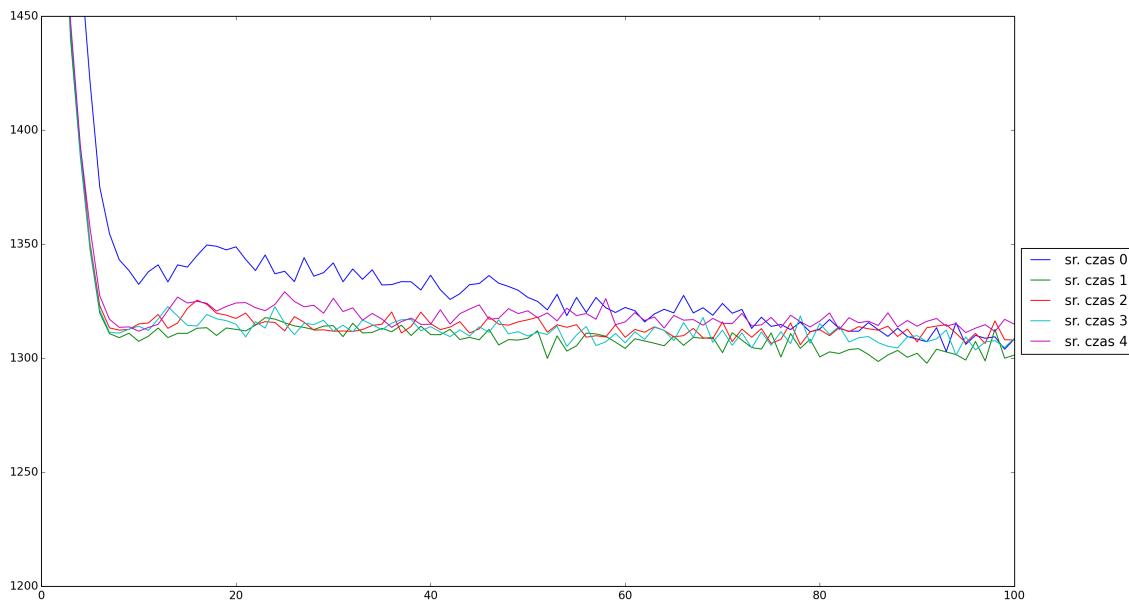
Rysunek 4.24: Sieć miasta Sioux Falls, rozwiązańe nr. 16.



Rysunek 4.25: Sieć miasta Sioux Falls, rozwiążanie nr. 17.

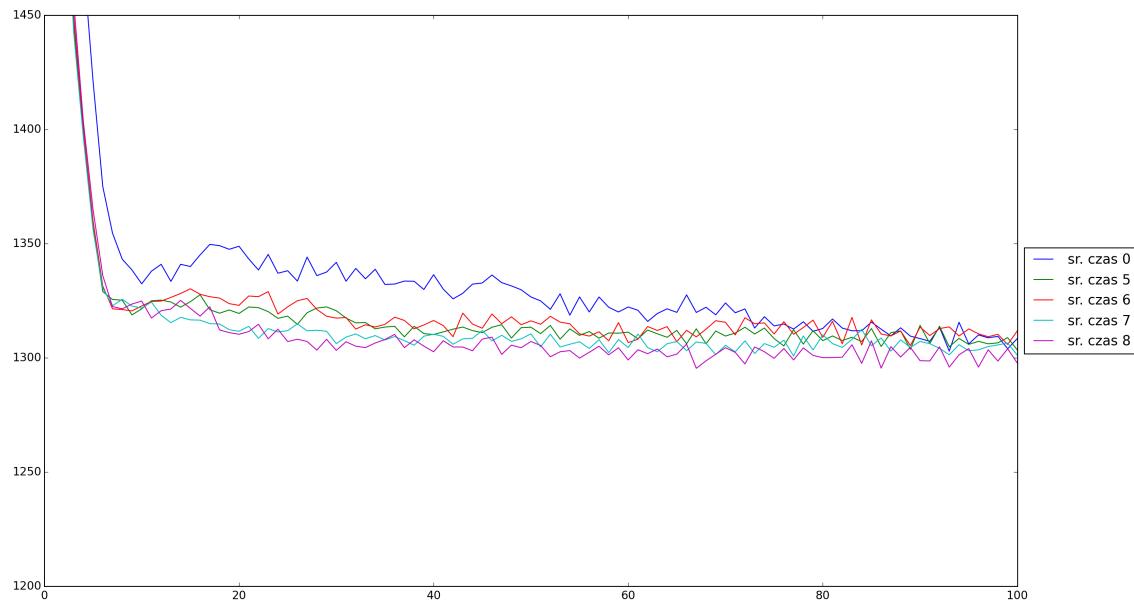
4.6 Analiza symulacji

Zgodnie z wcześniejszymi założeniami, podczas poszukiwań wybrałem ustawienie 10 iteracji symulacji, do oceny sieci. W celu zweryfikowania wyników, dla najlepszych 17 sieci przeprowadziłem ponowną symulację dla 100 iteracji. Wyniki te porównujemy z wynikami sieci wejściowej. Rysunki 4.26 - 4.30 przedstawiają wyniki tych porównań. Dla poprawienia czytelności wykresy zostały rozdzielone. Numer sieci 0 odnosi się do sieci wejściowej, natomiast pozostałe numery, do poszczególnych numerów ID sieci wybranych przez algorytm.

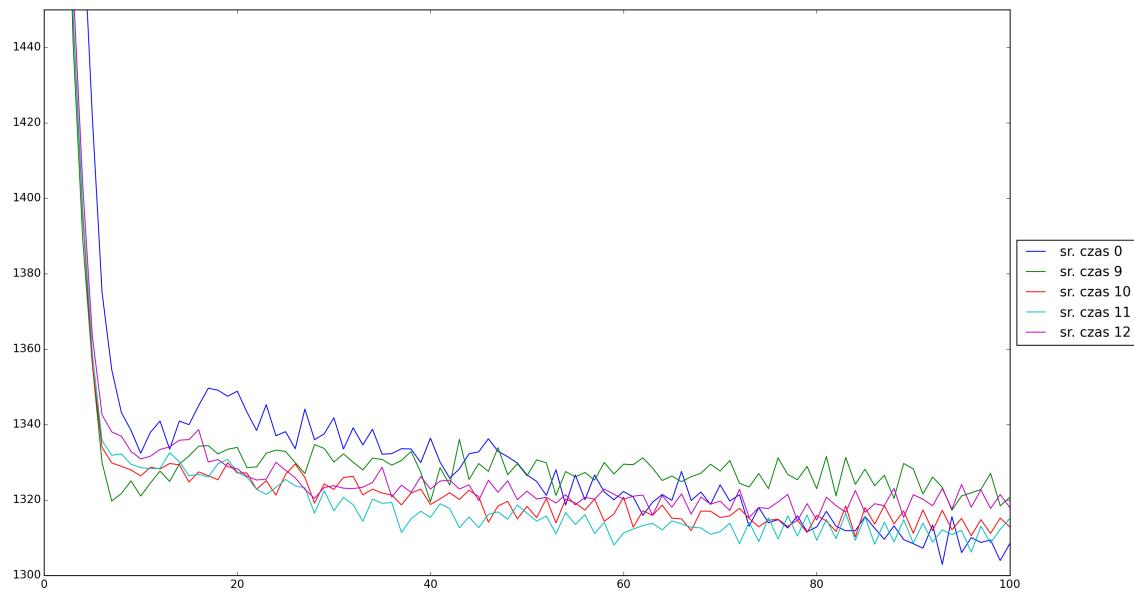


Rysunek 4.26: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.

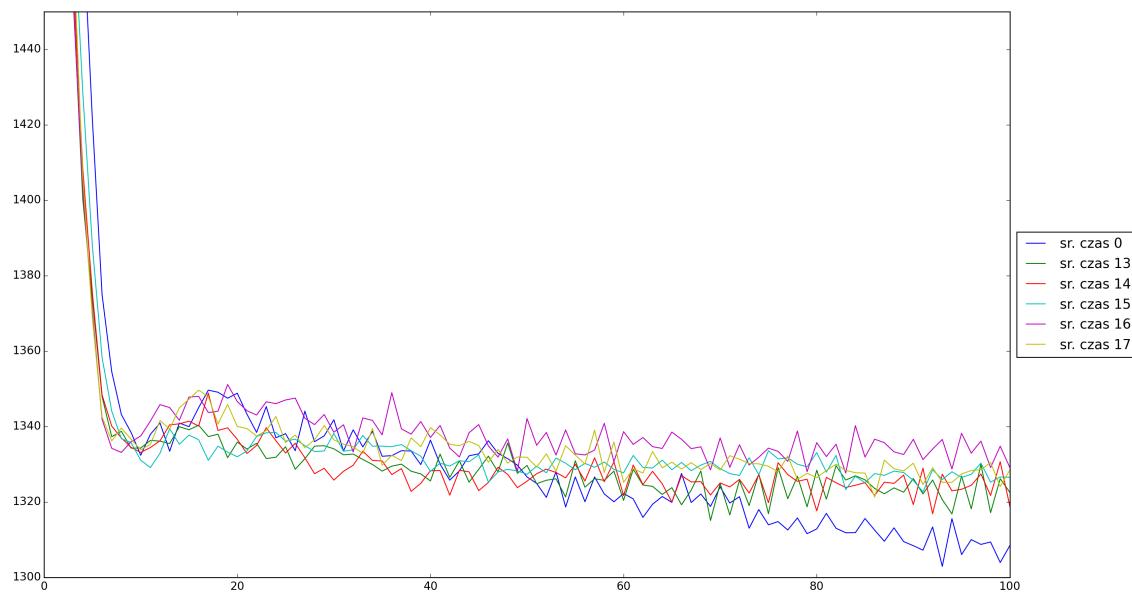
4.6. ANALIZA SYMULACJI



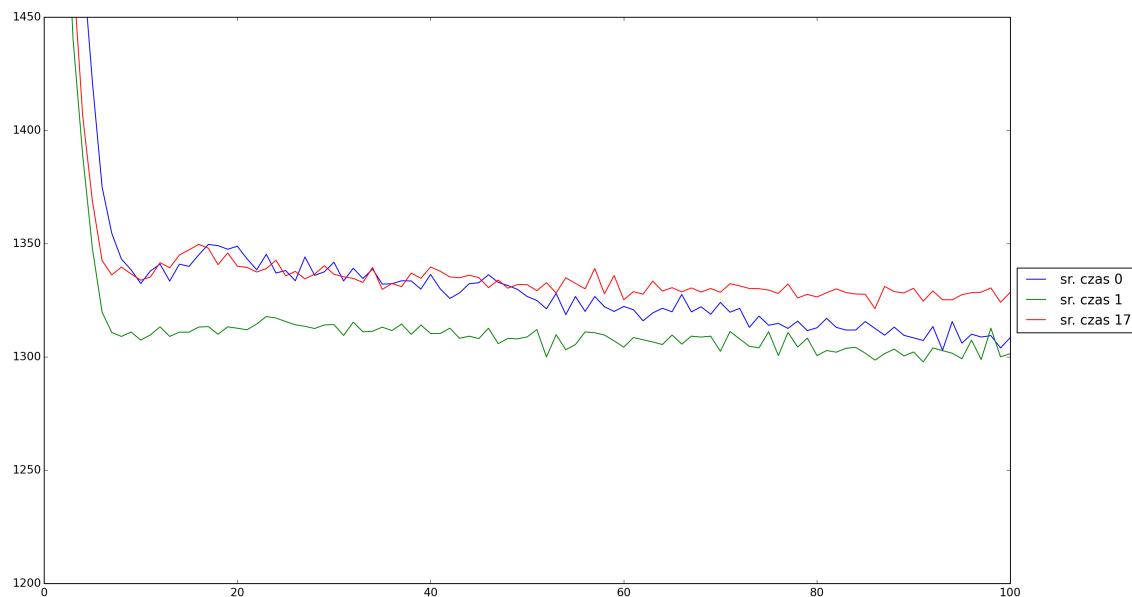
Rysunek 4.27: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.



Rysunek 4.28: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.



Rysunek 4.29: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.



Rysunek 4.30: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci najlepszej z ujęciem sieci wejściowej.

Rozdział 5

Podsumowanie.

Uzyskane wyniki potwierdzają przede wszystkim badawczą stronę pracy. Dla zadanej sieci udało się zoptymalizować ruch drogowy, stosując zamknięcie określonych węzłów. Trzeba jednak pamiętać, co już zostało wspomniane, że wynik jest czysto teoretyczny. Symulacja nie odzwierciedla prawdziwego ruchu drogowego w mieście a ponadto zastosowaliśmy jedynie uproszczenie, w którym nie są obecne wszystkie węzły rzeczywistej sieci drogowej. W związku z powyższym, uzyskane wyniki, można traktować co najwyżej jako sugestię.

5.1 Dyskusja wyników.

Gdyby założyć, że system ten miałby dostarczyć pewnej sugestii w sprawie podjęcia decyzji dotyczącej usprawnienia ruchu, spełnia on swoje zadanie. Przeszukiwanie przestrzeni rozwiązań wykorzystując algorytmy genetyczne, nie jest może najefektywniejszym przykładem, jednak skutecznym. Działanie podparte sumienną symulacją może przekazać miarodajne wyniki.

Symulator MATSim zadziałał zgodnie z oczekiwaniami, wykazując dużą zdolność adaptacji agentów do sieci. Przypadek ten można opisać, zadając sobie pytanie, skąd właściwie biorą się korki i zatory drogowe? Są one zwykle efektem braku poprawnego planu podróży uczestników ruchu. Symulator potrafi znaleźć taki plan, który w sposób idealny, dobiera czas podróży oraz najlepszą drogę. Jest to oczywiście możliwe dzięki powtarzającym się w każdej iteracji **tym samym** warunkom drogowym panującym w sieci. Nie jest to więc sytuacja rzeczywista.

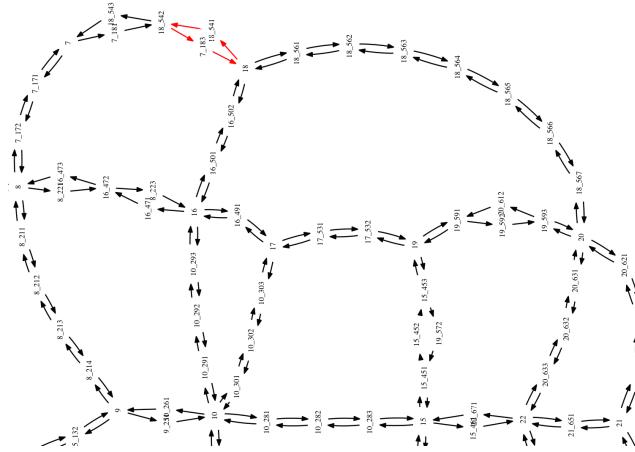
Mając na uwadze powyższe, wybory w iteracjach wcześniejszych są mniej idealne, można powiedzieć, bardziej ludzkie. Wykresy 4.26 - 4.30 wyraźnie pokazują, że dla zmodyfikowanej sieci, znacznie trudniej jest popełnić błąd przy planowaniu trasy, mając

ograniczoną wiedzę na temat warunków drogowych. Wymuszenie więc określonych dróg na agentach pozwoliło im szybciej dokonywać prawidłowych wyborów.

W przypadku długiej symulacji, część sieci zmodyfikowanych osiągała gorszy rezultat niż sieć wejściowa. Oprócz wyżej wymienionych czynników, musimy bowiem pamiętać, że generalnie większa liczba węzłów oznacza większe możliwości przepływowne sieci. Efekt dłuższej symulacji można porównać z dopasowaniem sieci drogowej do potrzeb zadanego modelu agentów. Taka sieć spełniałaby idealnie wymogi danej symulacji. Nie sprawdziłaby się, jednak gdyby model został zmieniony. W przypadku długiej nauki agentów, na temat danej sieci, mamy sytuację odwrotną.

5.2 Analiza wyników.

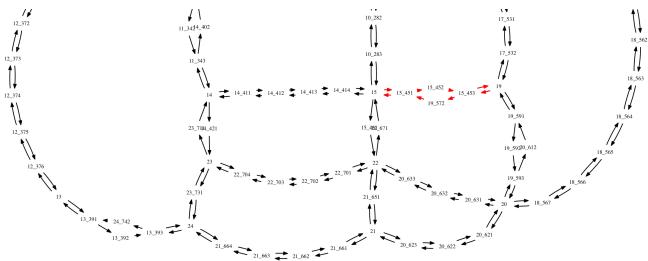
Pierwszym krokiem, który pomaga w analizie otrzymanych wyników jest ich zgrupowanie. Dość intuicyjnym jest grupowanie na zbiory, zamkijające podobne obszary (węzły). Na pierwszy rzut oka widać wyraźną zbieżność w jednym rejonie. Niestety, choć zdecydowanie musi mieć on wpływ na poprawę czasu symulacji, nie wydaje się on punktem „strategicznym” komunikacji miejskiej. Mowa oczywiście o zamknięciu węzłów między 18 i 18_542. Przedstawiamy zaznaczony fragment na rysunku 5.1, obróconym o 90° w lewo.



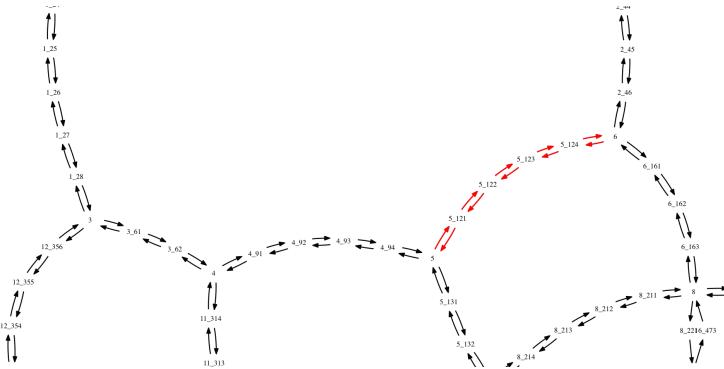
Rysunek 5.1: Fragment grafu z zaznaczonym obszarem 1.

Ciekawym jest jednak obszar wspólny dla wyników 4.13, 4.15, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.24. Jest to obszar znajdujący się pomiędzy wspomnianymi wierzchołkami 15 i 19. Przedstawiamy zaznaczony fragment na rysunku 5.2.

Drugim obszarem, który został wybrany w przypadku paru rozwiązań, są ulice pomiędzy skrzyżowaniem (wierzchołkiem) 5 i 6. Został on wybrany przez 4.10, 4.12, 4.14, 4.16, 4.20. Przedstawiamy zaznaczony fragment na rysunku 5.3.

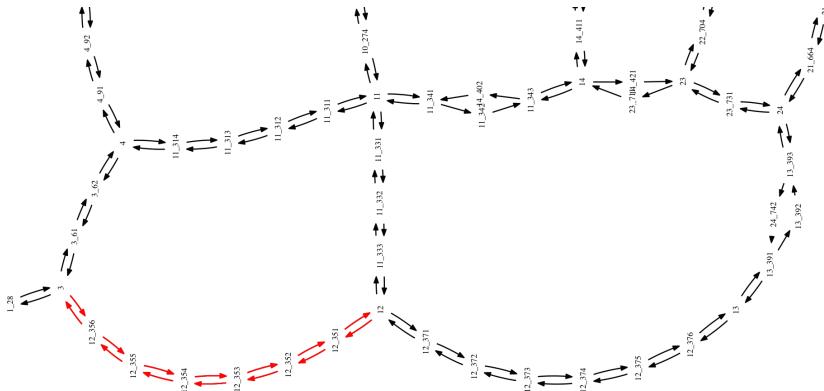


Rysunek 5.2: Fragment grafu z zaznaczonym obszarem 2.



Rysunek 5.3: Fragment grafu z zaznaczonym obszarem 3.

Ostatni obszar, który charakteryzował się wspólnym wynikiem dla więcej niż jednego rozwiązania, tj: 4.11, 4.16, 4.23 znajduje się pomiędzy wierzchołkiem 3 i 12. Przedstawiamy zaznaczony fragment na rysunku 5.4, obróconym o 90° w lewo.



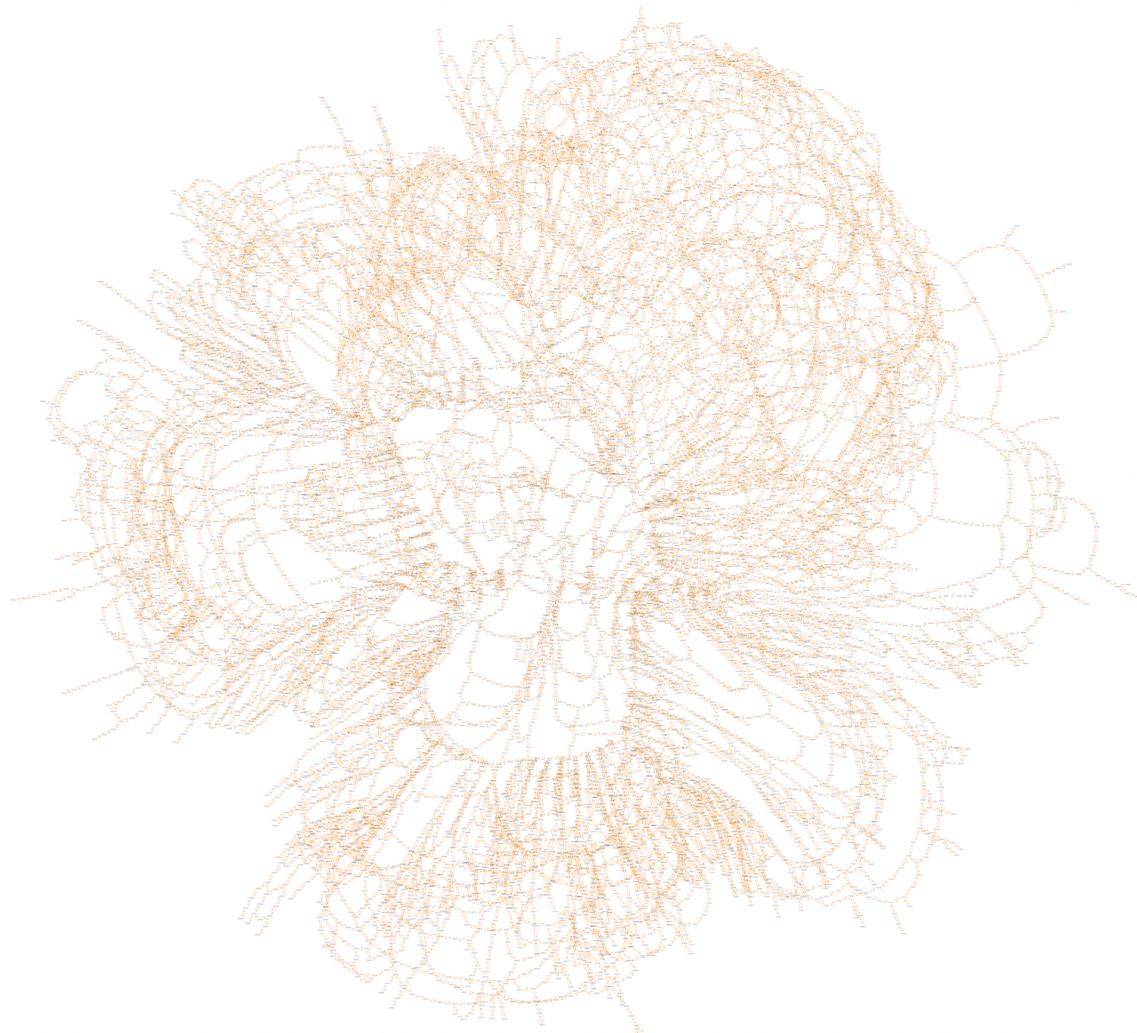
Rysunek 5.4: Fragment grafu z zaznaczonym obszarem 4.

Powyższe sugestie mogą zostać użyte w celu poszerzonych badań nad danymi obszarami, decydując czy faktycznie mają one wpływ na komunikację w mieście. Otrzymane wyniki nie oznaczają również, że nie istnieje bardziej optymalne rozwiązanie. Przestrzeń przeszukiwań dla powyższego grafu wynosiła 2^{90} , a więc zbadanie wszystkich możliwych rozwiązań problemu jest praktycznie niewykonalne.

5.3 Perspektywy dalszych badań w dziedzinie.

Dość oczywistym kierunkiem rozwoju może być oczywiście dalsze przeszukiwanie posiadanego problemu. Jak już zostało wspomniane wyżej, ilość dostępnych rozwiązań jest bardzo duża.

Ciekawszym wydaje się być jednak przeszukiwanie nowych zbiorów. W przypadku symulatora MATSim mamy do swojej dyspozycji dość pokaźną liczbę przykładów opartych o rzeczywiste miasta, na których były prowadzone symulacje. Jeden z dostępnych modeli jest odwzorowaniem miasta Berlin, przy okazji, wykonanym z dużo większą dokładnością niż Sioux Falls 5.5.



Rysunek 5.5: Sieć drogowa miasta Berlin w postaci grafu.

Ponadto, w pracy można wykorzystać inne twierdzenia i paradoksy dotyczące m.in. transportu miejskiego. Niestety, jak już było wspomniane wcześniej, wiele z tych praw jest opartych o psychologiczne tezy, bez matematycznego podparcia, co powoduje, że wyniki są trudne do przewidzenia.

5.4 Struktura projektu.

Załączone na płycie źródła pozwalają na odtworzenie pełnego projektu używając wymienionych wcześniej środowisk programistycznych, Eclipse [16] i PyDev [18]. Poniżej przedstawiona zostaje struktura katalogów, w kolejności alfabetycznej, wraz z krótkim opisem zawartości.

fakematsim/

Jest to sztuczna implementacja symulatora pozwalająca na testowanie działania algorytmu genetycznego w oparciu o losowe wyniki. Działanie opiera się o rozpakowanie gotowego folderu z wcześniej przygotowanymi obliczeniami oraz podmianę wyniku na losową liczbę. Projekt jest przygotowany w oparciu o strukturę projektu *Maven*. Jego skompilowane źródła znajdują się w katalogu *matsim*.

java/

Tutaj znajduje się główny projekt zarządzający aplikacją obliczeniową. Jest on przygotowany w oparciu o strukturę projektu *Maven*. Zawiera przykładowe pliki konfiguracyjne i przygotowane testy *JUnit* pozwalające na ich bezpośrednie wykorzystanie.

literature/

W tym folderze zostały zebrane wszystkie źródła literaturowe wykorzystane przy tworzeniu pracy.

matsim/

Folder ten zawiera skompilowane wersje symulatora MATSim oraz jego falsyfikatu wykorzystywanego podczas testów. Symulator został skompilowany ze źródeł dostępnych na repozytorium głównym projektu¹. Drobne modyfikacje dotyczyły tylko parametrów logowania oraz uruchomienia symulatora. Nie zostały dokonane żadne zmiany ingerujące w przebieg samej simulacji.

sioux-out/

Jest to domyślny katalog z danymi otrzymanymi w wyniku działania projektu.

paper/

Katalog ze źródłami pracy pisemnej w formacie TeX.

¹<https://svn.code.sf.net/p/matsim/source/matsim/trunk>

5.4. STRUKTURA PROJEKTU.

python/

Znajdują się tutaj wszystkie wykorzystane w projekcie skrypty Python wraz z testami jednostkowymi. Jest to również katalog domyślny wywołań skryptów podczas obliczeń.

README.md

Plik zawierający opis instalacji wymaganych przez projekt zależności i bibliotek na podstawie systemu Linux Ubuntu 12.04 LTS.

scenarios/

Zawiera przykładowe scenariusze, które mogą być wykorzystane przy pracy z symulatorem MATSim. Znajduje się tutaj, oprócz wykorzystanego miasta Sioux Falls, Berlin i Bruksela. Ponadto zawiera parę przykładowych plików konfiguracyjnych symulatora.

seminar/

Zawiera prezentację wykorzystaną podczas seminarium dyplomowego, przedstawiającą wstępne założenia projektu.

Spis rysunków

2.1	Przykładowy graf nieskierowany	9
2.2	Przykładowy graf skierowany.	10
2.3	Fragment sieci drogowej miasta Sioux Falls, Południowa Dakota.	11
2.4	Siec drogowa miasta Sioux Falls w postaci grafu.	11
2.5	Graf z dopasowaną geometrią.	11
2.6	Wyjściowy układ drogowy	13
2.7	Uzupełniony układ drogowy	14
2.8	Przykładowy graf z zaznaczonymi składowymi silnie spójnymi.	15
2.9	Ogólny schemat algorytmu genetycznego.	18
2.10	Ogólny schemat operacji krzyżowania.	18
2.11	Ogólny schemat operacji mutacji.	19
2.12	Fragment sieci w postaci tablicy binarnej	19
2.13	Fragment sieci w postaci grafu	19
2.14	Przykład grafu z zaznaczonym punktem artykulacji.	20
3.1	Logo symulatora transportu MATSim	21
3.2	Logo biblioteki Apache Commons Math	22
3.3	Logo biblioteki NetworkX	22
3.4	Logo Java.	23
3.5	Logo IDE Eclipse.	23
3.6	Logo Python.	23
3.7	Logo PyDev.	23
3.8	Logo systemu Linux Ubuntu.	23
3.9	Logo chmury Microsoft Windows Azure.	24
4.1	Graf sieci miasta Sioux Falls wykorzystany w badaniach.	28
4.2	Rozkład budynków na grafie miasta Sioux Falls.	29
4.3	Natężenie ruchu w mieście Sioux Falls w godzinach 6.00-7.00.	30
4.4	Natężenie ruchu w mieście Sioux Falls w godzinach 16.00-17.00.	30
4.5	Graf czasu trwania symulacji MATSim dla stu iteracji.	31

SPIS RYSUNKÓW

4.6	Graf średniego czasu przejazdów dla stu iteracji.	32
4.7	Graf wyników agentów dla stu iteracji.	33
4.8	Graf wyników algorytmu genetycznego.	34
4.9	Sieć miasta Sioux Falls, rozwiązanie nr. 1.	36
4.10	Sieć miasta Sioux Falls, rozwiązanie nr. 2.	37
4.11	Sieć miasta Sioux Falls, rozwiązanie nr. 3.	37
4.12	Sieć miasta Sioux Falls, rozwiązanie nr. 4.	38
4.13	Sieć miasta Sioux Falls, rozwiązanie nr. 5.	38
4.14	Sieć miasta Sioux Falls, rozwiązanie nr. 6.	39
4.15	Sieć miasta Sioux Falls, rozwiązanie nr. 7.	39
4.16	Sieć miasta Sioux Falls, rozwiązanie nr. 8.	40
4.17	Sieć miasta Sioux Falls, rozwiązanie nr. 9.	40
4.18	Sieć miasta Sioux Falls, rozwiązanie nr. 10.	41
4.19	Sieć miasta Sioux Falls, rozwiązanie nr. 11.	41
4.20	Sieć miasta Sioux Falls, rozwiązanie nr. 12.	42
4.21	Sieć miasta Sioux Falls, rozwiązanie nr. 13.	42
4.22	Sieć miasta Sioux Falls, rozwiązanie nr. 14.	43
4.23	Sieć miasta Sioux Falls, rozwiązanie nr. 15.	43
4.24	Sieć miasta Sioux Falls, rozwiązanie nr. 16.	44
4.25	Sieć miasta Sioux Falls, rozwiązanie nr. 17.	44
4.26	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.	45
4.27	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.	46
4.28	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.	46
4.29	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych z ujęciem sieci wejściowej.	47
4.30	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci najlepszej z ujęciem sieci wejściowej.	47
5.1	Fragment grafu z zaznaczonym obszarem 1.	50
5.2	Fragment grafu z zaznaczonym obszarem 2.	51
5.3	Fragment grafu z zaznaczonym obszarem 3.	51
5.4	Fragment grafu z zaznaczonym obszarem 4.	51
5.5	Sieć drogowa miasta Berlin w postaci grafu.	52

Spis tabelic

4.1 Zbiór najlepszych otrzymanych sieci.	35
--	----

SPIS TABLIC

Spis listingów

4.1 Plik konfiguracyjny projektu	25
4.2 Ustawienia algorytmu genetycznego podczas badań.	33

SPIS LISTINGÓW

Bibliografia

- [1] Leslie Arthur Keith Bloy, *An investigation into Braess' paradox*, 02/2007
- [2] Rric Pas and Shari Principio *Braess' paradox: Some new insights*, April 1996
- [3] 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.
- [4] Ana L. C. Bazzan and Franziska Klügl *Reducing the Effects of the Braess Paradox with Information Manipulation*
- [5] Dietrich Braess. *Über ein Paradoxon aus der Verkehrsplanung. „Unternehmensforschung“*. 12, s. 258–268, 1968 (niem.).
- [6] Mitchell Melanie *An Introduction to Genetic Algorithms* First MIT Press paperback edition, 1998
- [7] 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
- [8] A. Chakirov *Enriched Sioux Falls Scenario with Dynamic Demand* MATSim User Meeting, Zurich/Singapore, June 2013.
- [9] Łukasz Kowalik *Algorytmy i struktury danych, grafy* Wykład, 2003.
- [10] Marta Grzanek *Sztuczna inteligencja, Klasyczny algorytm genetyczny* Wykład, 2007.
- [11] Dilvan de Abreu Moreira *Agents: A Distributed Client/Server System for Leaf Cell Generation* Thesis, 1995
- [12] <http://matsim.org>
- [13] <http://commons.apache.org/proper/commonsmath>

BIBLIOGRAFIA

- [14] <http://networkx.github.io>
- [15] <http://www.java.com/pl/>
- [16] <https://eclipse.org>
- [17] <http://pl.python.org>
- [18] <http://pydev.org>
- [19] <http://www.ubuntu.com>
- [20] <http://azure.microsoft.com>
- [21] http://pl.wikipedia.org/wiki/Paradoks_Braessa
- [22] <http://urbnews.pl/paradoksbraessa/>
- [23] http://pl.wikipedia.org/wiki/Paradoks_DownsaThomsona
- [24] http://pl.wikipedia.org/wiki/Prawo_Lewisa_Mogridge'a
- [25] <http://mathworld.wolfram.com/StronglyConnectedDigraph.html>

Abstract

The purpose of the present master thesis was to find an optimal solution of the given road network using genetic algorithm. The optimisation is performed using fixed, set in advance parameters. Paper covers two main fields of the subject. Firstly it introduces the topic of road network optimisation and Braess paradox. Secondly it describes the techniques used to achieve the final goal.

The application performing the optimisation process uses an external ranking system. In this case, a multiagent simulator, MATsim. In paper we describe the aspects of the simulation concering the final results of the project.

The solution is prepared using Java and Python programming languages. For maximal portability, there is a list of required tools, available. During the research, the process was conducted by a machine operating on Linux Ubuntu 12.04 LTS.

The final result proves the validity of the introduced methods. The genetic algorithm, regarding searching of the optimal solution. And the simlation based rank using the multiagent simulator resulted in 17 solutions of the given network with given parameters.

The discussion regarding the MATSim's iteration results and configuration also explain the differences in long-term evaluation of the simulation. Since the optimal parameters for any simulation are discussable, in the thesis the main goal was to find a simple, fast solution to experienced problem. Therefore, lower average time of travel for agents during longer iterations would be a separate case of study.