



Politechnika Łódzka

Instytut Informatyki

PRACA DYPLOMOWA MAGISTERSKA

Optymalizacja struktury sieci drogowej

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Promotor: dr hab. inż. Aneta Poniszecka-Marańda

Współpromotor: mgr inż. Łukasz Chomątek

Dyplomant: inż. Michał Siatkowski

Nr albumu: 186865

Kierunek: Informatyka

Specjalność: Sztuczna Inteligencja i Inżynieria Oprogramowania

Łódź 10.04.2015

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 optymalizacji ruchu drogowego	5
1.2 Cel pracy	6
1.3 Zakres pracy	6
1.3.1 Studia literaturowe	6
1.3.2 Propozycja rozwiązania problemu optymalizacji sieci drogowej	6
1.3.3 Uwagi dotyczące proponowanego rozwiązania	7
1.3.4 Opis stworzonego rozwiązania	7
1.4 Układ pracy	7
2 Wybrane elementy optymalizacji struktury sieci drogowej	9
2.1 Paradoks Braessa	9
2.1.1 Przykładowy wyjściowy układ drogowy	10
2.1.2 Przykładowy uzupełniony układ drogowy	11
2.1.3 Wyjaśnienie intuicyjne paradoksu Braessa	12
2.2 Klasyczny algorytm genetyczny	13
2.2.1 Podstawowe pojęcia algorytmów genetycznych	14
2.2.2 Klasyczny algorytm genetyczny	15
2.2.3 Metody selekcji osobników	18
2.2.4 Mocne i słabe strony algorytmów genetycznych	19
2.3 Teoria grafów	20
2.3.1 Geneza teorii grafów	20
2.3.2 Podstawowe pojęcia dotyczące grafów	22
2.3.3 Przeszukiwanie grafów	24
2.3.4 Graf skierowany silnie spójny	26
2.4 Sieć drogowa w postaci grafu	27
2.5 Przystosowanie struktury grafu do użycia w algorytmie genetycznym	29

SPIS TREŚCI

3 Metoda oceny struktury sieci drogowej	31
3.1 Symulator transportu ruchu drogowego	31
3.1.1 Symulacja ruchu drogowego	33
3.1.2 Symulacja transportu oparta na agentach	34
3.2 Ustawienia symulatora transportu	36
3.3 Opis badanego miasta Sioux Falls	39
4 Proces realizacji projektu GRoNO	43
4.1 Technologie i metodologie programistyczne	43
4.2 Biblioteka komponentów algorytmów genetycznych	43
4.3 Biblioteka do obsługi grafów	45
4.4 Uruchomienie obliczeń w chmurze	46
4.5 Obsługa projektu GRoNO	47
4.6 Schemat działania projektu GRoNO	49
4.7 Struktura plików projektu GRoNO	52
5 Analiza procesu optymalizacji sieci drogowej	55
5.1 Użyte ustawienia projektu GRoNO	55
5.2 Wyniki optymalizacji sieci drogowej miasta Sioux Falls	56
5.3 Analiza uzyskanych wyników sieci drogowych	68
5.4 Analiza zastosowanych ustawień symulacji	73
6 Podsumowanie	79
6.1 Perspektywy dalszych badań w dziedzinie	80
Spis rysunków	80
Spis tabel	83
Spis listingów	85
Bibliografia	87
Abstract	91

Rozdział 1

Wstęp

W poniższym rozdziale zostały opisane motywy wyboru tematu pracy dyplomowej. Omówiono najważniejsze aspekty problemu optymalizacji ruchu drogowego oraz przedstawiono cel niniejszej pracy magisterskiej. Na zakończenie przybliżono strukturę pracy wraz z krótkim omówieniem kolejnych jej rozdziałów.

1.1 Problematyka optymalizacji ruchu drogowego

Problemy komunikacji w dzisiejszych miastach są wszystkim znane. Zatory drogowe i „korki” w godzinach szczytu są chlebem powszednim. Pomimo wielu prób i sposobów rozwiązania powyższego problemu, wciąż nie istnieje metoda jednoznacznie rozstrzygająca tę kwestię. Bezspornie dotyczy to wszystkich miast na świecie. Z teoretycznego punktu widzenia, jedynym rozwiązaniem jest komunikacja publiczna. Oczywistym jest jednak, że nigdy nie będzie możliwa sytuacja, 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** [18]. 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 zastosowanie w przypadku sieci komputerowych oraz istniejącą jego analogię dla doświadczeń fizycznych.

1.2 Cel pracy

Tematem niniejszej pracy dyplomowej jest optymalizacja struktury sieci drogowej. Opierając się na wspomnianym paradoksie Braessa, sformułowany został cel pracy, którym jest stworzenie rozwiązania, dokonującego optymalizacji zadanej sieci drogowej. Optymalizację przeprowadzono poprzez wykorzystanie algorytmów genetycznych, które zakładają użycie pewnych z góry ustalonych parametrów. Modyfikacja sieci drogowej powinna odbyć się poprzez zastosowanie założenia paradoksu, dokładniej przez zamknięcie wybranych ulic. W efekcie dla danej sieci drogowej, średni czas podróży powinien ulec skróceniu.

1.3 Zakres pracy

Poniżej przedstawiono zakres prac wykonanych w celu zgłębienia tematu problematyki optymalizacji ruchu drogowego. Zaprezentowano teoretycznie zagadnienia użyte w pracy oraz krótko przedstawiono stworzone rozwiązanie.

1.3.1 Studia literaturowe

Badania rozpoczęto 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 [19]. Udowodniono również jego występowanie w innych dziedzinach życia, co czyni go jedynie bardziej interesującym. Przykładem może być eksperyment przeprowadzony przez Max Planck Institute for Dynamics and Self-Organization. Poprzez symulację, udowodniono że zjawisko to występuje w sieciach elektrycznych, w których produkcja energii jest zdecentralizowana [27]. Z kolei badania międzynarodowego zespołu naukowców z Institut Néel oraz IEMN¹, INP² i UCL³ w 2012 roku dowiodły występowanie paradoksu w mezoskopowych systemach elektronowych [28].

1.3.2 Propozycja rozwiązania problemu optymalizacji sieci drogowej

Dla zastosowań transportu drogowego oczywistym rozwiązaniem problemu komunikacji jest stworzenie idealnej sieci odpowiadającej potrzebom danego miasta. Warto

¹Centre national de la recherche scientifique, Francja

²Institut polytechnique de Grenoble, Francja

³Université catholique de Louvain, Belgia

jednak zauważyc, że rozbudowa lub modyfikacja tej infrastruktury jest kosztowna i czasochłonna. Z tego powodu w niniejszej pracy dyplomowej sprawdzono rozwiązanie zaproponowane przez Braessa. Ponieważ istnieją prace negujące lub podważające paradoks [8], zdecydowano, by przy potwierdzaniu wyniku optymalizacji nie kierować się wyłącznie w nim zawartymi założeniami. W celu sprawdzenia jego słuszności, wybrany został zewnętrzny sposób oceniania. Taką rolę spełnia system symulacji ruchu drogowego. W efekcie ocena rozwiązania jest niezależna od metody twierdzenia, poprzez symulację rzeczywistego ruchu w pewnej sieci drogowej. Wynik symulacji jest jednoznaczną wartością liczbową, przedstawiającą średni czas przejazdów wszystkich pojazdów, biorących udział w danym scenariuszu. Zakładając niezmienne plany docelowe przejazdów, poprzez manipulację strukturą drogową, dąży się oczywiście do minimalizacji średnich ich czasów.

Ponieważ nie znaleziono przesłanek wskazujących na jednoznaczną ocenę co do słuszności zamknięcia danej ulicy, w niniejszej pracy zdecydowano się na losowe przeszukiwanie przestrzeni rozwiązań. Jednym z dostępnych rozwiązań w przypadku takich poszukiwań są algorytmy genetyczne, które zostały wykorzystane w pracy.

1.3.3 Uwagi dotyczące proponowanego rozwiązania

Paradoks Braessa nie jest jedynym twierdzeniem traktującym o problemach komunikacyjnych miast. Wiele teorii jest opartych głównie na socjologicznych lub psychologicznych założeniach. Są to na przykład paradoks Downsa Thomsona [19] lub prawo Lewisa Mogridge'a [20]. Niestety, biorąc pod uwagę złożoność problemu, są one niemniej ważne [5]. Zatem wynik otrzymany podczas eksperymentu nie może być dowodem ani decydującym głosem w decyzjach dotyczących ustalania rzeczywistej sieci drogowej miasta.

1.3.4 Opis stworzonego rozwiązania

W efekcie powyższych badań stworzono rozwiązanie, które przy pomocy algorytmów genetycznych dokonuje optymalizacji zadanej sieci drogowej. Rozwiązanie, **projekt wykonany w ramach niniejszej pracy dyplomowej został nazwany GRoNO**, z angielskiego *Genetic Road Network Optimiser*.

1.4 Układ pracy

Na początku niniejszej pracy przedstawione zostały zagadnienia dotyczące trudności komunikacyjnych miast oraz znane i wykorzystywane powszechnie ich rozwiązania.

1.4. UKŁAD PRACY

Przybliżony został cel pracy wraz z opisem zaproponowanej metody optymalizacji problemu.

Rozdział 2 zawiera opis teoretyczny zagadnień związanych z optymalizacją sieci drogowych. Omówiony został wykorzystywany paradoks Braessa. Przedstawiono metodę optymalizacji oraz wykorzystywane struktury matematyczne. Objaśniono prawa stosowane podczas ich modyfikacji.

W rozdziale 3 opisano metodę zastosowaną w celu oceny struktury sieci drogowej. Omówiono działanie wybranego symulatora transportu drogowego. Przeanalizowano wyniki uzyskane podczas próbnych symulacji i zaprezentowano wnioski wykorzystane do konfiguracji na potrzeby optymalizacji sieci drogowej. Ostatecznie, przedstawiono miasto użyte podczas badań, które zostało dostarczone przez twórców symulatora.

Wszelkie zagadnienia teoretyczne związane ze stworzonym projektem *GRoNO* omówiono w rozdziale 4. Wymieniono biblioteki użyte w projekcie. Krótko omówiono aspekty implementacji projektu oraz jego uruchomienia. Przedstawiono również strukturę plików dostarczonych na płycie razem z niniejszą pracą dyplomową.

Rozdziałem pracy, w którym przedstawiono praktyczne aspekty wykonanego rozwiązania jest rozdział 5. Omówiono parametry uruchomienia projektu *GRoNO* wykorzystane podczas badań wraz z ich wartościami. Zaprezentowano wyniki optymalizacji przykładowej sieci drogowej. Rezultaty zawierają omówienie i analizę.

W rozdziale 6, podsumowującym, zawarte są wnioski wraz z omówieniem wyników badań. Prezentowane są możliwości rozwoju pracy.

Rozdział 2

Wybrane elementy optymalizacji struktury sieci drogowej

Poniższy rozdział zawiera opis teoretyczny zagadnień matematycznych wykorzystanych podczas realizacji projektu *GRONO*. Przedstawiono opis wykorzystywanego paradoksu Braessa. Następnie opisano wybraną metodę optymalizacji, czyli algorytmy genetyczne. Kolejny podrozdział opisuje teorię grafów. Przybliżono w nim użyte struktury — grafy oraz prawa wykorzystane podczas ich modyfikacji. Na koniec przedstawiono sposób reprezentacji sieci drogowej jako grafu oraz użycia tych grafów w algorytmach genetycznych.

2.1 Paradoks Braessa

Jak już wcześniej wspomniano, *paradoks Braessa* 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 [18]. 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, a 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 pojazdów pokonujących dany odcinek.
4. Układ sieci drogowej i czasy przejazdu poszczególnych odcinków są znane pojazdom.

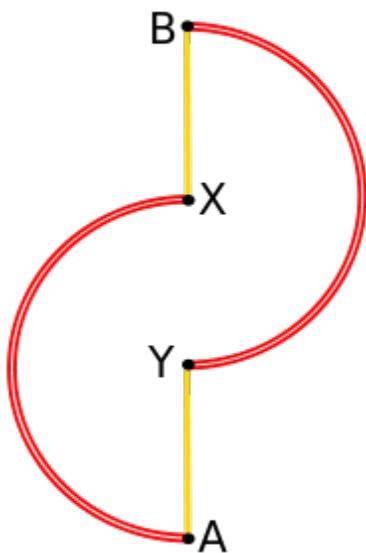
5. Celem pojazdów jest przejazd przez sieć z węzłów początkowych do docelowych po trasie złożonej z odcinków drogowych tak, by zminimalizować łączny czas ich pokonania.
6. Decyzję o wyborze tras pojazdy podejmują indywidualnie i niezależnie od siebie.

2.1.1 Przykładowy wyjściowy układ drogowy

Paradoks w oryginalnym artykule wy tłumaczone jest na prostym przykładzie, który został zaprezentowany poniżej [6, 7]. Za punkt wyjściowy przyjmuje się pewny układ dróg o znanych czasach przejazdów.

Sieć drogowa i auta

Sytuacja, w której ujawnia się paradoks Braessa jest skonstruowany z czterech miast A , B , X i Y . Są one połączone odcinkami drogowymi, jak na rysunku 2.1, z odpowiednimi czasami przejazdu, przy czym p oznacza gęstość ruchu w tysiącach aut.



Rys. 2.1: Schemat przykładowego wyjściowego układu drogowego

Poniżej znajduje się opis dostępnej sieci drogowej przedstawionej na rysunku 2.1.

Autostrady to:

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

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

Drogi lokalne to:

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

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

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

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 dotyczącą wyboru trasy przy niezmienionych decyzjach pozostałych aut. Jeśli p i q oznaczają liczby aut w tysiącach, pokonujących odpowiednio trasę AXB i AYB , to na podstawie danych do rysunku 2.1 otrzymuje się 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 równania jest $p = q = 3$. Wynik może zostać wykorzystany do obliczenia średniego czasu przejazdu aut po dostępnej sieci drogowej. Przy tej gęstości ruchu pokonanie obu dostępnych tras zabiera $50 + 3 + 30 = 83$ minuty.

2.1.2 Przykładowy uzupełniony układ drogowy

Pamiętając o początkowym stanie dróg i średnim czasie przejazdu, układ drogowy zostaje uzupełniony o autostradę, mającą na celu skrócenie czasu potrzebnego na podróż pomiędzy punktami w grafie.

Sieć drogowa i auto

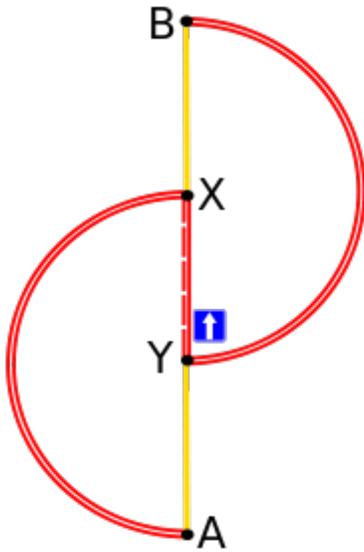
Do wyjściowego układu drogowego dodana zostaje autostrada YX . Uzupełniony układ drogowy zaprezentowany jest na rysunku 2.2.

Poniżej znajduje się opis dotyczący uzupełnionego fragmentu sieci drogowej, przedstawionej na rysunku 2.2.

Autostrady to:

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

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



Rys. 2.2: Schemat przykładowego uzupełnionego układu drogowego

Analiza równowagi Nasha

Jeśli p , q i r oznaczają liczby aut w tysiącach pokonujących odpowiednio trasę AXB , AYB i $AYXB$, to na podstawie danych do rysunku 2.2 otrzymuje się równania:

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

Rozwiązaniem równania jest $p = q = r = 2$. Wynik może zostać wykorzystany do obliczenia średniego czasu przejazdu aut po dostępnej sieci drogowej. Czas przejazdu dla każdej z tych dróg wynosi wówczas $50 + 2 + 10 \cdot (2 + 2) = 92$ minuty.

2.1.3 Wyjaśnienie intuicyjne paradoksu Braessa

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 zagęszcza ruch na drogach lokalnych, powodując wydłużenie czasu podróży [16].

W 1983 roku Steinberg i Zangwill przeprowadzili eksperyment, który miał na celu oszacować występowanie paradoksu Braessa. Zaskakujący wynik, dla losowo dodawanych łącz potwierdził skomplikowanie problemu. Przede wszystkim udowodnione zostało, że przy zastosowaniu odpowiednich założeń, dla dowolnej sieci transportowej możliwe jest wystąpienie paradoksu. Niestety, szacunkowo pojawienie się paradoksu Braessa jest czysto losowe [9]. Paradoks mimo wszystko został z powodzeniem zastosowany w wielu miastach na świecie:

- Korea Południowa, Seul. Zostało zaobserwowane przyśpieszenie ruchu w mieście po usunięciu autostrady w ramach projektu renowacji okolicy Cheonggyecheon.
- Niemcy, Stuttgart. Zostały zlikwidowane drogi zbudowane w latach 60-tych XX wieku.
- USA, Nowy Jork. Czasowe zamknięcie ulicy 42 zniwelowało ilość zatorów w okolicy.
- W 2008 roku Youn, Gastner i Jeong wykazali specyficzne drogi w Bostonie, Nowym Jorku i Londynie, których zamknięcie może zmniejszyć przewidywany czas podróży.
- Kanada, Winnipeg.

2.2 Klasyczny algorytm genetyczny

Dla dużej klasy ważnych zadań nie opracowano dotąd dostatecznie szybkich algorytmów ich rozwiązywania. Dla takich zadań optymalizacji często można znaleźć wydajny algorytm, który nie gwarantuje jednak uzyskania rozwiązania optymalnego, a tylko w przybliżeniu optymalne. Dla niektórych trudnych zadań optymalizacji można używać algorytmów probabilistycznych jednak one również nie gwarantują, że zostanie uzyskane rozwiązanie optymalne.

Ogólnie, jakąkolwiek czynność do wykonania można przedstawić jako rozwiązywanie zadania, co można z kolei rozumieć jako przeszukiwanie przestrzeni możliwych rozwiązań. Ponieważ zależy nam na „najlepszym” rozwiązaniu, można uważać to zadanie za proces optymalizacji. W małych przestrzeniach klasyczne metody pełnego przeszukiwania zwykle wystarczają. W większych przestrzeniach trzeba niestety stosować specjalizowane metody sztucznej inteligencji. Wśród tych metod znajdują się algorytmy genetyczne [2].

2.2.1 Podstawowe pojęcia algorytmów genetycznych

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 informację genetyczną swoim potomkom. Krzyżowanie informacji genetycznej otrzymanej od „rodziców” prowadzi do sytuacji, w której kolejne pokolenia są przeciętnie coraz lepiej dostosowane do warunków środowiska. Zachodzi tu więc swoisty proces optymalizacji [1].

Definicja 1 *Populacją nazywa się zbiór osobników o określonej liczbieności.*

Definicja 2 *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 3 *Chromosomy, inaczej łańcuchy lub ciągi kodowe, to uporządkowane ciągi genów.*

Definicja 4 *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 5 *Fenotyp jest zestawem wartości, odpowiadających danemu genotypowi, czyli zdekodowaną strukturą, a więc jest zbiorem parametrów zadania (rozwiązaniem, punktem przestrzeni poszukiwań).*

Definicja 6 *Allel to wartość danego genu, określona jako wartość cechy lub wariant cechy.*

Definicja 7 *Locus to pozycja, która wskazuje miejsce położenia danego genu w łańcuchu, czyli chromosomie.*

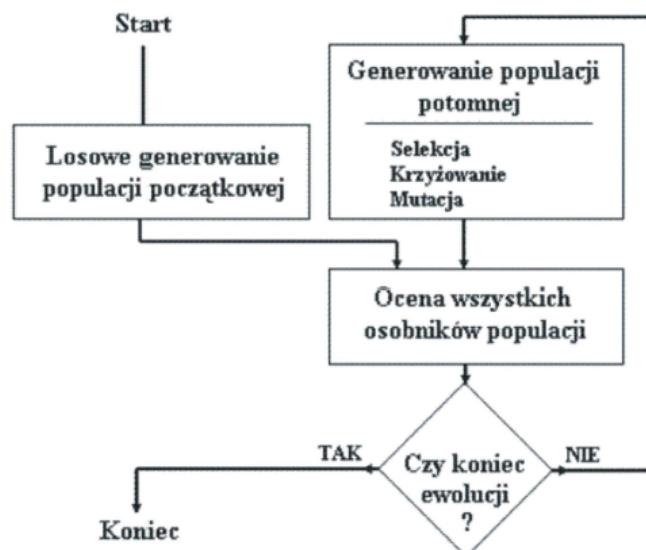
Definicja 8 *Funkcja przystosowania nazywana też funkcją dopasowania lub funkcją oceny. Stanowi ona miarę przystosowania (dopasowania) danego osobnika w populacji.*

2.2.2 Klasyczny algorytm genetyczny

Na podstawowy (klasyczny) algorytm genetyczny, nazywany także elementarnym lub prostym, składają się następujące 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 przedstawionym na rysunku 2.3. Poniżej omówiono wszystkie kroki algorytmu [4].



Rys. 2.3: Ogólny schemat algorytmu genetycznego

Inicjacja, czyli utworzenie populacji początkowej, polega na losowym wyborze żądanej liczby chromosomów (osobników) reprezentowanych przez ciągi binarne o określonej długości. Ta metoda tworzenia populacji początkowej w pewnych sytuacjach nie jest efektywna, gdyż wiele osobników może nie spełniać ograniczeń występujących w rozwiązywanym zadaniu. W takich przypadkach dobrze jest odwołać się do wiedzy o rozwiązywanym problemie i „zaprojektować” zestaw chromosomów.

Ocena przystosowania chromosomów w populacji opiera się na obliczeniu wartości funkcji przystosowania dla każdego chromosomu z tej populacji. Zwykle im większa jest wartość funkcji, tym lepsza „jakość” chromosomów. Postać funkcji przystosowania zależy od rodzaju rozwiązywanego problemu. Zwykle zakłada się, że funkcja przystosowania przyjmuje zawsze wartości nieujemne, a ponadto, że rozwiązywany problem optymalizacji jest problemem poszukiwania maksimum tej funkcji. Oczywiście, pierwotną postać funkcji przystosowania można łatwo sprowadzić do innej postaci.

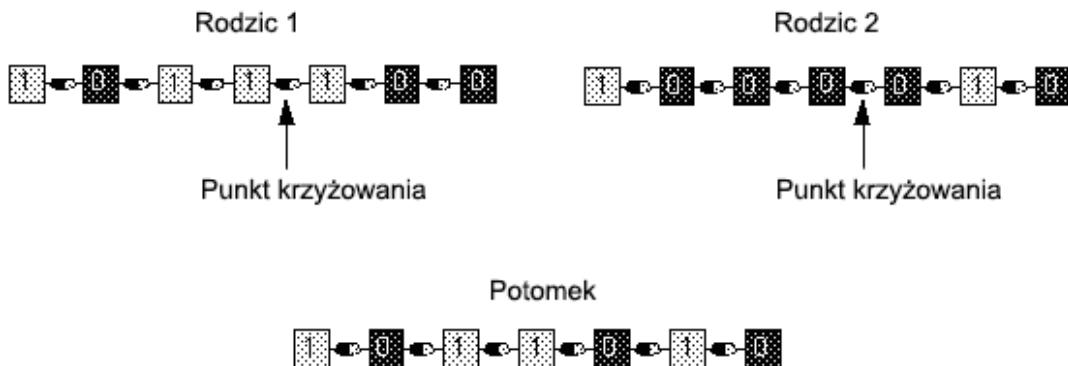
Sprawdzanie warunku zatrzymania. Określenie warunku zatrzymania algorytmu genetycznego zależy od konkretnego zastosowania tego algorytmu. W zagadnieniach optymalizacji, jeśli znana jest wartość oczekiwana funkcji przystosowania, zatrzymanie algorytmu może nastąpić po uzyskaniu tej wartości. Ewentualnie określana jest pewna dokładność porównania. Zatrzymanie algorytmu może również nastąpić, gdy jego dalsze działanie nie poprawia już uzyskanej najlepszej wartości. Zwykle jednak, algorytm jest zatrzymywany po upływie określonego czasu działania lub po określonej liczbie iteracji. Jeśli warunek zatrzymania jest spełniony, następuje przejście do ostatniego kroku, czyli wyprowadzenie „najlepszego” chromosomu.

Selekcja chromosomów polega na wybraniu na podstawie obliczonych wartości funkcji przystosowania, tych chromosomów, które będą brały udział w tworzeniu potomków do następnego pokolenia, czyli następnej generacji. Wybór ten zwykle odbywa się zgodnie z zasadą naturalnej selekcji, to znaczy największe szanse na udział w tworzeniu nowych osobników mają chromosomy o największej wartości funkcji przystosowania. Istnieje wiele metod selekcji populacji.

Zastosowanie operatorów genetycznych. W klasycznym algorytmie genetycznym stosuje się dwa podstawowe operatory genetyczne: operator krzyżowania oraz operator mutacji. Należy jednak zaznaczyć, że operator mutacji odgrywa zdecydowanie drugoplanową rolę w stosunku do operatora krzyżowania. Oznacza to, że krzyżowanie w klasycznym algorytmie genetycznym występuje prawie zawsze, podczas gdy mutacja dość rzadko.

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 utworzenia nowego chromosomu. Podstawowa operacja krzyżowania opiera się na jednym punkcie krzyżowania i została przedstawiona na poglądowym rysunku 2.4 [2]

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 się tego poprzez losową



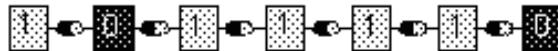
Rys. 2.4: Ogólny schemat operacji krzyżowania osobników

zmianę dowolnego genu w chromosomie. Sytuacja jest przedstawiona na poglądowym rysunku 2.5 [2].

Przed mutacją:



Po mutacji:



Rys. 2.5: Ogólny schemat operacji mutacji osobników

Utworzenie nowej populacji. Chromosomy otrzymane w wyniku działania operatorów genetycznych na chromosomach tymczasowej populacji rodzicielskiej wchodzą w skład nowej populacji. Populacja ta staje się tak zwaną populacją bieżącą dla danej iteracji algorytmu genetycznego. W każdej kolejnej iteracji oblicza się wartość funkcji przystosowania każdego z chromosomów tej populacji. Następnie sprawdza się warunek zatrzymania algorytmu i albo wyprowadza się wynik w postaci chromosomu o największej (najmniejszej) wartości funkcji przystosowania, albo przechodzi się do kolejnego kroku algorytmu genetycznego, to jest selekcji. W klasycznym algorytmie genetycznym cała poprzednia populacja chromosomów zastępowana jest przez tak samo liczną nową populację potomków.

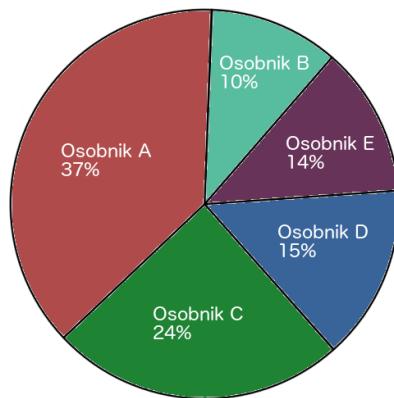
Wyprowadzenie „najlepszego” chromosomu. Jeżeli spełniony jest warunek zatrzymania algorytmu genetycznego, należy wyprowadzić wynik algorytmu genetycznego,

czyli podać rozwiązanie problemu. Najlepszym rozwiązaniem jest chromosom o największej lub najmniejszej wartości funkcji przystosowania.

2.2.3 Metody selekcji osobników

W algorytmach genetycznych wyróżnia się etap selekcji, który z bieżącej populacji osobników wybiera te o największej wartości funkcji przystosowania do populacji rodziniejskiej.

Metoda ruletki (koła ruletki) jest jedną z najbardziej podstawowych metod selekcji. Polega na utworzeniu koła ruletki z polami odpowiadającymi poszczególnym chromosomom. Wielkość pól jest proporcjonalna do wartości funkcji przystosowania. Proces selekcji oparty jest na obrocie ruletką tyle razy, ile osobników jest w populacji i na wyborze za każdym razem jednego chromosomu do nowej populacji. Pewne chromosomy są wybierane więcej niż jeden raz, niektóre dokładnie raz, a niektóre wcale [2]. Jest to metoda prosta i daje w miarę zadowalające wyniki. Ma jednak wady. Jedną z nich jest możliwość stosowania metody ruletki tylko do jednej klasy zadań, tzn. tylko do maksymalizacji (lub tylko do minimalizacji). Takie ograniczenie jest niewątpliwie jej wadą. Bardziej istotną wadą metody ruletki jest zbyt wcześnie eliminowanie z populacji osobników o bardzo małej wartości funkcji przystosowania. Może to prowadzić do zbyt wczenej zbieżności algorytmu. Metoda ta została zobrazowana na rysunku 2.6.



Rys. 2.6: Schemat selekcji osobników metodą koła ruletki

W selekcji rankingowej osobniki populacji są ustawiane kolejno w zależności od funkcji przystosowania to znaczy od najlepiej do najgorszej przystosowanego. Każdemu osobnikowi przyporządkowana jest liczba zwana rangą i oznaczająca jego pozycję na liście. Liczba kopii danego osobnika wprowadzonych do nowej populacji jest ustalana na podstawie wcześniejszej zdefiniowanej funkcji, zależnej od rangi.

Selekcja turniejowa polega na podzieleniu populacji na podgrupy k-elementowe (k to rozmiar turnieju – zwykle 2 lub 3) i wyborze z każdej podgrupy osobnika o najlepszym

przystosowaniu. Można to zrobić poprzez wybór losowy lub wybór deterministyczny. Wtedy wyboru dokonuje się z prawdopodobieństwem równym 1. Taki mechanizm wyboru rodziców posiada szereg zalet. Po pierwsze, w przypadku małych grup o rozmiarze k zapewnia to szanse przetrwania $k - 2$ najlepszym osobnikom, co z kolei gwarantuje, że dostosowanie populacji nie pogarsza się w trakcie kolejnych iteracji. Po drugie, bez względu na to, jak dobrze dostosowany jest osobnik w porównaniu z resztą populacji, produkuje on co najwyżej jednego potomka w każdej iteracji. Własność ta zapobiega przedwczesnej zbieżności populacji – problemowi, z którym boryka się większość modeli ewolucji. W selekcji proporcjonalnej osobnik o wysokim dostosowaniu dominuje początkowe etapy ewolucji, co skutecznie ogranicza obszar poszukiwań i powoduje zbieżność populacji do rozwiązań suboptimalnych. Po trzecie, metoda turniejowa nadaje się zarówno do problemów maksymalizacji, jak i minimalizacji [3].

W klasycznym algorytmie genetycznym możliwa jest sytuacja, w której do nowej populacji nie zostaną wybrane najlepsze osobniki poprzedniej populacji. Chcąc zapobiec takiej sytuacji stosuje się **strategię elitarnej selekcji**. W jej przypadku nacisk położony jest na zachowanie w kolejnych iteracjach najlepiej przystosowanych osobników. Dzięki temu zachowywane jest najlepsze rozwiązanie (lub N najlepszych rozwiązań) pomiędzy iteracjami [2]. Strategia ta powoduje, że potomstwo elity będzie dominować przyszłe populacje. Może to ograniczać zdolności eksploracyjne algorytmu, gdyż początkowa elita niekoniecznie zawiera geny charakteryzujące najlepsze osobniki. Rozwiązaniem jest utrzymywanie elity o niewielkim rozmiarze, podobnie jak ma to miejsce w pojedynczej selekcji turniejowej, gdzie elita składa się z dwóch osobników.

2.2.4 Mocne i słabe strony algorytmów genetycznych

Początkowo algorytmy genetyczne były głównie założeniem teoretycznym. Dziś są szeroko wykorzystywanymi rozwiązaniami do wielu problemów badawczych, jak również są stosowane w przypadku zastosowań komercyjnych. Oczywiście jedną z przyczyn, dla której algorytmy genetyczne zauważają swój sukces jest rosnąca moc obliczeniowa komputerów oraz rozwój Internetu. Są one wykorzystywane w bardzo wielu różnych dziedzinach jak badania rynku akcji i prognozy portfela, planowanie, projektowanie inżynierii kosmicznej i mikroprocesorowej, biochemii i biologii molekularnej oraz planowania tras w portach lotniczych i liniach montażowych. Potencjał algorytmów ewolucyjnych nie jest z pewnością jeszcze w pełni wykorzystany.

To wszystko jest możliwe dzięki wielu mocnym stronom algorytmów genetycznych:

- Mogą rozwiązać każdy problem optymalizacji, który może być opisany za pomocą kodowania chromosomów.

2.3. TEORIA GRAFÓW

- Rozwiązuje problemy posiadające wiele rozwiązań.
- Budowa algorytmów genetycznych nie ma związku z rozwiązywanym problemem, dzięki czemu mogą rozwiązywać problemy wielowymiarowe, nieróżniczkowalne, nieciągłe, a nawet nieparametryczne.
- Są bardzo łatwe do zrozumienia i praktycznie nie wymagają znajomości matematyki.
- Są łatwo adoptowane do istniejących symulacji i modeli.

Oczywiście, istnieją też wady tych rozwiązań:

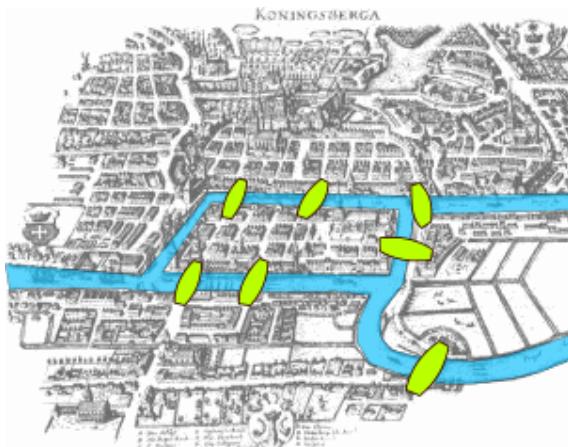
- Pewne problemy optymalizacji nie mogą być rozwiązane za pomocą algorytmów genetycznych. Dzieje się tak, gdy nie jest możliwe określenie prawidłowej funkcji przystosowania chromosomu, co zaburza system ewolucji najlepszych rozwiązań.
- Nie ma pewności, że algorytm genetyczny znajdzie optymalne rozwiązanie.
- Podobnie, jak w przypadku innych metod sztucznej inteligencji, nie jest możliwe przewidzenie czasu trwania optymalizacji przy pomocy algorytmu genetycznego.
- Algorytmy genetyczne czasu rzeczywistego są zawodne z powodu przypadkowych rozwiązań i konwergencji. Oznacza to, że podczas gdy populacja zawsze zmierza do lepszego wyniku, nie dotyczy to pojedynczego osobnika.

2.3 Teoria grafów

Wiele sytuacji z życia codziennego może zostać wygodnie opisanych za pomocą diagramów składających się z zestawu punktów oraz linii łączących określone pary tych punktów. Punkty mogą reprezentować ludzi a połączenia znajomości lub może to być odzwierciedlenie ośrodków komunikacyjnych oraz ich połączeń. Warto zauważyć, że w dziedzinie tych problemów nie jest interesującym sposób połączenia ani typ punktów, a jedynie abstrakcyjnie matematyczny koncept takiej sytuacji [12].

2.3.1 Geneza teorii grafów

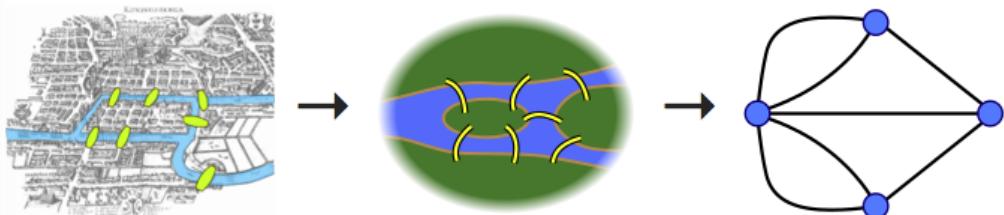
Zagadnienie mostów królewieckich jest historycznym problemem matematycznym. Rozwiązanie zaproponowane w 1735 roku przez szwajcarskiego matematyka i fizyka Leonharda Eulera dało początek teorii grafów. Królewiec leżący w Królestwie Prus (dzisiaj Kaliningrad, Rosja), znajduje się nad brzegiem rzeki Pregoła. Miasto posiada dwie duże



Rys. 2.7: Mapa królewieckich mostów z czasów Eulera

wyspy, które połączone są ze sobą siedmioma mostami. Historyczna mapa przedstawiona na rysunku 2.7 świetnie ilustruje tę sytuację [11].

Problemem postawionym matematykowi było znalezienie takiej drogi, która przekroczy każdy most raz i tylko raz. Wyspa nie była dostępna ze strony miasta w żaden inny sposób niż przez most. Dodatkowo przekroczenie każdego mostu musiało odbyć się w pełni, to znaczy niedozwolonym było zawracanie na środku mostu. Pierwszą obserwacją Eulera był fakt, że dokładna pozycja w mieście nie jest istotna. To pozwoliło mu na uproszczenie problemu do abstrakcyjnej formy, dając początki teorii grafów (Rys. 2.8).



Rys. 2.8: Schemat przekształcenia problemu mostów królewieckich w graf

Matematyk dowódł, że nie jest możliwe rozwiązanie tego problemu. W efekcie stworzył jednak definicję, która jasno określa warunki, jakie muszą być spełnione, by taka sytuacja mogła mieć miejsce. Rozważył on też podobny problem, w którym zakończenie drogi jest tym samym punktem, co jej początek. Obie definicje do dziś nazywa się nazwiskiem odkrywcy.

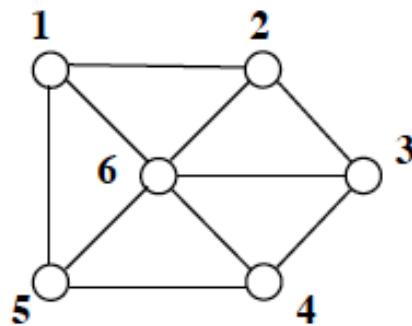
Definicja 9 *Ścieżka Eulera* (ang. Eulerian path) jest ścieżką w grafie, która przechodzi dokładnie jeden raz przez każdą jego krawędź. Aby taka ścieżka istniała, graf musi być spójny oraz każdy jego wierzchołek za wyjątkiem dwóch musi posiadać parzysty stopień.

Definicja 10 *Cykł Eulera* (ang. *Eulerian cycle*) jest ścieżką w grafie, która przechodzi przez wszystkie jego krawędzie i kończy się w wierzchołku startowym. Aby istniał cykl Eulera, graf musi być spójny oraz każdy jego wierzchołek musi posiadać stopień parzysty.

Opis zagadnienia opublikowany przez Eulera w pracy *Solutio problematis ad geometriam situs pertinentis* w *Commentarii academiae scientiarum Petropolitanae* jest uznawany za pierwszą pracę na temat teorii grafów.

2.3.2 Podstawowe pojęcia dotyczące grafów

Definicja 11 *Grafem (nieskierowanym)* nazywa się parę zbiorów (V, E) . Elementy zbioru V nazywają się wierzchołkami, natomiast elementy zbioru E to krawędzie. Każda krawędź jest parą wierzchołków, tzn. $E \subseteq u, v : u, v \in V$ [10].



Rys. 2.9: Przykładowy graf nieskierowany

Przykładowy graf nieskierowany (Rys. 2.9) może zostać opisany przez zbiory:

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

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

Definicja 12 *Rząd grafu* (ang. *graph order*) to liczba wierzchołków w grafie.

Definicja 13 *Rozmiar grafu* (ang. *graph size*) to liczba krawędzi w grafie.

Wierzchołki grafu przechowują informację, natomiast krawędzie określają sposób poruszania się po grafie: z wierzchołka u można przejść do wierzchołka v tylko wtedy, gdy istnieje ścieżka (ciąg krawędzi), która prowadzi w grafie od wierzchołka u do v .

Definicja 14 *Grafem zerowym* (ang. *null graph*) jest graf, który posiada wierzchołki, lecz nie posiada żadnych krawędzi.

Definicja 15 Wierzchołek niepołączony krawędzią z żadnym innym wierzchołkiem grafu nazywa się *wierzchołkiem izolowanym* (ang. *isolated vertex*).

Definicja 16 Dane dwa wierzchołki mogą być połączone ze sobą za pomocą więcej niż jednej krawędzi, którą nazywa się **krawędzią wielokrotną** (ang. *multi-edge*).

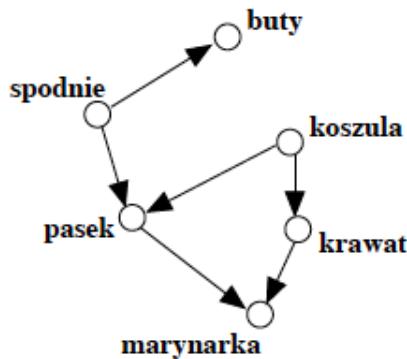
Definicja 17 Wierzchołek może łączyć się krawędzią z samym sobą. Otrzymuje się wtedy tzw. **pętlę** (ang. *loop*).

Definicja 18 Graf zawierający pętle lub krawędzie wielokrotne nazywa się **multigrafem** (ang. *multigraph*).

Definicja 19 Graf nieposiadający pętli oraz krawędzi podwójnych nazywa się **grafem prostym** (ang. *simple graph* lub *strict graph*).

Definicja 20 **Stopniem wierzchołka** (ang. *degree*) nazywa się liczbę krawędzi, które łączą się z danym wierzchołkiem. Jeśli wierzchołek posiada pętle, to każdą z nich liczy się jako dwie krawędzie.

Definicja 21 **Grafem skierowanym** nazywa się taki graf, w którym każda krawędź ma zdefiniowany początek i koniec, tzn. pary grafu muszą być 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 oznacza się ją przez $u \rightarrow v$ [10].



Rys. 2.10: Przykładowy graf skierowany

Przykładowy graf skierowany (Rys. 2.10) może zostać opisany przez zbiory:

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

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

Definicja 22 **Ścieżka lub droga** (ang. *path*) jest uporządkowanym ciągiem kolejnych krawędzi, po których należy przejść, aby dotrzeć z wierzchołka startowego (ang. *start vertex*) do wierzchołka końcowego (ang. *end vertex*). W grafie może istnieć wiele różnych ścieżek pomiędzy dwoma wybranymi wierzchołkami.

Definicja 23 *Najkrótsza ścieżka (ang. shortest path)* to ta, która zawiera najmniej krawędzi/wierzchołków. *Długość ścieżki (ang. path length)* to liczba zawartych w niej krawędzi/wierzchołków.

Definicja 24 *Mówiąc, że ścieżka jest prosta (ang. straight path lub simple path), jeśli każdą krawędź/wierzchołek przechodzi się tylko jeden raz.*

Definicja 25 *Cykl (ang. cycle)* to ścieżka, która rozpoczyna się i kończy w tym samym wierzchołku. *Uwaga, nie mylić cyklu z pętlą – pętla to pojedyncza krawędź.*

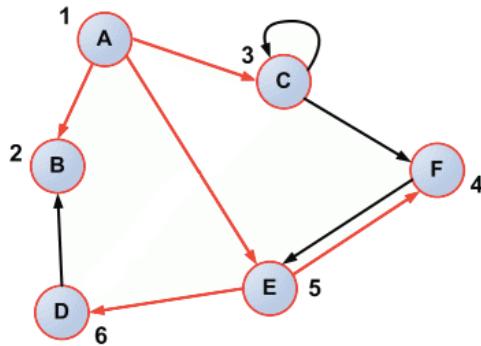
Definicja 26 *Graf nazywa się acyklicznym (ang. acyclic graph), jeśli nie posiada żadnych cykli.*

2.3.3 Przeszukiwanie grafów

Przejście grafu (*ang. graph traversal*) polega na przechodzeniu przez wierzchołki, do których prowadzą ścieżki. Algorytm przejścia daje nam pewność, że żaden taki wierzchołek nie zostanie pominięty.

Przechodzenie grafów w głąb – DFS

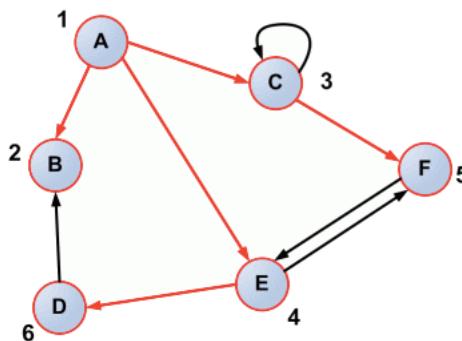
Przeszukiwanie w głąb (*ang. Depth-First Search, DFS*) jest algorytmem o prostej do zrozumienia idei. Na początku wszystkie wierzchołki grafu oznacza się jako nieodwiedzone, a w momencie dojścia do danego węzła grafu staje się on odwiedzony. Otóż rozpoczęjąc przechodzenie od wybranego, startowego wierzchołka grafu, algorytm porusza się „w głąb” grafu poprzez wybieranie kolejnych krawędzi, składających się na możliwie najprostszą ścieżkę aż do momentu, gdy nie można przejść do kolejnego, nieodwiedzonego jeszcze wierzchołka. Wówczas należy wrócić do węzła ostatnio odwiedzonego i sprawdzić kolejną drogę (za pomocą krawędzi wychodzących z niego do nieodwiedzonych węzłów). Czynność tę powtarza się rekurencyjnie dopóty, dopóki wszystkie wierzchołki, do których istnieje ścieżka z węzła startowego, nie zostaną odwiedzone. W przypadku, gdy w grafie będą jeszcze nieodwiedzone wierzchołki, wówczas obiera się jeden z nich jako nowy węzeł startowy i ponownie rozpoczyna się przeszukiwanie. Metoda jest kontynuowana do momentu odwiedzenia wszystkich wierzchołków grafu. Metoda DFS jest zatem dobrym przykładem algorytmu z powrotami. Przykładowe przejście grafu z wykorzystaniem powyższej metody zostało zobrazowane na obrazku 2.11. Numery przy węzłach odpowiadają kolejności, w której zostały odwiedzone.



Rys. 2.11: Przykładowy graf z oznaczeniem kolejności odwiedzonych węzłów przy przeszukiwaniu metodą DFS

Przechodzenie grafów wszerz – BFS

Przeszukiwanie wszerz (*ang. Breadth-First Search, BFS*) jest drugim z najbardziej znanych sposobów przeszukiwania grafu. Zasada jest odmienna niż w przypadku metody DFS – w pierwszej kolejności przeglądany jest węzeł początkowy (źródłowy), następnie odwiedzane są po kolejnych węzły sąsiadujące ze źródłem, później wierzchołki sąsiadnie (te jeszcze nieodwiedzone) sąsiadów źródła itd. A zatem, jeśli wykorzystuje się listy sąsiedztwa do reprezentacji grafu, to najpierw przegląda się wierzchołki z listy sąsiedztwa węzła początkowego, a potem wierzchołki z list sąsiedztwa węzłów, które znajdują się na liście przynależnej do źródła. W skrócie można powiedzieć, że w pierwszej kolejności algorytm dociera do węzłów oddalonych od wierzchołka początkowego o jedną krawędź, potem o dwie krawędzie itd. – stąd właśnie nazwa algorytmu, gdyż porusza się „wszerz” na każdym kolejnym poziomie. Przykładowe przejście grafu z wykorzystaniem powyższej metody zostało zobrazowane na obrazku 2.12. Numery przy węzłach odpowiadają kolejności, w której zostały odwiedzone.

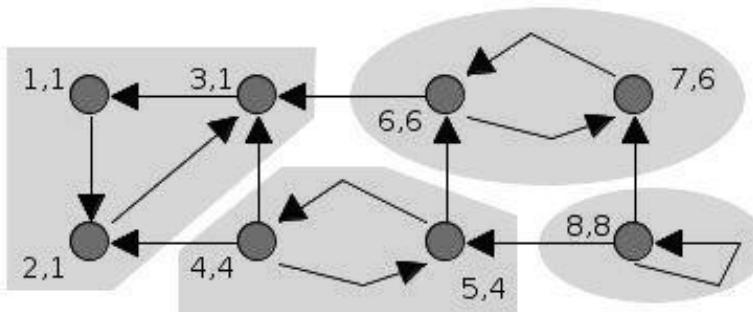


Rys. 2.12: Przykładowy graf z oznaczeniem kolejności odwiedzonych węzłów przy przeszukiwaniu metodą BFS

2.3.4 Graf skierowany silnie spójny

Definicja 27 *Grafem skierowanym silnie spójnym nazywa się graf skierowany, w którym możliwe jest 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 jedną wychodzącą [21].*

Definicja 28 *Składowa silnie spójna (ang. strongly connected component) jest maksymalnym podgrafem, w którym istnieją ścieżki pomiędzy każdym dwoma wierzchołkami. Jeśli podgraf ten obejmuje wszystkie wierzchołki grafu, to mówi się, że dany graf skierowany jest silnie spójny (ang. strongly connected digraph). W grafach nieskierowanych każdy graf spójny jest również silnie spójny. Poglądowo sytuacja ta jest przedstawiona na rysunku 2.13.*



Rys. 2.13: Przykładowy graf z zaznaczonymi składowymi silnie spójnymi

Algorytm Kosaraju

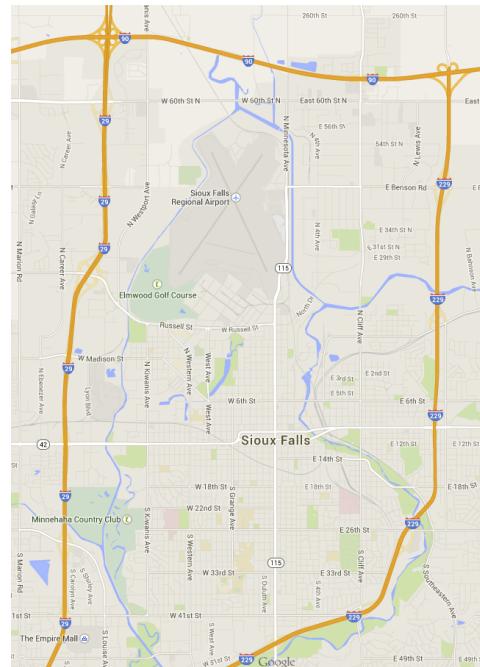
Najprostszą metodą poszukiwania składowych silnie spójnych w grafie jest **algorytm Kosaraju**. Metoda ta korzysta z reguły, że jeżeli wszystkie krawędzie grafu zostałyby odwrócone, składowe silnie spójne takiego grafu pozostają takie same, jak w przypadku grafu wejściowego. Wykorzystując tę wiedzę, algorytm wykonuje przejście w głąb by znaleźć kolejność występowania wierzchołków. Następnie dokonywane jest ponowne przejście w głąb, ale w odwrotnej kolejności, bazując na porządku ustalonym podczas poprzedniego przejścia. Wykorzystując wcześniej wspomnianą regułę, tworząc „odwrócony” graf, zaczynając od ostatniego wierzchołka pierwszego przejścia, wszystkie wierzchołki z którymi jest on połączony tworzą składową silnie spójną. Ostatni krok powtarza się do momentu, gdy wszystkie wierzchołki zostaną sprawdzone.

Algorytm Tarjana

Algorytm ten w porównaniu do poprzednika jest dwukrotnie szybszy, ponieważ wykorzystuje jedynie jedno przejście w głąb. Podstawowym założeniem **algorytmu Tarjana** [17] 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 tworzy się 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.

2.4 Sieć drogowa w postaci grafu

W przypadku sieci drogowej mowa oczywiście o abstrakcyjnej strukturze sieci [15]. Przez sieć drogową rozumie się bowiem układ dróg lub ulic na przykład w mieście. W ramach niniejszej pracy wykorzystano pewien fragment większej sieci. Było to miasto Sioux Falls w Południowej Dakocie, prezentowane na rysunku 2.14.

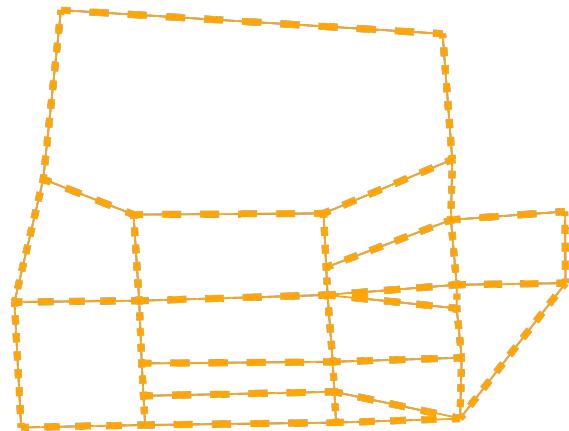


Rys. 2.14: Fragment sieci drogowej miasta Sioux Falls, Południowa Dakota

Posługując się powyższą definicją grafu skierowanego, tworząc graf z pewnej sieci drogowej, przyjmuje się, że zbiorem V wierzchołków są skrzyżowania, natomiast zbiór krawędzi E odnosi się do ulic pomiędzy tymi skrzyżowaniami. W pracy posługiwano

2.4. SIEĆ DROGOWA W POSTACI GRAFU

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. Efekt transformacji fragmentu sieci drogowej zaprezentowanej w postaci grafu skierowanego pokazano na rysunku 2.15.



Rys. 2.15: Sieć drogowa miasta Sioux Falls w postaci grafu

Na pierwszy rzut oka nie jest widoczne podobieństwo pomiędzy mapą a jej odzwierciedleniem w postaci grafu. Wynika to z braku dopasowania geometrycznego po przekształceniu. Na rysunku 2.16 prezentowana jest sieć drogowa miasta w postaci grafu, po przekształceniu geometrycznym, tak by odzwierciedlała on mniej więcej ulice widoczne na mapie.



Rys. 2.16: Graf sieci drogowej miasta z dopasowaną geometrią

2.5 Przystosowanie struktury grafu do użycia w algorytmie genetycznym

Niniejsza praca skupia się na optymalizacji grafów. W tym wypadku klasyczna odmiana algorytmu genetycznego musiałaby zostać zmodyfikowana na potrzeby przedstawienia grafów jako chromosomów. Dodatkowo, wymagałoby to zastosowania nowych sposobów krzyżowania i mutacji jednostek.

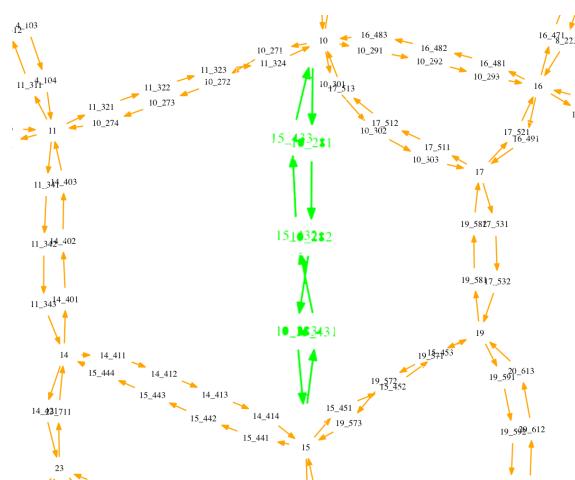
Zdecydowano, że zamiast przystosowywać algorytm genetyczny do nowej struktury, struktura zostanie przystosowana do algorytmu. Ponieważ postawionym zadaniem jest zdecydowanie o zamknięciu lub nie danej ulicy (krawędzi grafu), postanowiono przedstawić tę strukturę jako listę opisującą taki stan. Zgodnie z tą myślą, sieć drogowa (graf) jest tłumaczona na listę elementów przyjmujących wartości:

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

Tworzona w ten sposób lista stanów jest tak naprawdę tablicą zawierającą elementy o wartościach prawda lub fałsz (1 lub 0). Bez problemu może być zatem modyfikowana przez klasyczny algorytm genetyczny. Bardzo łatwym jest również odtworzenie obecnego stanu sieci drogowej. Przykładową sytuację przedstawiają rysunki 2.17 i 2.18.

1 | 1 | 0 | 0 | 0 | 0 | 0

Rys. 2.17: Fragment sieci drogowej w postaci tablicy binarnej



Rys. 2.18: Fragment sieci drogowej w postaci grafu

2.5. PRZYSTOSOWANIE STRUKTURY GRAFU DO UŻYCIA W ALGORYTMIE GENETYCZNYM

Przedstawiony na rysunku 2.17 fragment sieci zawiera kolejne elementy zerowe, które w tłumaczeniu na przykładowy graf są zaznaczone kolorem zielonym (Rys. 2.18). W tym przypadku krawędzie te zostaną wyłączone z ruchu.

W powyższym rozdziale zostały przybliżone matematyczne prawa i twierdzenia użyte podczas realizacji pracy dyplomowej. Wyjaśniony został paradoks Braessa, który jest kluczowym założeniem pracy. Zaprezentowano struktury i prawa użyte w części praktycznej projektu *GRONO*. W kolejnym rozdziale znajduje się opis wybranego systemu symulacji wraz z przedstawieniem miasta użytego w badaniach.

Rozdział 3

Metoda oceny struktury sieci drogowej

W niniejszym rozdziale opisana została metoda oceny sieci drogowej, modyfikowanej w pracy. Objascono dokładnie wykorzystany w tym celu system symulacji transportu ruchu drogowego wraz z wykorzystanymi ustawieniami. Następnie przedstawiono badane przykładowe miasto, które zostało wykorzystane podczas badań wykonanych w części praktycznej niniejszej pracy dyplomowej.

3.1 Symulator transportu ruchu drogowego

MATSim jest środowiskiem do implementacji szerokiej skali symulacji transportu, opartej 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 [22]. Obecnie *MATSim* umożliwia modelowanie ruchu drogowego na podstawie symulacji transportu opartej na agentach. Odbywa się ono poprzez dostarczony kontroler, który w sposób iteracyjny uruchamia symulacje oraz analizuje wyniki uzyskane poprzez wszystkie moduły systemu symulacji.

Możliwe jest wykorzystanie podstawowego zestawu funkcji *MATSim* bez ingerencji w środowisko. Odbywa się ono poprzez dostarczenie swoich danych wejściowych oraz dostosowanie konfiguracji symulacji. Dla bardziej zaawansowanych zastosowań może się jednak okazać konieczne dostarczenie własnych implementacji.

Głównymi cechami, którymi szczyci się symulator *MATSim* są:

- **Symulacja zachowań mobilnych oparta na agentach.** Pokrycie odwzorowanych aktywności dotyczy jednego pełnego dnia, podczas którego (zwykle) duża liczba jednostek, agentów symulacji, jednocześnie bierze udział w symulowanym scenariuszu. Możliwe jest odtworzenie zachowań zarówno prywatnego samochodu, jak i transportu publicznego w bardzo dużej szczególowości. Pozwala to na śledzenie

3.1. SYMULATOR TRANSPORTU RUCHU DROGOWEGO

każdego uczestnika symulacji (agenta) osobno, poprzez cały ich dzień - od wyjścia z domu do pracy, poprzez wszystkie jego poboczne działania, aż do momentu powrotu.

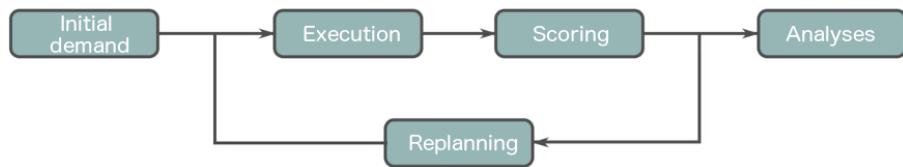
- **Szybkość i stabilność działania.** *MATSim* został przystosowany do obsługi nawet bardzo dużych scenariuszy symulacji transportu składających się z milionów biorących w niej udział agentów oraz setek tysięcy odcinków dróg.
- **Analiza dostarczanych wyników.** Podczas symulacji transportu przechowywane są kluczowe wartości pozwalające szybko ocenić obecny stan tego procesu. Możliwe jest porównywanie wartości uzyskanych przez symulator z rzeczywistymi danymi. Oprócz tego, *MATSim* dostarcza szczegółowy opis przeprowadzonej symulacji, który może zostać łatwo przetworzony przez zewnętrzne aplikacje w celu stworzenia autorskiej, szczegółowej analizy.
- **Podejście modułowe.** Pozwala na łatwą wymianę lub dodawanie potrzebnych funkcjonalności. Umożliwia zastosowanie własnych algorytmów zachowań agentów symulacji lub stworzenie nowatorskiego silnika symulacji przy wykorzystaniu istniejących rozwiązań *MATSim*, dotyczących planowania tras.
- **Stworzenie rozwiązanie w ramach licencji otwartego oprogramowania.** Platforma jest rozpowszechniana na licencji *GNU Public License (GPL)*. Oznacza to, że korzystanie z *MATSim* jest zupełnie darmowe. Ponadto, twórcy dostarczają kompletny kod źródłowy, który można modyfikować. Projekt *MATSim* został stworzony w oparciu o język programowania *Java*, co pozwala na jego wykorzystanie we wszystkich głównych systemach operacyjnych, w tym *Linux*, *Windows* i *Mac OS X*.
- **Aktywny rozwój.** Naukowcy z wielu lokalizacji wciąż pracują nad rozwojem *MATSim*. Główne jednostki rozwoju platformy to: Uniwersytet Techniczny w Berlinie, Politechnika Federalna w Zurychu oraz prywatna firma Senozon.

Oczywiście *MATSim* nie jest jedynym dostępnym symulatorem transportu. Podobnych rozwiązań jest wiele. Wybór motywowany był jednak przede wszystkim dostępnością materiałów szkoleniowych i przykładowymi scenariuszami symulacji ruchu drogowego. *MATSim* jest żywym projektem, opartym o licencję otwartego oprogramowania (*ang. open source*). Dysponuje szerokim zasobem przykładów i gotowych scenariuszy.

3.1.1 Symulacja ruchu drogowego

Podczas podstawowej symulacji *MATSim*, potrzeby przemieszczania się agentów są symulowane w zadanym środowisku, na przykład sieci drogowej. Przebieg symulacji można podzielić na pięć etapów [14] (Rys. 3.1):

- *initial demand* — przygotowywanie zapotrzebowania,
- *execution* — uruchomienie symulacji,
- *scoring* — obliczanie wyników agentów,
- *replanning* — ponowne planowanie,
- *analysis* — analiza wyników.



Rys. 3.1: Schemat etapów symulacji *MATSim*

Etap *initial demand* zajmuje się przygotowaniem planów dnia agentów. Podczas tego stadium, wczytany zostaje pełny zestaw agentów wraz z ich planami dnia, co najmniej jednym dla każdego agenta. Taki plan składa się z listy aktywności (na przykład bycie w domu, praca), podróży (składa się na to również wybór środka transportu) oraz informacji pobocznych (jak czas wyjścia z danego miejsca) i dokładnych rozkładów tras. Plan opisuje *intencje* agenta. Jeżeli agent przyjmie zbyt optymistyczne założenia dotyczące na przykład przejazdu do pracy i utknie w „korku” lub spóźni się na autobus, możliwe jest, że plan nie zostanie zrealizowany zgodnie z wcześniejszymi założeniami.

Etap *execution* jest również nazywany *mobility simulation — symulacją transportu*. W tej fazie plany agentów zostają uruchomione wraz z reprezentacją świata fizycznego. Oznacza to przemieszczanie agentów wraz z ich pojazdami po sieci drogowej (w rzeczywistości infrastrukturze miasta). Podczas tego etapu plany agentów oddziałują na siebie oraz wpływają na symulowaną rzeczywistość. Jeżeli zbyt wielu agentów wybierze jedną drogę w tym samym czasie, tworzą się „korki” na symulowanych drogach. Dlatego agenci planują swoje *intencje* na dany dzień, jednak nie są w stanie zaplanować dokładnie przebiegu swojego dnia.

Etap *scoring* następuje po zakończeniu symulacji i jego głównym celem jest ocena agentów na podstawie przebiegu ich akcji w ciągu dnia. Funkcja ocenająca jest w pełni

konfigurowalna, jednak główną zasadą jest zwiększanie wyniku agenta za czas spędzony podczas jego aktywności oraz pomniejszanie go za czas spędzony w podróży. Zatem najwyższe wyniki uzyskają agenci omijający korki i wykorzystujący maksymalnie czas na zaplanowane akcje.

Etap *replanning* jest najważniejszą fazą symulacji. W tym punkcie agenci „wyciągają wnioski” ze swojego dnia i starają się nie popełnić tych samych błędów, wciąż dając do lepszego wyniku. Typowym przykładem takich zmian może być zmiana czasów zakończenia swoich planowanych akcji, zmiana środka transportu lub wybranej drogi. Modyfikacje te są dokonywane przez moduły strategii (*ang. Strategy Modules*).

Etap *analysis* jest ostatnią fazą symulacji i wykorzystywany jest dla przedstawienia wyników wydajnościowych uzyskanych podczas jej trwania. Mogą to być na przykład takie dane, jak typy wybieranego transportu, przebyte kilometry, średni dystans i czas, zależnie od godziny i trybu podróży.

W skrócie, dla zadanej liczby uruchomień symulacji powtarzany jest scenariusz tego samego dnia, dla tego samego zestawu agentów. Ci ostatni, jak zostało wspomniane, po każdym uruchomieniu są oceniani i na podstawie poprzednio uzyskanych wyników, w kolejnej iteracji próbują swoje wyniki polepszyć. Jest to oczywiście sytuacja nirealna. Agenci „przeżywają” bowiem w kółko ten sam dzień, wykonując te same czynności oraz próbując jak najlepiej dopasować swój plan transportu do panującej sytuacji na drodze.

3.1.2 Symulacja transportu oparta na agentach

Przy pomocy *MATSim* możliwa jest symulacja ruchu dla codziennych zachowań komunikacyjnych. W tym celu wykorzystywani są agenci, reprezentujący pojedyncze jednostki ludzkie. Każdy agent posiada własny plan dnia, pracę i cele dodatkowe, które realizuje według swojego uznania. Symulacja zwykle pokrywa cały dzień czynności wykonywanych przez wszystkich agentów jednocześnie. Pozwala to na śledzenie pojedynczych jednostek od momentu wyjścia z domu do pracy, robienie zakupów i wieczornego powrotu. Celem każdego agenta jest uzyskanie jak najwyższego „wyniku” w ciągu dnia. Agent zdobywa punkty poprzez realizowanie swoich planów (na przykład: praca, zakupy), natomiast traci je podczas komunikacji lub za spóźnienia.

Agenci potrafią modyfikować swoje początkowe plany, tak by nowe ich warianty pozwoliły im na zdobycie wyższego wyniku. Główne założenie procesu optymalizacji wywodzi się z algorytmów (co-)ewolucyjnych. Częstka „co” została dodana ze względu na oddzielność algorytmu genetycznego dla każdego agenta z osobna, jednak wyniki uzyskanych planów dnia są zależne od całego systemu agentów.

Poprzez plany dnia, agenci bezpośrednio wymuszają zapotrzebowanie na transport. Pełen zestaw agentów w danym symulowanym scenariuszu ruchu drogowego nazwano

populacją scenariusza. Populacja zawiera dane w postaci hierarchicznej struktury (pliku XML), którego przykład został zaprezentowany na listingu 3.1.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE population SYSTEM "http://www.matsim.org/files/dtd/
    population_v5.dtd">
<population>
    <person id="1">
        <plan selected="yes" score="93.2987721">
            <act type="home" link="1" end_time="07:16:23" />
            <leg mode="car">
                <route type="links">1 2 3</route>
            </leg>
            <act type="work" link="3" end_time="17:38:34" />
            <leg mode="car">
                <route type="links">3 1</route>
            </leg>
            <act type="home" link="1" />
        </plan>
    </person>
    <person id="2">
        <plan selected="yes" score="144.39002">
            ...
        </plan>
    </person>
</population>
```

Listing 3.1: Przykładowy plik populacji scenariusza zawierający plany dnia agentów

Struktura tego pliku może zostać opisana w następujący sposób:

- plik zawiera listę agentów symulowanych w danym scenariuszu,
- każdy agent posiada listę swoich planów dnia, które będzie realizować,
- każdy plan składa się z listy typów aktywności i tras.

Podczas każdej iteracji symulacji, dla każdego agenta zostaje wybrany jeden plan. W przypadku posiadania więcej niż jednego planu przez agenta, w procesie optymalizacji może zostać wybrany inny plan w kolejnej iteracji symulacji. Lista typów aktywności i tras opisuje dokładnie planowane i wykonywane przez agenta akcje. Każda czynność ma przypisane swoje miejsce wykonywania, którym może być dany węzeł sieci (ulica) lub pozycja na mapie znajdująca się pod wskazanymi koordynatami (budynek). Trasa opisuje dokładny plan agenta, w jaki sposób zamierza dotrzeć z jednego miejsca do drugiego. Opis ten zawiera sposób komunikacji: auto, autobus, rower lub droga pokonywana pieszo.

Na szczęście, do poprawnego wykonania symulacji wystarczające jest podanie jedynie niektórych parametrów:

- Każdy agent musi posiadać co najmniej jeden plan.
- Plan nie musi być domyślnie wybrany ani nie musi posiadać obliczonego wyniku.
- Miejsce odbywania się aktywności wystarczy, by było opisane jedynie przez koordynaty, w którym się znajduje.
- Planowane aktywności muszą mieć oznaczony czas zakończenia.
- Trasa musi posiadać jedynie zdefiniowany sposób transportu, bez obliczonej dokładnej trasy dotarcia do celu.

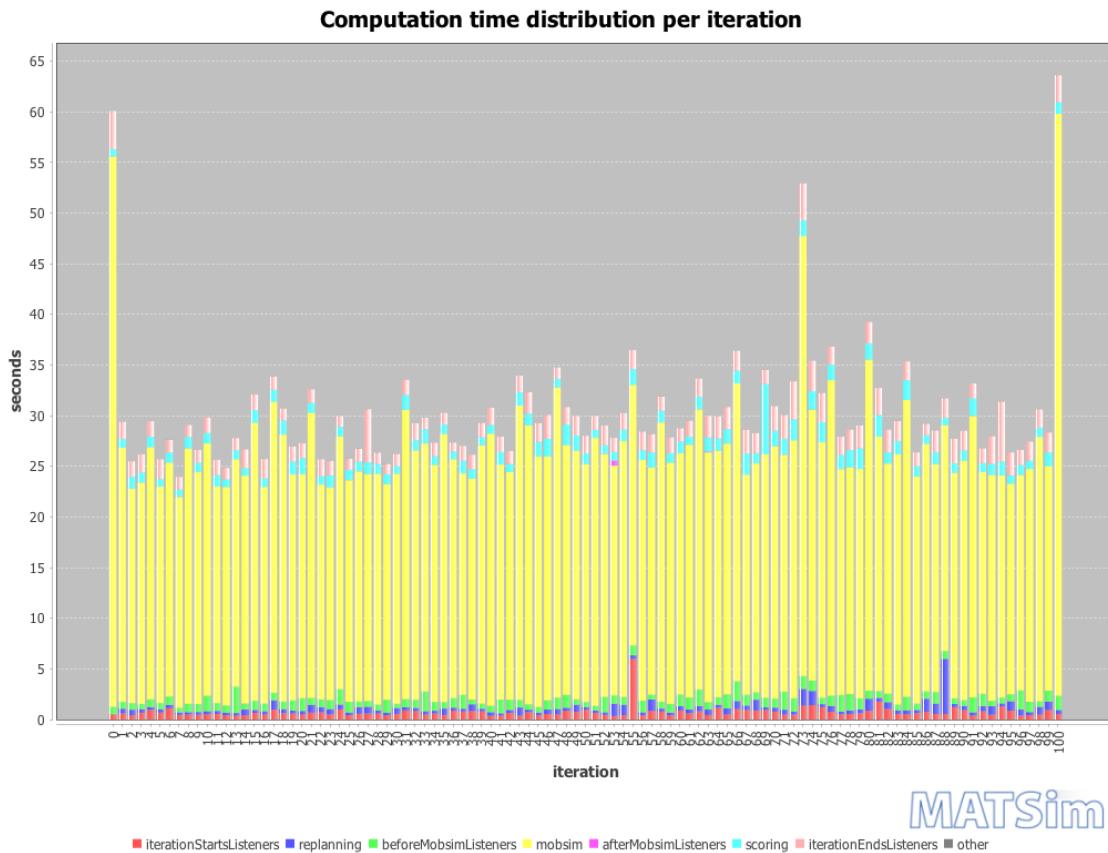
3.2 Ustawienia symulatora transportu

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, przybliżonego w podrozdziale 3.3. Symulator zawiera jednak ważny parametr, a mianowicie 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. Krok ten został już opisany w sekcji 3.1.2. 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 3.2 przedstawiony został graf czasu uruchomienia symulacji jednej sieci dla stu iteracji. Podczas symulacji wykorzystywana była domyślna sieć miasta Sioux Falls.

Schemat etapów symulacji został przedstawiony w sekcji 3.1.1. Nazwy procesów z wykresu w odniesieniu do wcześniej objaśnionych etapów symulacji to:

1. *iterationStartListeners* — *initial demand*,
2. *beforeMobSimListeners*, *mobSim*, *afterMobSimListeners* — *execution*,
3. *scoring* — *scoring*,
4. *replanning* — *replanning*,
5. *iterationEndsListeners* — *analyses*.

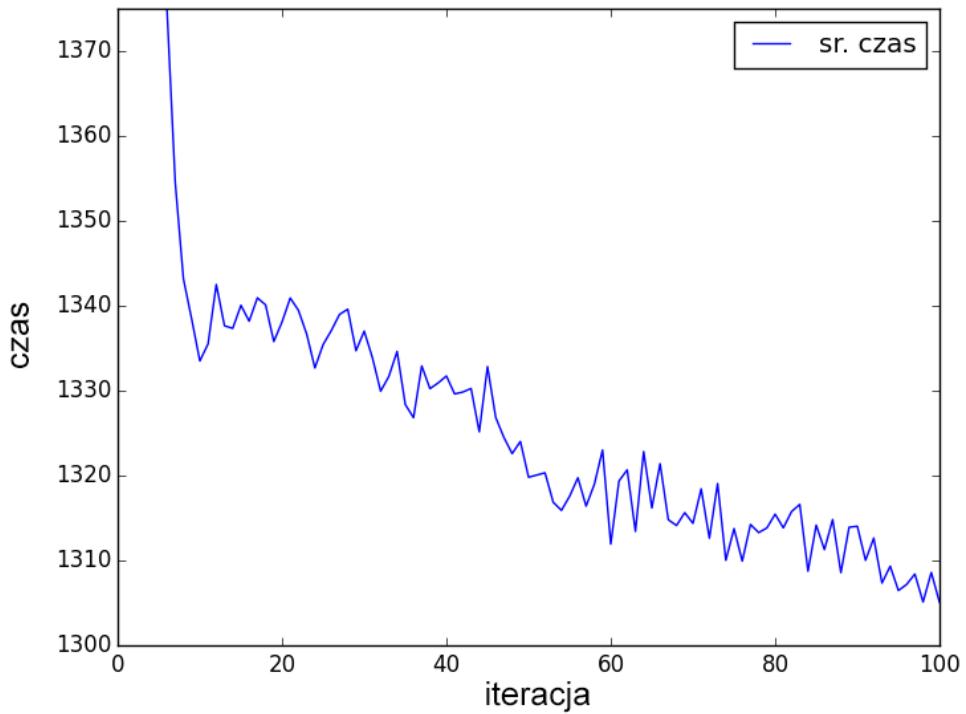


Rys. 3.2: Graf czasu trwania symulacji *MATSim* dla stu iteracji

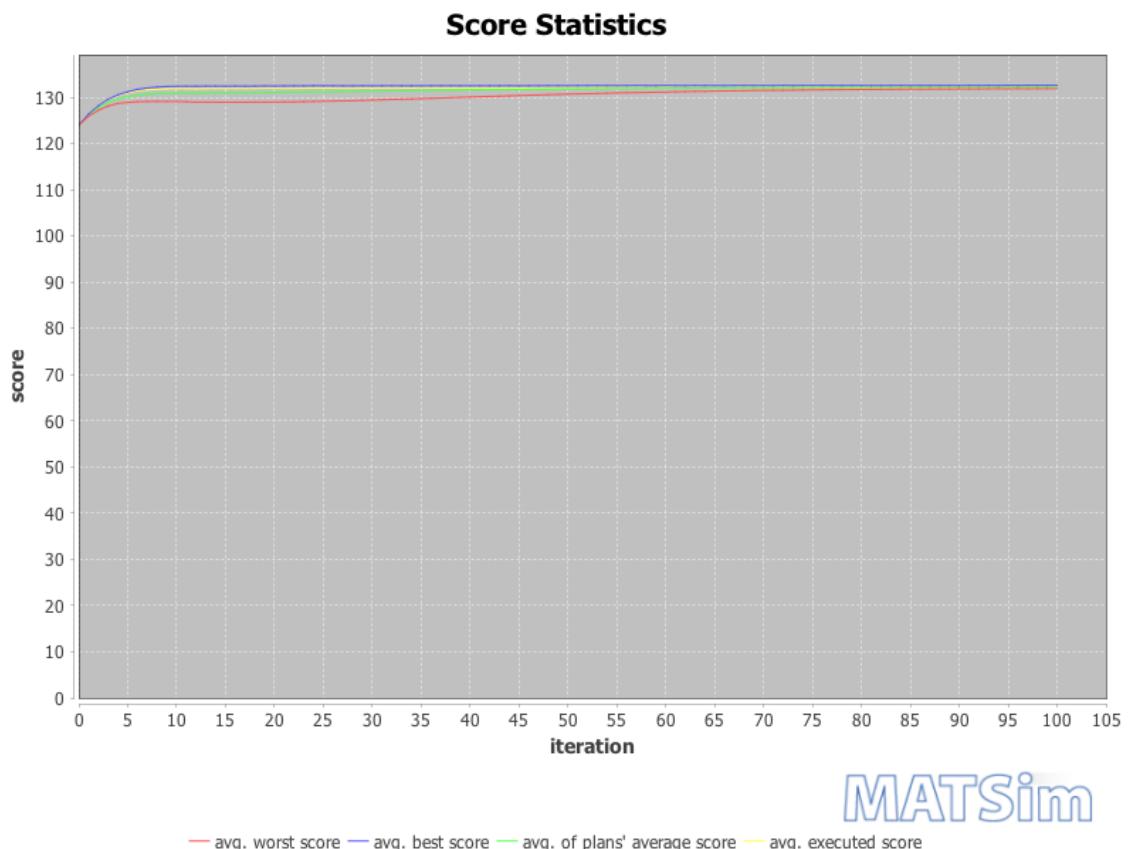
Analizując wykres czasu trwania kolejnych iteracji na rysunku 3.2, można również odczytać czas całej symulacji, który wyniósł dokładnie 52 min 43 sek. W przypadku analizy wielu (setek) sieci, czas jest więc wyraźnie zbyt długi. Przeanalizowano wpływ ilości iteracji na średni czas przejazdu (Rys. 3.3). Natomiast średni wynik uzyskany przez agentów został przedstawiony na rysunku 3.4.

Na rysunkach 3.3 i 3.4 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*. Ponadto, analizując rysunek 3.2 można wnioskować, iż najbardziej czasochłonnym etapem jest etap realnej symulacji ruchu. W związku z tym, najlepszym sposobem na ograniczanie czasu potrzebnego do ukończenia obliczeń, jest zmniejszenie liczby iteracji w symulacji.

Po analizie danych z rysunków 3.3 i 3.4 zdecydowano, by w projekcie *GRoNO* użyć 10 iteracji podczas analizy każdej sieci. Wartość ta optymalnie łączy zalety szybkiego spadku wartości, średnich czasów podróży oraz czasu potrzebnego na obliczenia.



Rys. 3.3: Graf średniego czasu przejazdów agentów dla stu iteracji symulacji



Rys. 3.4: Graf wyników agentów dla stu iteracji symulacji

3.3 Opis badanego miasta Sioux Falls

Główną zaletą korzystania z *MATSim* jest dość pokaźny zbiór danych przykładowych. Jednym z nich jest materiał zaprezentowany przez twórców aplikacji, który udostępnili w 2013 roku na spotkaniu użytkowników platformy [13]. Przykład ten dotyczy miasta Sioux Falls w Południowej Dakocie. Domyślnie scenariusz symulacji ruchu drogowego składa się z:

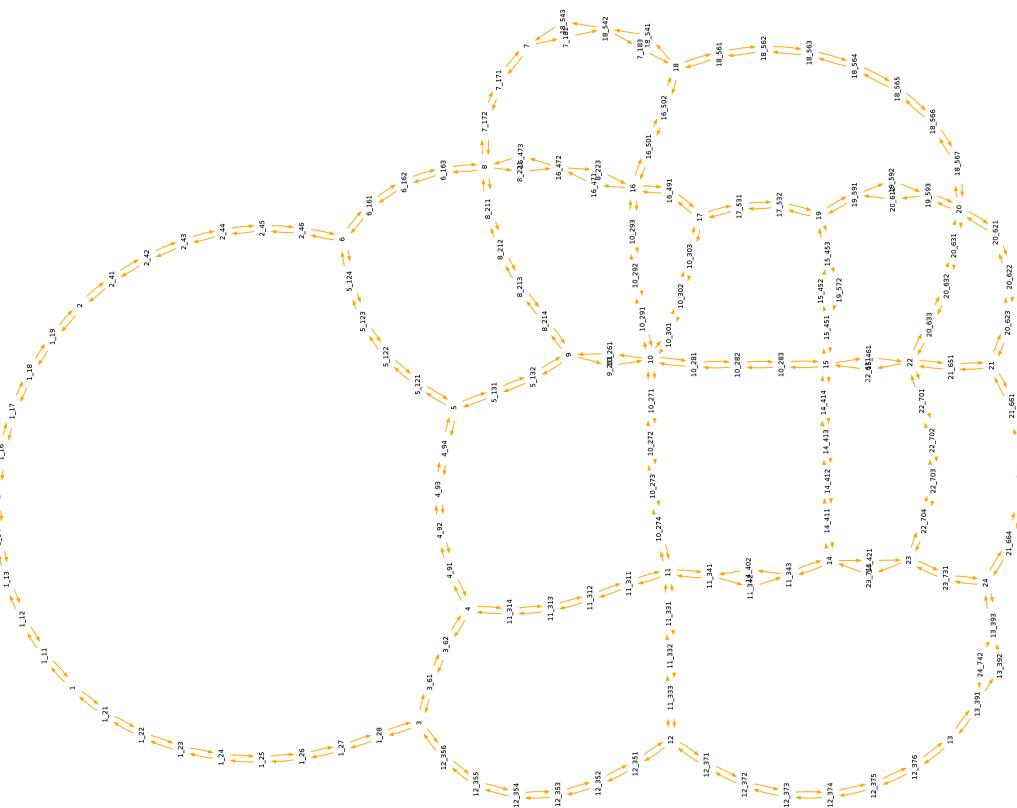
- dwóch grup zapotrzebowania bez charakterystyk socjodemograficznych:
 - 68094 agentów z samochodem oraz korzystających z transportu publicznego,
 - 40877 agentów posiadających samochód.
- dostosowanej sieci drogowej miasta Sioux Falls,
- transportu publicznego razem z rozkładem jazdy,
- przykładowych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

Dostarczony materiał danych jest zbyt obszerny na potrzeby eksperymentu zrealizowanego w niniejszej pracy. Zatem dostosowano go poprzez usunięcie elementów transportu publicznego oraz wyposażenie każdego agenta we własny samochód. Powyższe modyfikacje znacznie wzmogły ruch w mieście (co było pozytywnym efektem) i skróciły obliczenia związane z symulacjami. Na rysunku 3.5 przedstawiono sieć miasta odwzorowaną w postaci grafu. Rysunek został obrócony o 90° w lewo, dla zwiększenia jego czytelności. Natomiast na rysunku 3.6 zaprezentowano rozkład miejsc pracy, domostw i innych zakładów nałożonych na graf sieci drogowej miasta.

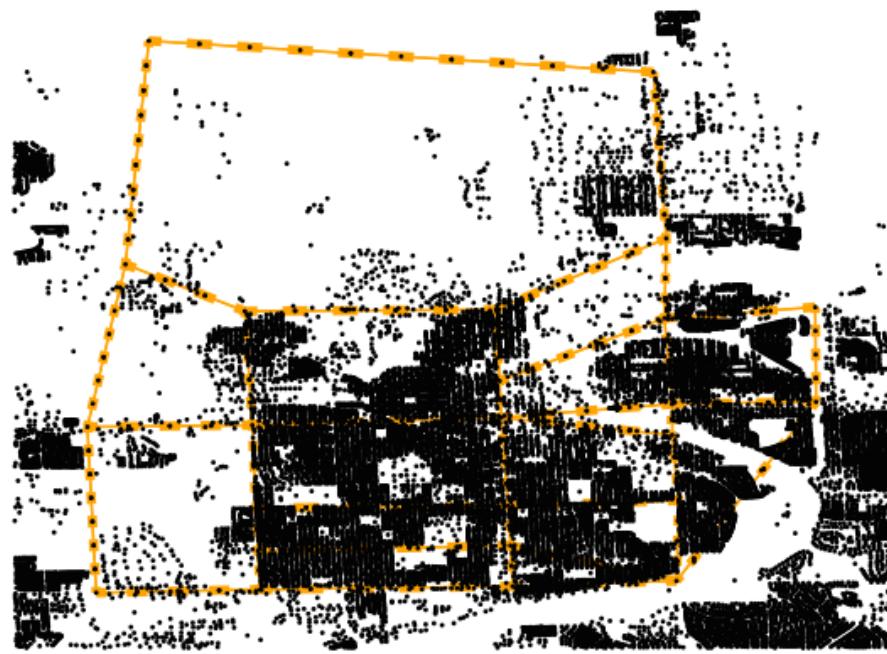
Na rysunku 3.7 i 3.8 zaprezentowano ruch drogowy w najbardziej intensywnych godzinach dnia, to jest w godzinach szczytu. Jest on przedstawiony jako natężenie ruchu na każdej krawędzi (ulicy), poprzez odniesienie go do możliwości przepustowości każdego węzła. Kolor zielony oznacza małe obciążenie, żółty — średnie, a czerwony wysokie natężenie ruchu w tym miejscu. Rysunki zostały obrócone o 90° w lewo, dla zwiększenia ich czytelności.

W powyższym rozdziale opisano wybrany system symulacji transportu ruchu drogowego *MATSim* wraz z jego ustawieniami. Przedstawiono zastosowane parametry symulatora oraz przykładowe miasto wykorzystane w badaniach, prezentowanych w części praktycznej. W kolejnym rozdziale omówiono teoretyczną stronę projektu *GRONO*, prezentując jego biblioteki oraz konfigurację uruchomienia.

3.3. OPIS BADANEGO MIASTA SIOUX FALLS

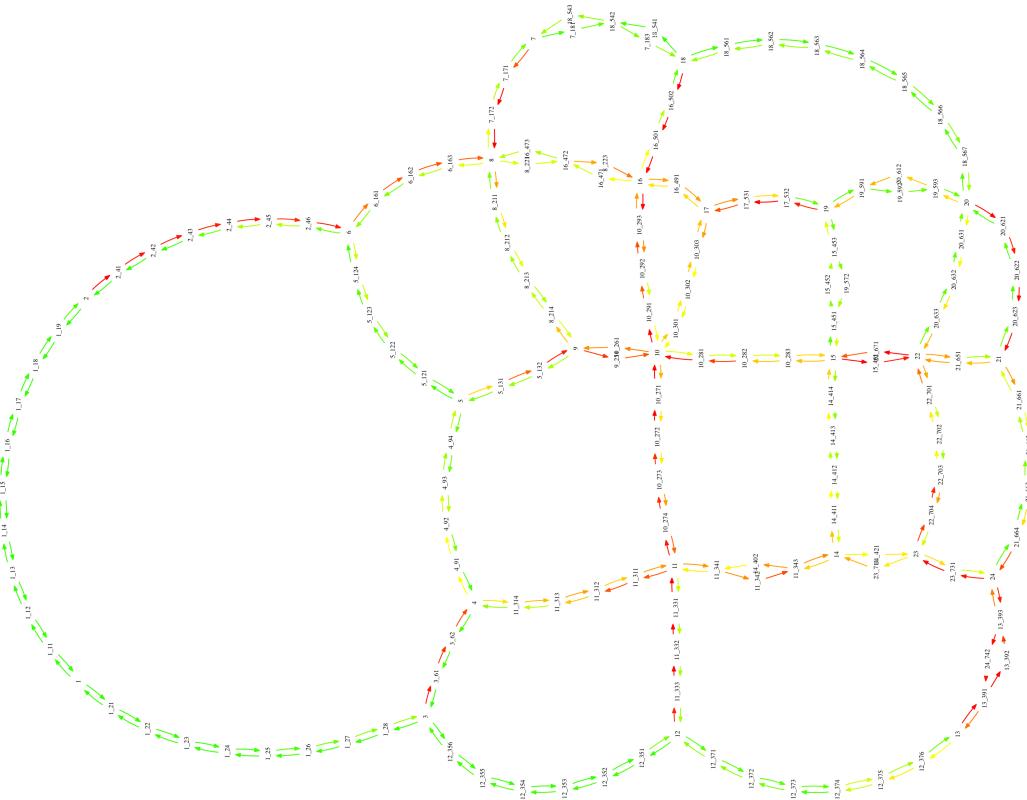


Rys. 3.5: Graf sieci prezentowanego miasta Sioux Falls

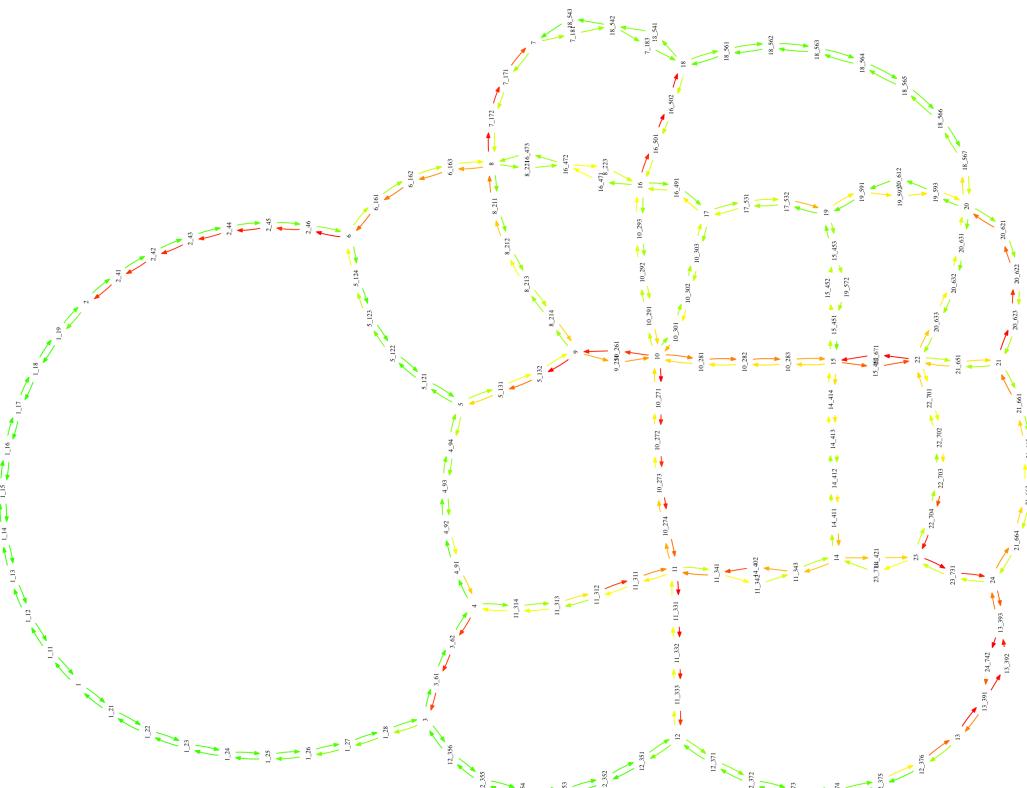


Rys. 3.6: Rozkład budynków na grafie miasta Sioux Falls

3.3. OPIS BADANEGO MIASTA SIOUX FALLS



Rys. 3.7: Natężenie ruchu w mieście Sioux Falls w godzinach 6.00 - 7.00



Rys. 3.8: Natężenie ruchu w mieście Sioux Falls w godzinach 16.00 - 17.00

3.3. OPIS BADANEGO MIASTA SIOUX FALLS

Rozdział 4

Proces realizacji projektu GRoNO

W niniejszym rozdziale opisano technologie informatyczne, które zostały użyte podczas tworzenia rozwiązania *GRoNO*. Wyjaśniono działanie kolejnych elementów projektu oraz przedstawiono użyte biblioteki zewnętrzne. Zaprezentowany został również sposób wdrożenia rozwiązania, jak i dane użyte podczas zrealizowanych badań.

4.1 Technologie i metodologie programistyczne

Użyte w projekcie technologie programistyczne są poniekąd wymuszone przez języki programowania, w których zostały stworzone wykorzystywane rozwiązania pomocnicze. W przypadku symulatora *MATSim* jest to język **Java**. Język **Python** został wykorzystany głównie ze względu na użytą bibliotekę *NetworkX*, obsługującą grafy.

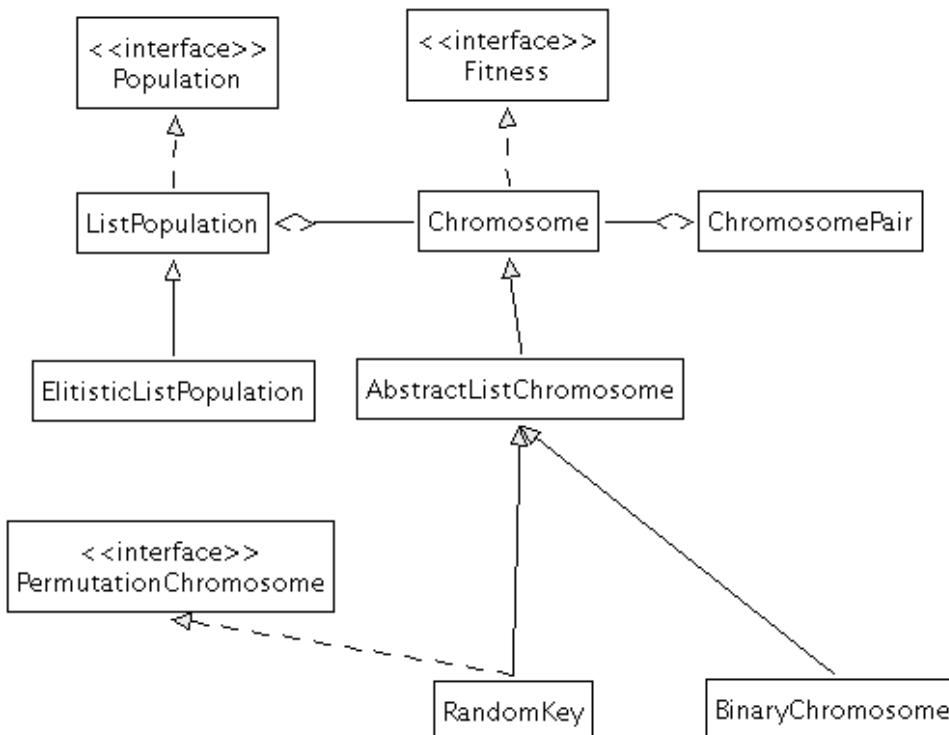
4.2 Biblioteka komponentów algorytmów genetycznych

Projekt **The Apache Commons** jest realizowany 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* [23].

Część *Apache Commons Proper* jest dedykowana do tworzenia i utrzymania komponentów wielokrotnego użycia, w której programiści mogą pracować wspólnie nad projektami współdzielonymi ze społecznością. Celem twórców jest tworzenie indywidualnych elementów posiadających jak najmniej zależności pomiędzy zewnętrznymi rozwiązaniami. Dzięki temu możliwe jest stworzenie niezależnych, integralnych modułów. Dodatkowo, ambicją projektu jest zapewnienie jedynie stabilnych elementów, oferujących wstępную kompatybilność.

W przypadku głównej części *Apache Commons Proper* wyróżnia się wiele modułów, różniących się funkcjonalnością i celem. *Apache Commons Math* jest modułem, zapewniającym rozwiązywanie problemów głównie matematycznych i statystycznych. Znajduje się w nim implementacja podstawowej formy algorytmu genetycznego, którą rozszerzono w niniejszej pracy dyplomowej.

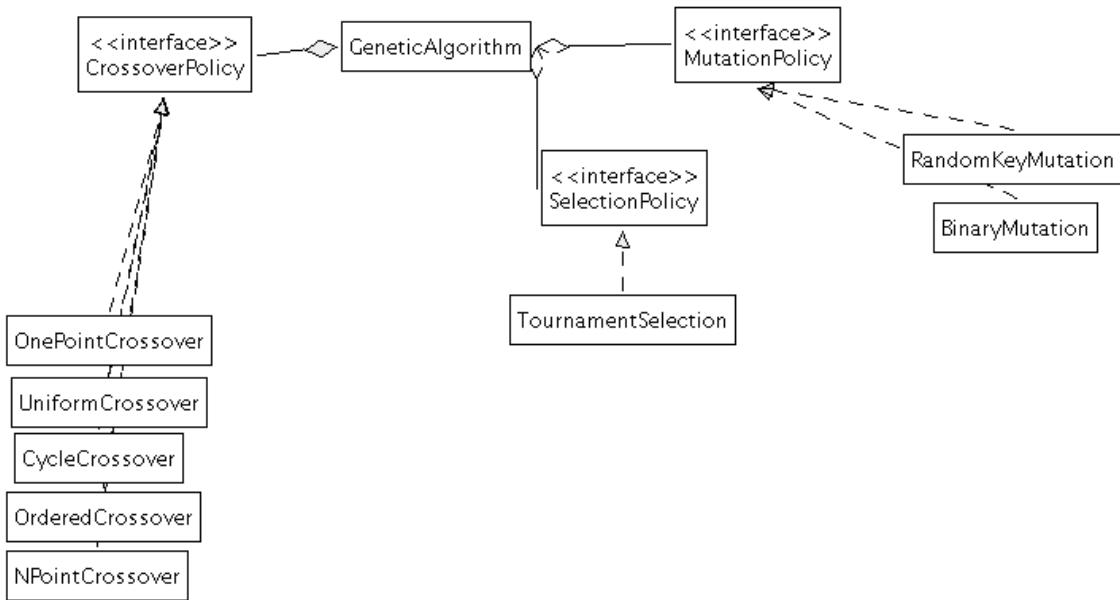
Na rysunkach 4.1 i 4.2 zaprezentowane zostały klasy z projektu *Apache Commons Math* wykorzystane w projekcie *GRoNO*.



Rys. 4.1: Diagram wybranych klas pakietu *Apache Math Genetics*, część 1

Analizując klasy dostarczone przez pakiet *Apache Math Genetics* (Rys. 4.1) wyraźnie widać jedynie podstawową strukturę algorytmu genetycznego. Biblioteka dysponuje oczywiście klasą `BinaryChromosome`, który idealnie spełnia swoje zadanie w klasycznym algorytmie genetycznym. Dodatkowo, na diagramie uwzględniono klasę `ElitisticListPopulation`, odpowiadającą elitarnej strategii selekcji, opisanej w podrozdziale 2.2.3. Dostarczone klasy w wygodny sposób pozwalają programistie na dołączenie własnej funkcji dostosowania poprzez implementację interfejsu `Fitness`.

Na diagramie zaprezentowanym na rysunku 4.2 przedstawiono klasy odpowiadające za sam przebieg działania algorytmu genetycznego. Znalazły się tutaj podstawowe klasy odpowiadające metodom, omawianym w podrozdziale 2.2.3, takie jak `BinaryMutation` — klasyczna mutacja, `TournamentSelection` — selekcja turniejowa oraz kilka klas dotyczących krzyżowania. W projekcie *GRoNO* wykorzystano krzyżowanie jednopunktowe, które zostało zrealizowane w klasie `OnePointCrossover`.



Rys. 4.2: Diagram wybranych klas z pakietu *Apache Math Genetics*, część 2

Oczywiście w projekcie wiele klas podstawowych zostało przeciążonych tak by realizowały niektóre specyficzne aspekty projektu *GRONO*. Schemat działania samego algorytmu został jednak zachowany.

4.3 Biblioteka do obsługi grafów

Ponieważ moduł do wizualizacji rozwiązań *MATSim* nie spełniał oczekiwania zarówno pod względem wydajności, jak i stabilności zdecydowano się na stworzenie autorskiej jego implementacji. W tym celu wykorzystano 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* [24]. Wybór biblioteki oparty był głównie na dostarczanych przez nią gotowych implementacjach, które w pełni pokrywały się z wymaganiami projektowymi.

Przede wszystkim bardzo wygodna w użyciu była pełna integracja biblioteki z popularną biblioteką graficzną *matplotlib*, skupiającą się na rysowaniu wykresów oraz grafach. Dzięki temu połączeniu struktury grafów tworzone w NetworkX mogły być wizualizowane bez konieczności konwersji.

Ponadto, biblioteka zawiera gotowe klasy związane z różnymi typami grafów, w tym grafami skierowanymi opisany w sekcji 2.3.2. Działając na gotowej strukturze można w pełni korzystać z wbudowanych algorytmów do analizy i przeszukiwania grafów. Metody te zostały przedstawione w sekcjach 2.3.3 oraz 2.3.4.

4.4 Uruchomienie obliczeń w chmurze

Ze względu na duże wymagania sprzętowe obliczeń, a zarazem ograniczoną przenośność rozwiązań z powodu skorzystania z języka *Python*, zdecydowano się na usprawnienie rozwiązania. Wykorzystując system *Linux Ubuntu* stworzono maszynę wirtualną, spełniającą wszystkie wymagania do uruchomienia aplikacji. Dzięki temu stało się możliwe wykorzystanie innych komputerów oprócz tego, na którym zostało stworzone rozwiązanie. Podczas badań wykorzystywany był system **Linux Ubuntu 12.04 LTS** [25].

Korzystając z systemu opisanego powyżej, dzięki któremu projekt jest możliwy do zainstalowania na innych maszynach, do obliczeń wykorzystano zewnętrznego dostawcę mocy obliczeniowej. Przy wyborze decydującym czynnikiem była cena rozwiązania, co w przypadku chmury **Microsoft Azure** [26] pozwoliło na darmowe rozwiązanie¹. Do wyboru użytkownik posiada dość szeroką gamę konfiguracji, które może dostosować idealnie do swoich potrzeb. W przypadku projektu *GRoNO* kluczowymi aspektami była ilość niezależnych wątków obliczeniowych. Z sekcji maszyn wirtualnych Linux, do projektu najlepiej nadawała się warstwa podstawowa, przeznaczona do wystąpień ogólnego zastosowania. Jest to ekonomiczna opcja dla obciążeń związanych z tworzeniem aplikacji i serwerów testowych, które nie wymagają równoważenia obciążień, automatycznego skalowania i maszyn wirtualnych, korzystających z dużej ilości pamięci. Na rysunku 4.3 przedstawiono ofertę maszyn wirtualnych możliwych do uruchomienia w chmurze Microsoft Azure dla klientów końcowych. W pracy wykorzystywana została opcja A4.

WYSTĄPIENIE	RDZENIE	PAMIĘĆ RAM	ROZMIARY DYSKÓW	CENA
A0	1	0.75 GB	20 GB	€0.0135/godzina (~€10/mies.)
A1	1	1.75 GB	40 GB	€0.0328/godzina (~€25/mies.)
A2	2	3.5 GB	60 GB	€0.0656/godzina (~€49/mies.)
A3	4	7 GB	120 GB	€0.1311/godzina (~€98/mies.)
A4	8	14 GB	240 GB	€0.2622/godzina (~€196/mies.)

Rys. 4.3: Dostępne konfiguracje warstwy podstawowej chmury Microsoft Azure

¹Dzięki darmowemu okresowi próbnemu.

4.5 Obsługa projektu GRONO

Ponieważ przeprowadzane w ramach projektu obliczenia wymagają wiele czasu, zdecydowano się na pominięcie interfejsu użytkownika przy projektowaniu aplikacji. Całość jest obsługiwana przez plik konfiguracyjny, który zostaje wczytany na początku działania programu i za jego pomocą kontrolowany jest przebieg obliczeń. Przykładowy plik konfiguracyjny z opisem jego funkcji przedstawiono na listingu 4.1. Pomimo dwóch osobnych technologii, w których został wykonany projekt *GRONO*, są one od siebie zależne. Całość jest obsługiwana przez część wykonaną w języku *Java*, skrypty w języku *Python* wspierają jedynie niektóre procesy.

```
<?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>
    </scenario>
    <genetics>
        <population-size>24</population-size>
        <max-generations>200</max-generations>
        <elitism-rate>2</elitism-rate>
        <crossover-rate>1.0</crossover-rate>
        <mutation-rate>0.8</mutation-rate>
        <tournament-arity>4</tournament-arity>
    </genetics>
</config>
```

Listing 4.1: Plik konfiguracyjny projektu GRONO

Analizując listing 4.1 można wyróżnić trzy podstawowe grupy konfiguracyjne dostępne w projekcie *GRONO*:

- *project*,
- *scenario*,
- *genetics*.

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

Opcje dostępne w grupie *project* są następujące:

- *name* — nazwa projektu, używana jako katalog wyjściowy,
- *output dir* — katalog, gdzie zapisano wyniki optymalizacji sieci drogowych,
- *threads* — ilość wątków, które mają być użyte podczas symulacji transportu ruchu drogowego,
- *log level* — poziom logowania *Log4J*²,
- *python path* — ścieżka instalacji języka Python,
- *python main* — folder ze skryptami pomocniczymi,
- *java-path* — ścieżka instalacji języka Java,
- *matsim-jar* — ścieżka do biblioteki *MATSim*,
- *matsim-xmx* — maksymalna pamięć RAM dostępna dla symulatora *MATSim*.

Opcje sekcji *scenario* to:

- *config* — ścieżka dostępu pliku konfiguracyjnego *MATSim*,
- *network* — ścieżka dostępu pliku z siecią wejściową scenariusza symulacji,
- *population* — ścieżka dostępu pliku z populacją scenariusza symulacji,
- *facilities* — ścieżka dostępu pliku z budynkami scenariusza symulacji,
- *iterations* — ilość iteracji symulacji *MATSim*.

Na zakończenie, opcje grupy *genetics* są następujące:

- *population size* — rozmiar populacji,

²zewnętrzna biblioteka do logowania, dostępne poziomy: DEBUG, INFO, WARN, ERROR i FATAL

- *max generations* — ilość testowanych generacji (warunek stopu),
- *elitism rate* — ilość najlepszych chromosomów biorących udział w kolejnej iteracji,
- *crossover rate* — szansa na krzyżowanie osobników z poprzedniej populacji przed dodaniem ich do kolejnej generacji,
- *mutation rate* — szansa na mutację pojedynczego genu wybranych osobników,
- *tournament rate* — ilość osobników biorących udział w turnieju.

4.6 Schemat działania projektu GRONO

Poniżej przedstawiono kolejność operacji wykonywanych w ramach projektu *GRONO*. Analiza dotyczy oczywiście ogólnego schematu, nie wywołań konkretnych metod w odpowiednich klasach. Analiza ta ma na celu przybliżenie sposobu działania optymalizacji oraz wyjaśnienie użyteczności bibliotek zewnętrznych oraz wykorzystania zewnętrznego systemu oceny – symulatora ruchu drogowego.

Wszystkie kroki następujące po uruchomieniu projektu *GRONO* ujęte są poniżej, następnie przedstawiono je na diagramie znajdującym się na rysunku 4.4. Cały proces zostaje następnie szerzej omówiony.

1. Wczytanie pliku konfiguracyjnego.
2. Przygotowanie struktury plików według podanych parametrów.
3. Wczytanie wejściowej sieci drogowej wraz z ustawieniami symulatora stosowanymi podczas wszystkich uruchomień.
4. Zastosowanie sieci wejściowej jako osobnika wzorcowego populacji genetycznego.
5. Ocena funkcji przystosowania osobnika wzorcowego.
 - (a) Sprawdzenie w bazie wyników czy dana sieć drogowa nie była już poddana symulacji.
Jeśli nie istnieje wynik symulacji dla danej sieci drogowej:
 - i. Uruchomienie symulatora *MATSim* z wcześniej ustalonimi parametrami i zadana siecią drogową.
 - ii. Zapisanie wyniku średniego czasu podróży agentów w bazie.
 - iii. Stworzenie dodatkowych grafów i wykresów wizualizujących otrzymaną sieć i jej wynik.

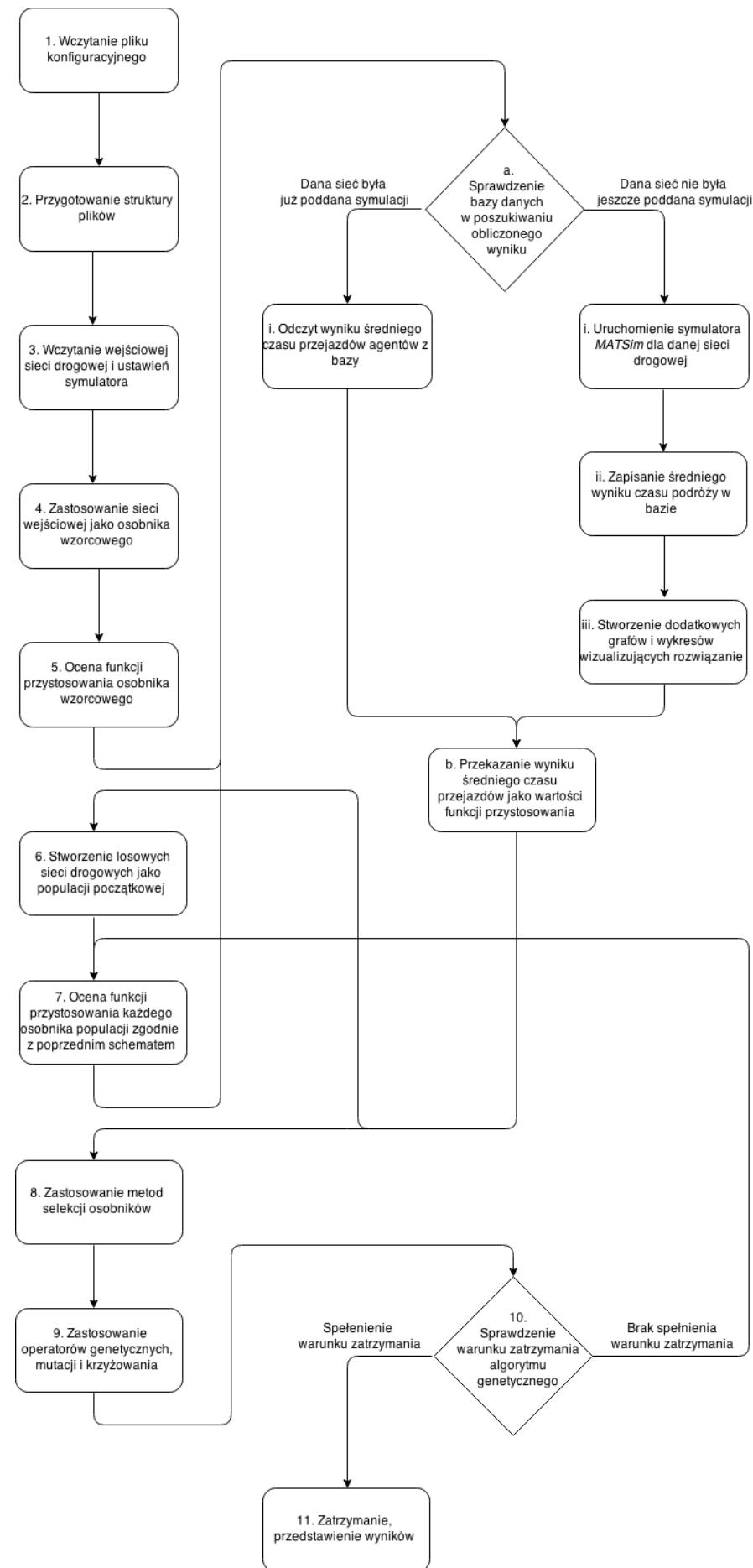
4.6. SCHEMAT DZIAŁANIA PROJEKTU GRONO

W przypadku gdy sieć była już poddana symulacji:

- i. Odczyt wyniku średniego czasu przejazdów agentów z bazy danych.
- (b) Przekazanie wyniku średniego czasu przejazdu agentów dla danej sieci drogowej jako wartość funkcji przystosowania algorytmu genetycznego.
6. Stworzenie losowych sieci drogowych jako populacji początkowej algorytmu genetycznego.
7. Ocena każdego osobnika zgodnie z działaniami opisanymi w punkcie 5.
8. Zastosowanie metod selekcji osobników.
9. Zastosowanie operatorów genetycznych, mutacji i krzyżowania.
10. Do momentu osiągnięcia zadanej liczby iteracji powtarzanie czynności od kroku 7.
11. Przy osiągnięciu warunku zatrzymania, przedstawienie wyników.

Przede wszystkim analizując schemat z rysunku 4.4 można zauważyc, że struktura w dużej części pokrywa się z klasycznym algorytmem genetycznym, opisany w sekcji 2.2.2. Oprócz pierwszych trzech kroków, reszta wywołań z głównej ścieżki jest oparta o ten algorytm, realizując zadanie optymalizacji sieci. Krok 1, wczytanie pliku konfiguracyjnego, został szerzej wytlumaczony w sekcji 4.5. W kroku 2, przygotowanie struktury plików, oraz 3, wczytanie wejściowej sieci drogowej i ustawień symulatora, wykonywane są mniej istotne operacje związane z przygotowaniem programu do uruchomienia obliczeń. Najistotniejsza część następuje po kroku 4, zastosowaniu sieci wejściowej jako osobnika wzorcowego, w którym rozpoczyna się właściwa optymalizacja. Kluczowym aspektem jest informacja na temat sieci drogowej, która zgodnie z opisem w podrozdziale 2.5 jest tłumaczona z postaci tablicy binarnej do grafu i odwrotnie. Proces ten został zrealizowany z użyciem biblioteki obsługującej grafy opisywanej w podrozdziale 4.3. W projekcie *GRONO* zastosowano lokalną bazę danych, w której przechowywane są wyniki symulacji sieci drogowych. Rozwiążanie to pozwoliło uniknąć sytuacji, w której badana jest dwukrotnie ta sama sieć, tracąc czas i moc obliczeniową na ponowną symulację, która w ostateczności daje ten sam wynik. Powtórzenia w przypadku losowego tworzenia i mieszania populacji były niestety nieuniknione. W kroku 9, zastosowaniu operatorów genetycznych, również zastosowane zostały pewne usprawnienia. W przypadku losowego zmieniania wartości w tablicy binarnej możliwa była bowiem sytuacja utworzenia sieci drogowej, której graf nie spełniałaby warunku spójności. Rozwiążanie problemu opisane zostało w podrozdziale 2.3.4. Generalnie, by uniknąć problematycznej sytuacji stworzenia niewłaściwej sieci drogowej, po zastosowaniu operatora krzyżowania

4.6. SCHEMAT DZIAŁANIA PROJEKTU GRONO



Rys. 4.4: Schemat działania aplikacji *GRoNO*

i mutacji, sieć zostaje sprawdzona. W przypadku niewłaściwej kombinacji, podejmowane są dodatkowe próby, których liczba jest określona w konfiguracji projektu *GRONO*. W przypadku wielokrotnych niepowodzeń operatory nie są stosowane i sieć przechodzi w stanie niezmienionym do kolejnej populacji. Sytuacja ta jest oczywiście wyjątkowo rzadka, ponieważ szanse wielokrotnego niepowodzenia w przypadku zamknięcia jednego węzła sieci są bardzo małe. Projekt *GRONO* powtarza kroki optymalizacji sieci drogowej dopóki nie zostanie osiągnięta założona w konfiguracji projektu liczba iteracji. Po zakończeniu działania programu prezentowane są wyniki, które zostały przedstawione w rozdziale 5.

4.7 Struktura plików projektu **GRONO**

Umieszczone na płycie dołączonej do niniejszej pracy źródła pozwalają na odtworzenie pełnego projektu, używając wymienionych wcześniej środowisk programistycznych, *Eclipse* i *PyDev*. Struktura katalogów, w kolejności alfabetycznej wraz z krótkim opisem zawartości, jest następująca:

fakematsim/

Jest to sztuczna implementacja symulatora *MATSim*, pozwalająca na testowanie działania algorytmu genetycznego wykorzystując losowe wyniki. Działanie opiera się na rozpakowaniu gotowego katalogu z wcześniej przygotowanymi obliczeniami oraz podmianę wyniku na liczbę losową. 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 wykorzystując strukturę projektu *Maven*. Zawiera przykładowe pliki konfiguracyjne i przygotowane testy *JUnit*, pozwalające na ich bezpośrednie wykorzystanie.

literature/

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

matsim/

Katalog ten zawiera skompilowane wersje symulatora *MATSim* oraz jego falsyfikatu wykorzystywanego podczas testów. Symulator został skompilowany ze źródeł dostępnych na repozytorium głównym projektu³. Drobne modyfikacje dotyczyły tylko parametrów logowania oraz uruchomienia symulatora. Nie zostały dokonane żadne zmiany ingerujące w przebieg samej symulacji.

sioux-out/

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

paper/

Katalog ze źródłami pracy pisemnej w formacie TeX.

python/

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

README.md

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

scenarios/

Zawiera przykładowe scenariusze symulacji transportu ruchu drogowego, które mogą być wykorzystane przy pracy z symulatorem *MATSim*. Znajduje się tutaj, oprócz wykorzystanego miasta Sioux Falls, również scenariusze dla miast Berlin i Bruksela. Ponadto, zawiera on parę przykładowych plików konfiguracyjnych symulatora.

W powyższym rozdziale opisano technologie informatyczne związane z projektem *GRONO*. Zaprezentowane zostały biblioteki informatyczne, z których skorzystano w trakcie realizacji projektu. Objasniono konfigurację niezbędną do uruchomienia projektu *GRONO* oraz opisano strukturę plików projektu. W kolejnym rozdziale przedstawione zostaną zoptymalizowane sieci drogowe, uzyskane w wyniku działania stworzonego programu.

³<https://svn.code.sf.net/p/matsim/source/matsim/trunk>

4.7. STRUKTURA PLIKÓW PROJEKTU GRONO

Rozdział 5

Analiza procesu optymalizacji sieci drogowej

W niniejszym rozdziale przedstawiono rezultaty działania projektu *GRoNO*. Omówiono również parametry konfiguracji wykorzystane podczas tych badań wraz z ich wartościami. Na zakończenie przedstawiono wyniki optymalizacji przykładowej sieci drogowej, które następnie przeanalizowano.

5.1 Użyte ustawienia projektu GRoNO

Podczas badań przeprowadzonych w ramach niniejszej pracy dyplomowej wykorzystane zostały ustawienia algorytmu genetycznego zgodne z listingiem 5.1. Wartości te były dobrane na podstawie doświadczenia nabyciego w trakcie studiów literaturowych.

```
<genetics>
    <population-size>24</population-size>
    <max-generations>200</max-generations>
    <elitism-rate>2</elitism-rate>
    <crossover-rate>1.0</crossover-rate>
    <mutation-rate>0.8</mutation-rate>
    <tournament-arity>4</tournament-arity>
</genetics>
```

Listing 5.1: Ustawienia algorytmu genetycznego użytego podczas badań

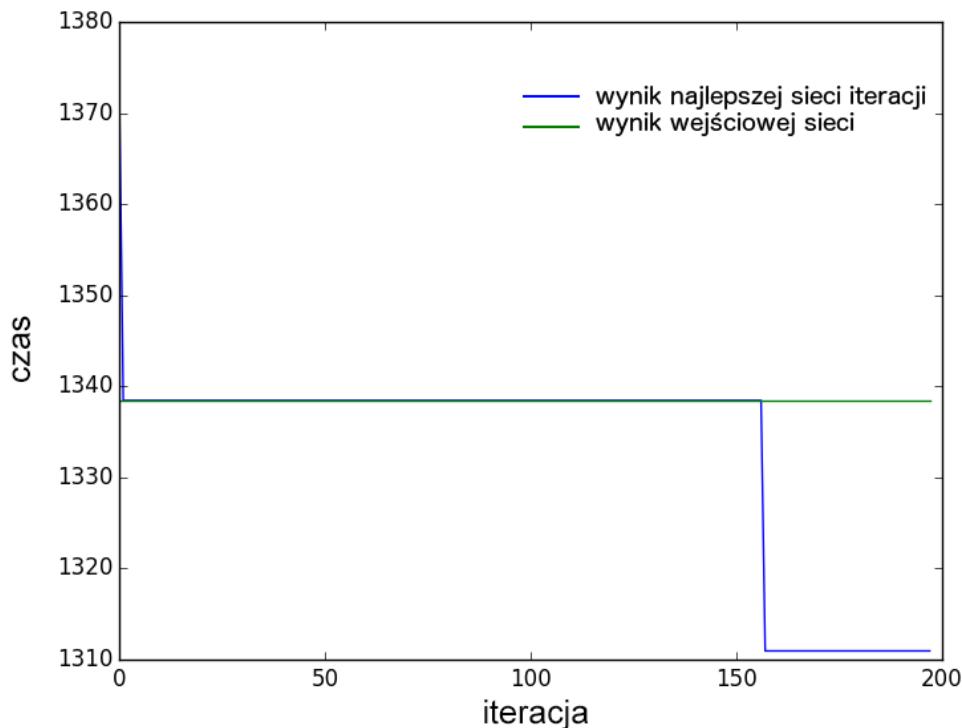
Poniżej znajduje się krótkie omówienie opcji sekcji *genetics* oraz wartości poszczególnych ustawień wykorzystanych w projekcie *GRoNO*, przedstawionych na listingu 5.1. Szczegółowe wyjaśnienie pełnej konfiguracji znajduje się w podrozdziale 4.5.

- *population size* — 24 — rozmiar populacji został dobrany do wielokrotności wątków na serwerze. Ponieważ dostępnych było 8 rdzeni obliczeniowych, wielokrotność pozwalała na zmaksymalizowanie liczby równoległych symulacji *MATSim* podczas pojedynczej generacji.
- *max generations* — 200 — warunek stopu był przede wszystkim motywowany ograniczeniem czasowym obliczeń. Przy liczbie 200 generacji spodziewane obliczenia na serwerze miały zostać zakończone w przeciągu 48 godzin.
- *elitism rate* — 2 — liczba kopiowanych najlepszych rozwiązań do kolejnej generacji była zwiększoną do dwóch z powodu większego rozmiaru populacji (24) i wybranej metody selekcji osobników.
- *crossover rate* — 1.0 — szansa na krzyżowanie osobników z poprzedniej populacji przed dodaniem ich do kolejnej generacji została ustawiona na 1, by mieć pewność, że zostaną stworzeni nowi osobnicy.
- *mutation rate* — 0.8 — szansa na mutację pojedynczego genu wybranych osobników została ustawiona na 0.8 ze względu na dużą przestrzeń przeszukiwań rozwiązania. Takie ustawienie pozwalało maksymalizować szanse znalezienia nowych ekstremów lokalnych.
- *tournament rate* — 4 — ilość osobników biorących udział w turnieju została zwiększoną proporcjonalnie do wielkości populacji. Wciąż liczba ta stanowi tylko 15% z całej generacji.

5.2 Wyniki optymalizacji sieci drogowej miasta Sioux Falls

Wykres przedstawiający zestawienie wyniku najlepszej sieci wyłonionej w danej iteracji, do wyniku sieci wejściowej przedstawiony został na rysunku 5.1. Jego analiza pokazuje, iż bardzo długo najlepsze rozwiązania z zamkniętymi jakimkolwiek węzłami sieci drogowej uzyskiwały gorsze wyniki globalne niż sieć drogowa z wszystkimi ulicami dostępnymi. Szczęśliwie po około 150 iteracjach zostało odnalezione minimum, w którym zamknięcie pewnych ulic powodowało uzyskanie lepszego wyniku.

Ponieważ w niniejszym projekcie wykorzystany został elitarny model populacji algorytmu genetycznego, wykres przedstawiony na rysunku 5.1 nie przedstawia praktycznie żadnych zmian w populacji, która została ulepszona. Pojedyncze rozwiązanie, które okazało się najlepszym wynikiem badań posiada jednak mutacje, które przedstawiają ciekawą



Rys. 5.1: Wykres zmian najlepszego wyniku iteracji

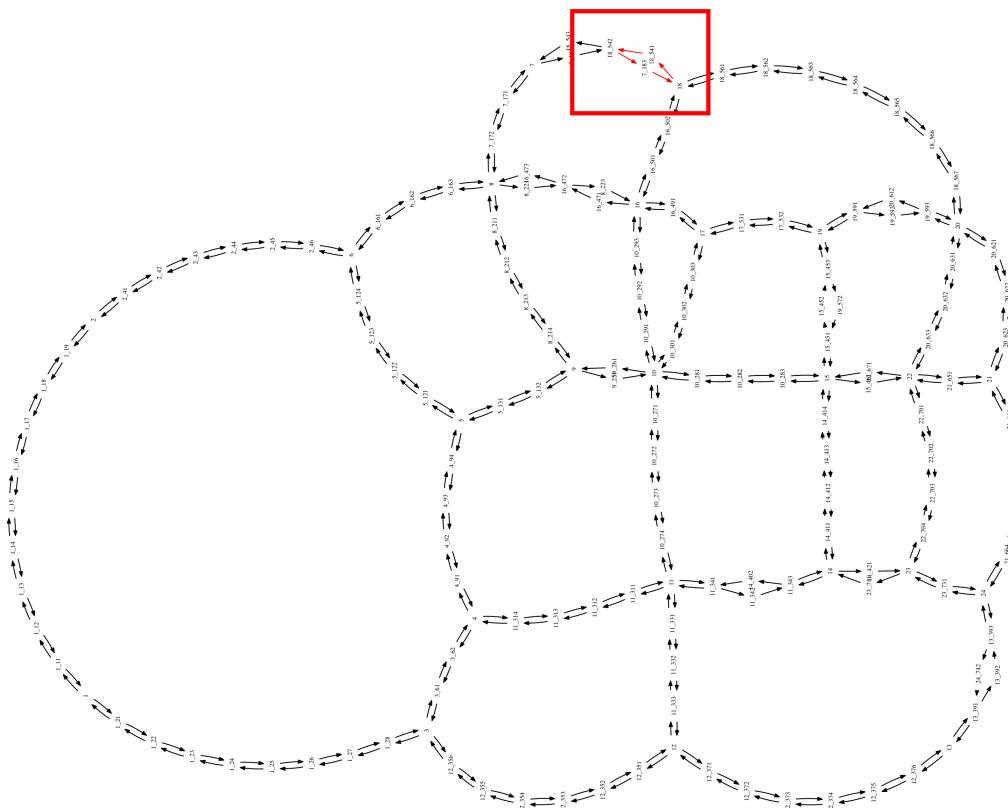
wariację tego rozwiązania. Wyniki wszystkich chromosomów były zachowywane w bazie danych, dzięki czemu możliwe jest przedstawienie wszystkich rozwiązań spełniających założenia projektowe.

Najlepsze wyniki uzyskane podczas badań przedstawione zostały w tabeli 5.1. Znajdują się tutaj wszystkie sieci drogowe, które uzyskały wynik lepszy od sieci wejściowej, to jest 1338.4505528474617. **Od teraz sieci te będą nazywane zgodnie z ich numerem identyfikacyjnym, podanym w pierwszej kolumnie.** Reprezentacja binarna opisuje zamknięte ulice w grafie. Kolejność ulic reprezentowana przez ciąg binarny jest zawsze taka sama. Została ona ustalona przez algorytm wczytywania grafu w bibliotece NetworkX (podrozdział 4.3) i taka reprezentacja daje mało informacji o rzeczywistym wyglądzie danej sieci drogowej. Kolumna *wynik sieci* przedstawia średni czas przejazdu agentów w dziesiątej iteracji symulacji *MATSim*, czyli badany przez nas wynik.

5.2. WYNIKI OPTYMALIZACJI SIECI DROGOWEJ MIASTA SIOUX FALLS

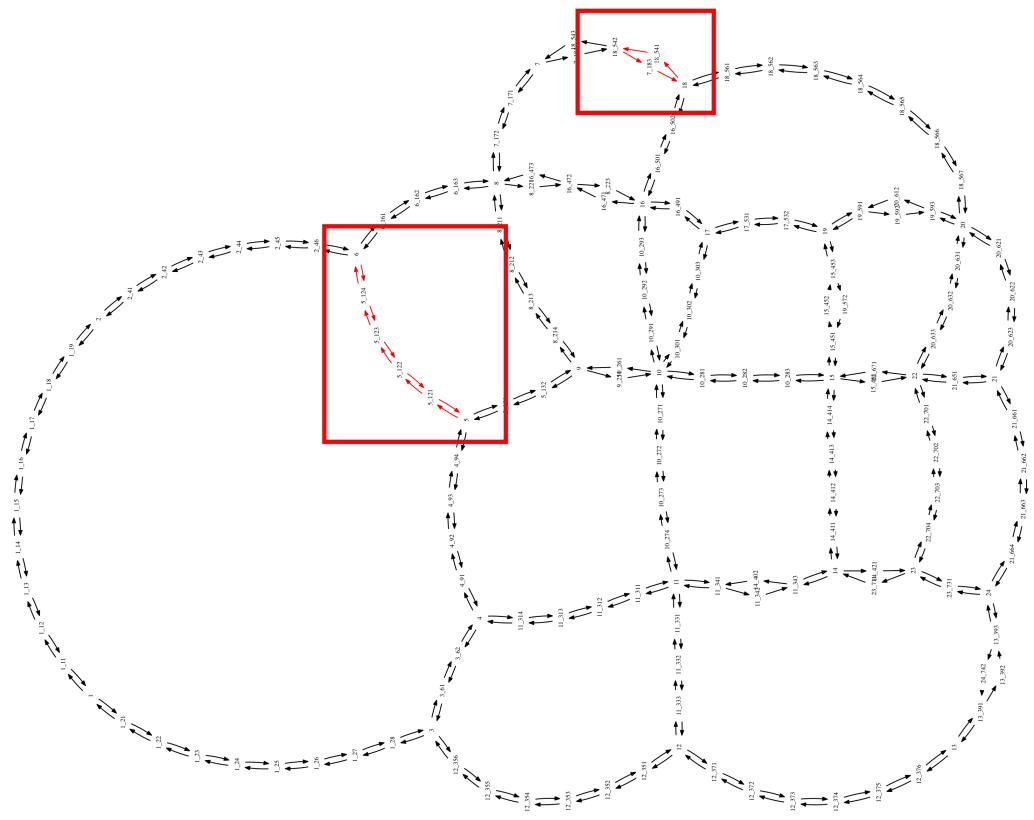
Tablica 5.1: Tabelaryczna reprezentacja otrzymanych najlepszych sieci drogowych

Na rysunkach 5.2 — 5.18 zaprezentowano grafy uzyskanych sieci drogowych, wizualizując rozwiązania wymienione w tabeli 5.1. Zostały one przedstawione kolejno oraz oznaczone odpowiednim numerem identyfikacyjnym. **W celu uproszczenia odnajdowania usuniętych węzłów zostały one zaznaczone czerwonymi prostokątami. Usunięte węzły oznaczone są kolorem czerwonym.** W niektórych przypadkach wycięte węzły są trudno widoczne, z powodu bardzo małej długości strzałki. Dotyczy to przede wszystkim rysunków: 5.6, 5.8, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.17. Tak, jak w poprzednich przypadkach, rysunki przedstawiające sieci drogowe zostały obrócone o 90° w lewo.

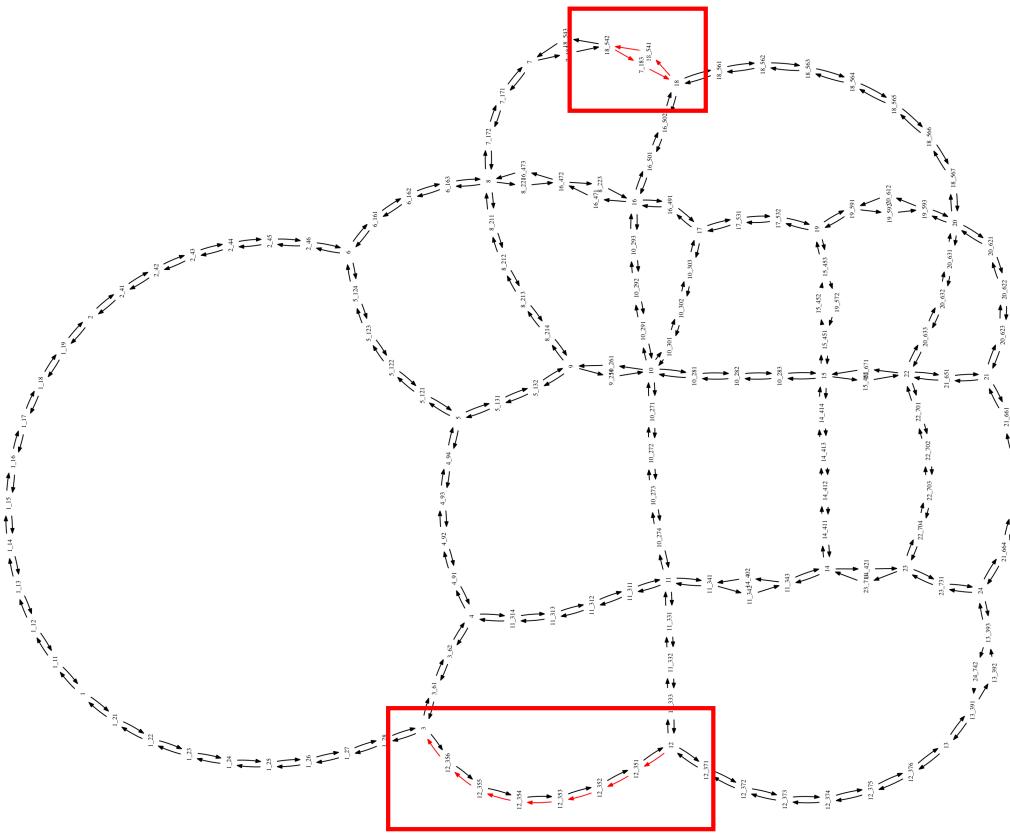


Rys. 5.2: Sieć miasta Sioux Falls, rozwiązanie nr 1

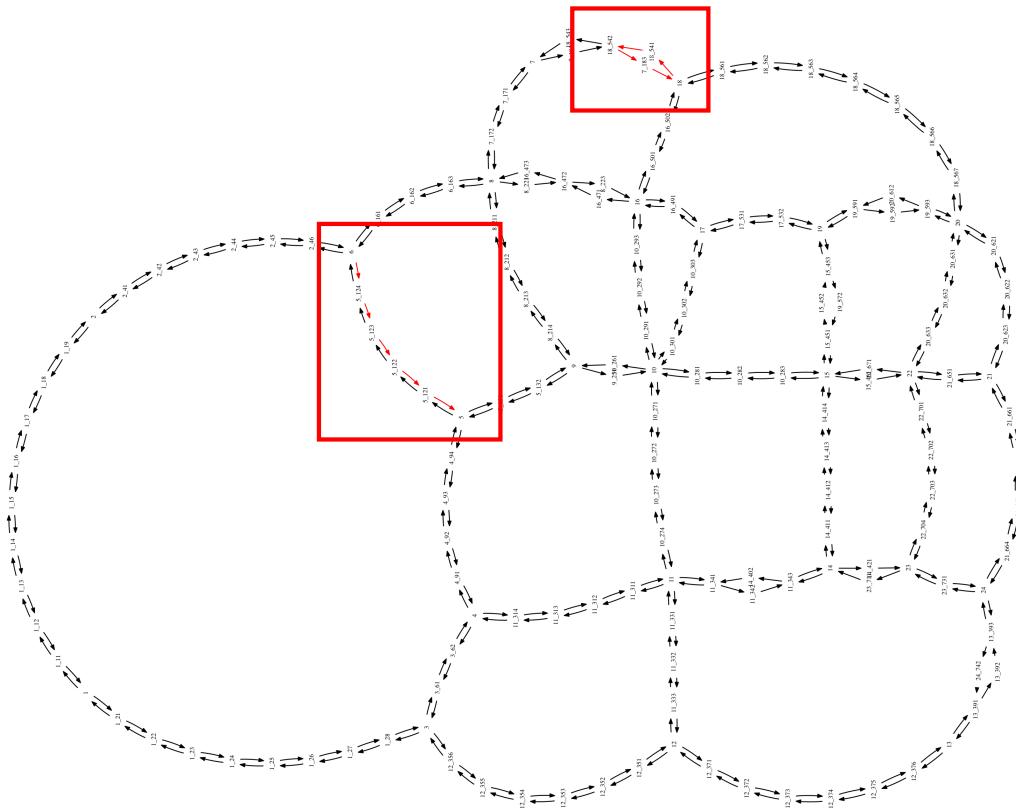
5.2. WYNIKI OPTYMALIZACJI SIECI DROGOWEJ MIASTA SIOUX FALLS



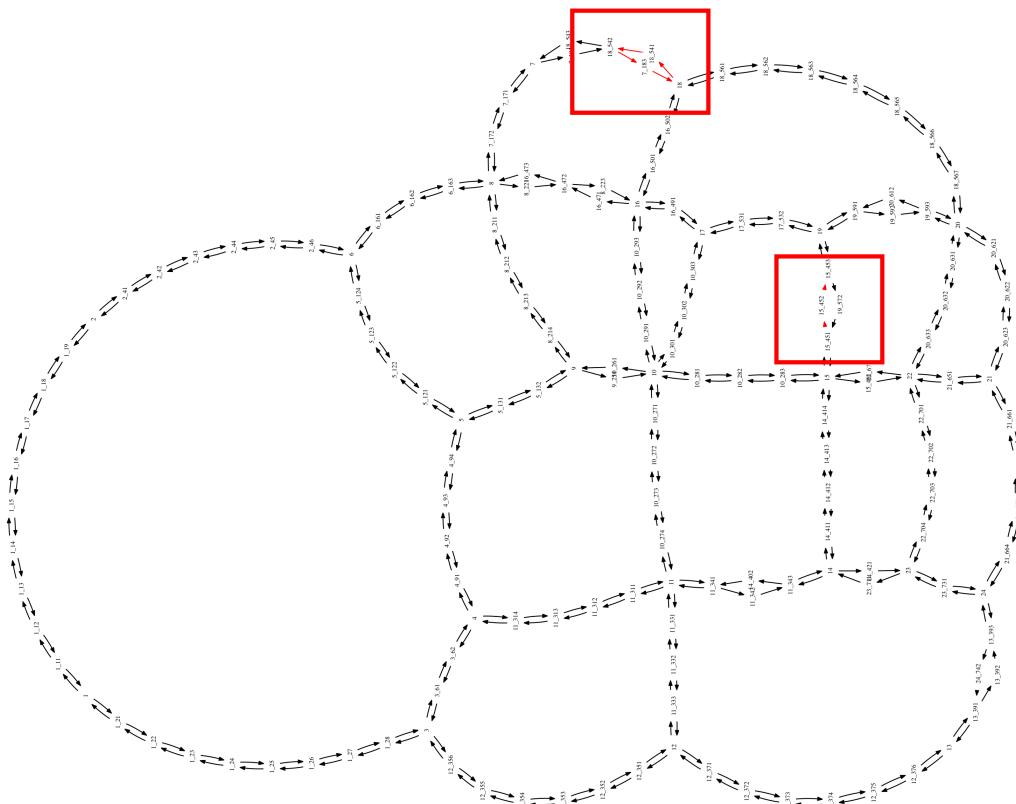
Rys. 5.3: Sieć miasta Sioux Falls, rozwiązańe nr 2



Rys. 5.4: Sieć miasta Sioux Falls, rozwiązańe nr 3

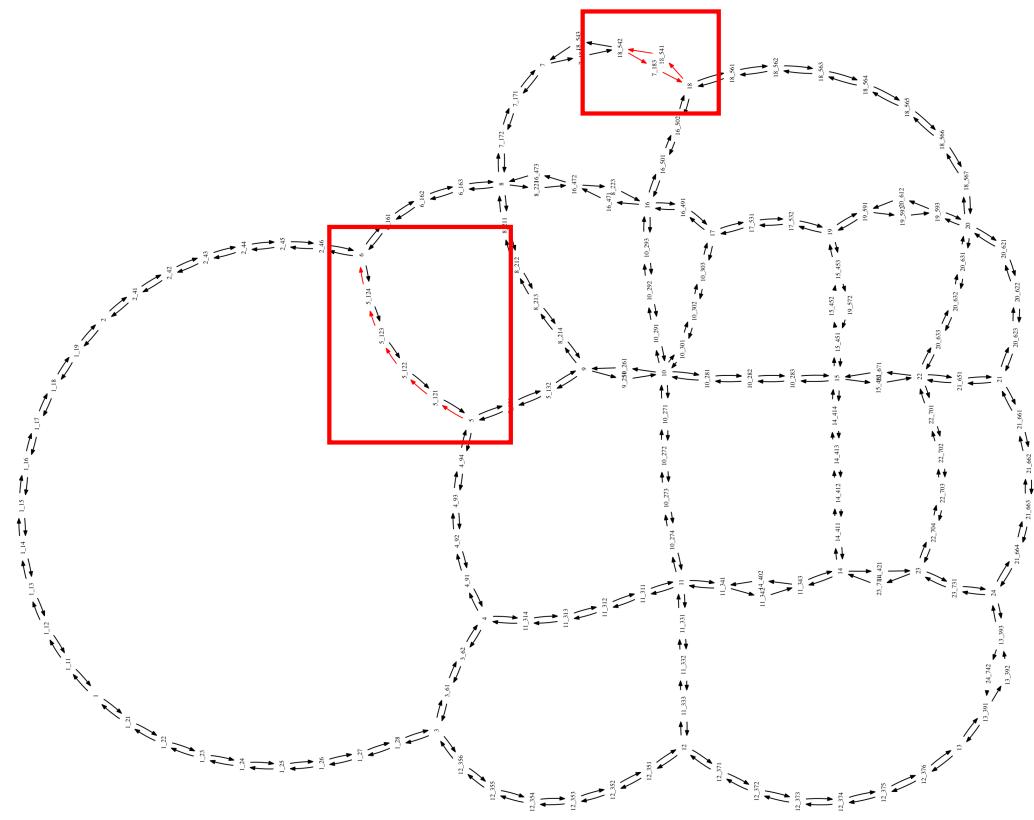


Rys. 5.5: Sieć miasta Sioux Falls, rozwiązanie nr 4

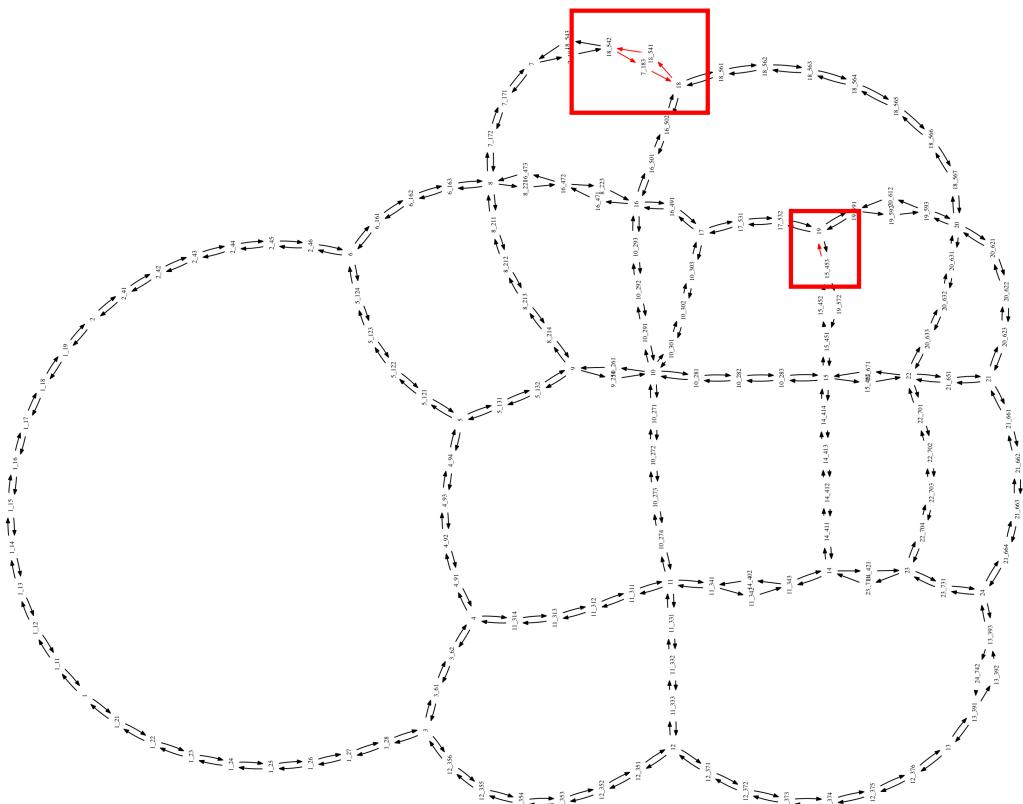


Rys. 5.6: Sieć miasta Sioux Falls, rozwiązanie nr 5

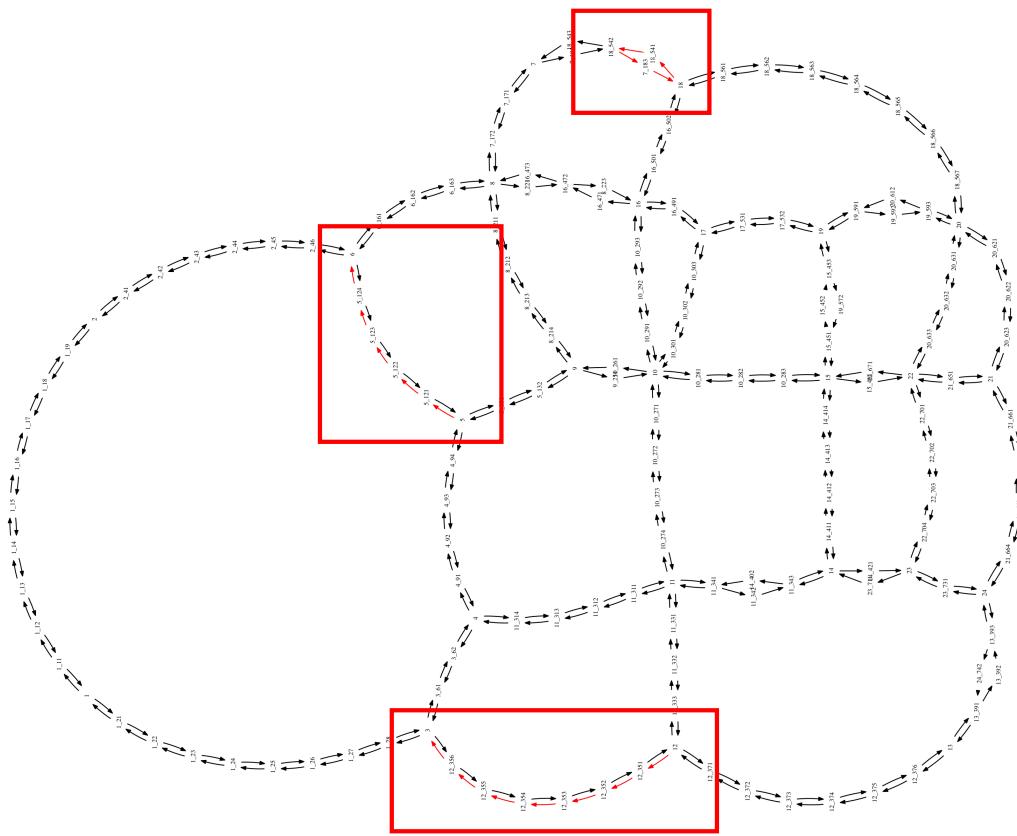
5.2. WYNIKI OPTYMALIZACJI SIECI DROGOWEJ MIASTA SIOUX FALLS



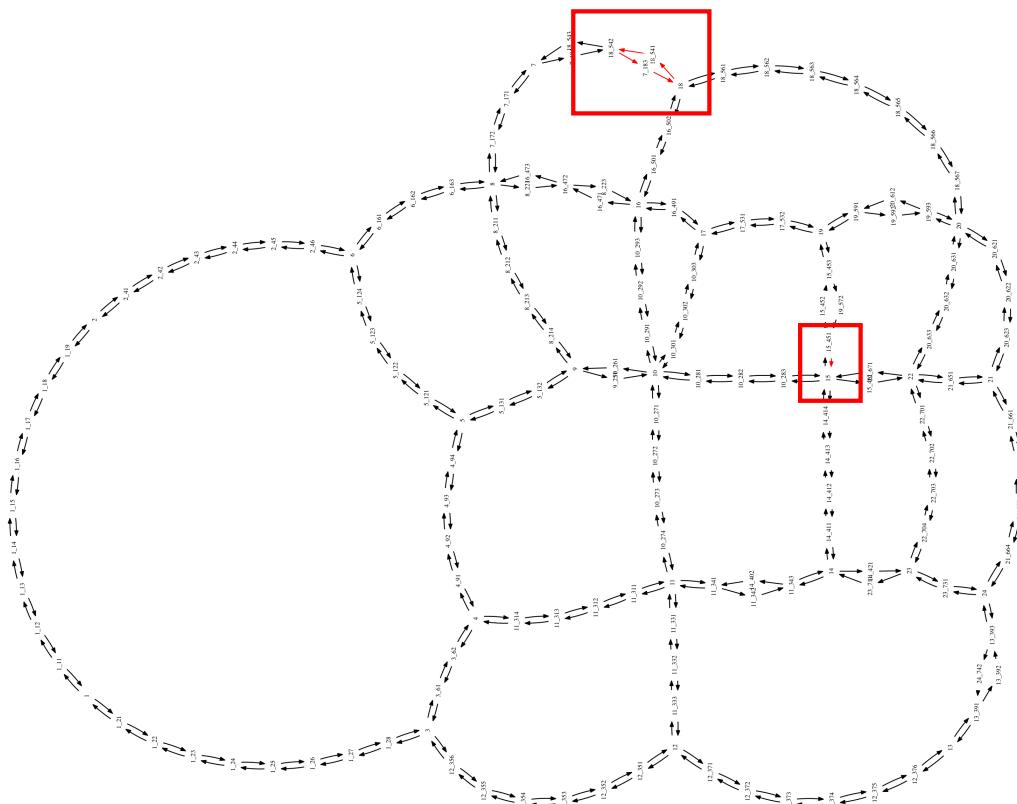
Rys. 5.7: Sieć miasta Sioux Falls, rozwiązańe nr 6



Rys. 5.8: Sieć miasta Sioux Falls, rozwiązańe nr 7

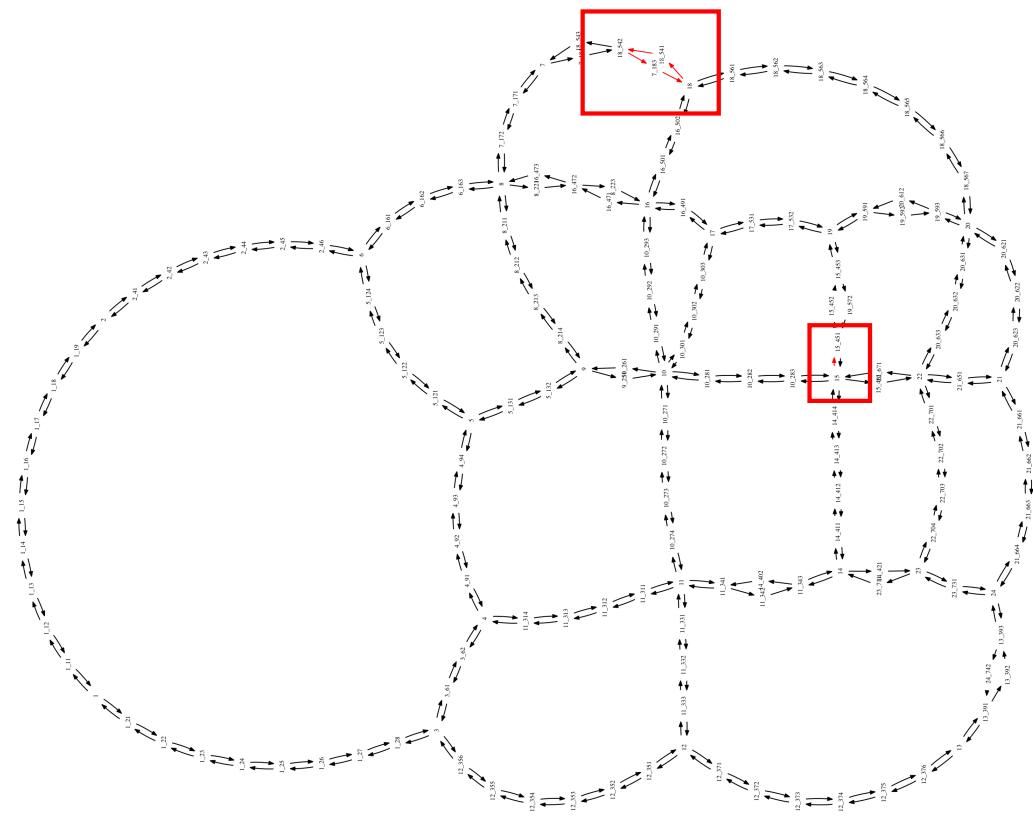


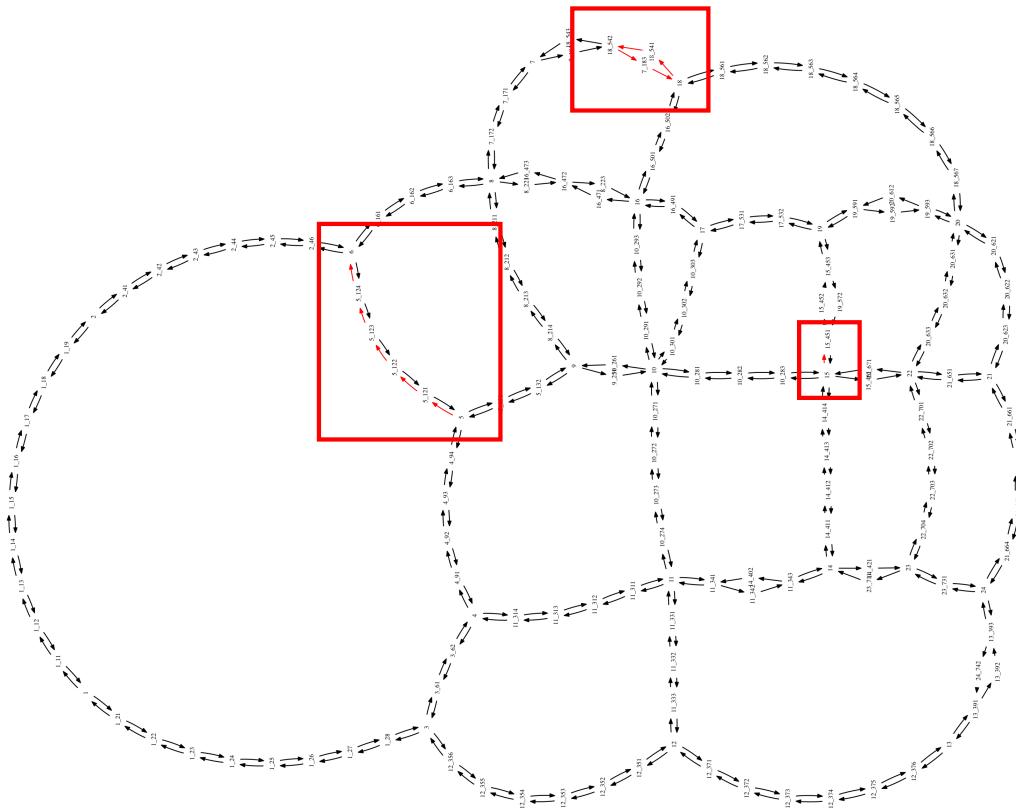
Rys. 5.9: Sieć miasta Sioux Falls, rozwiązańe nr 8



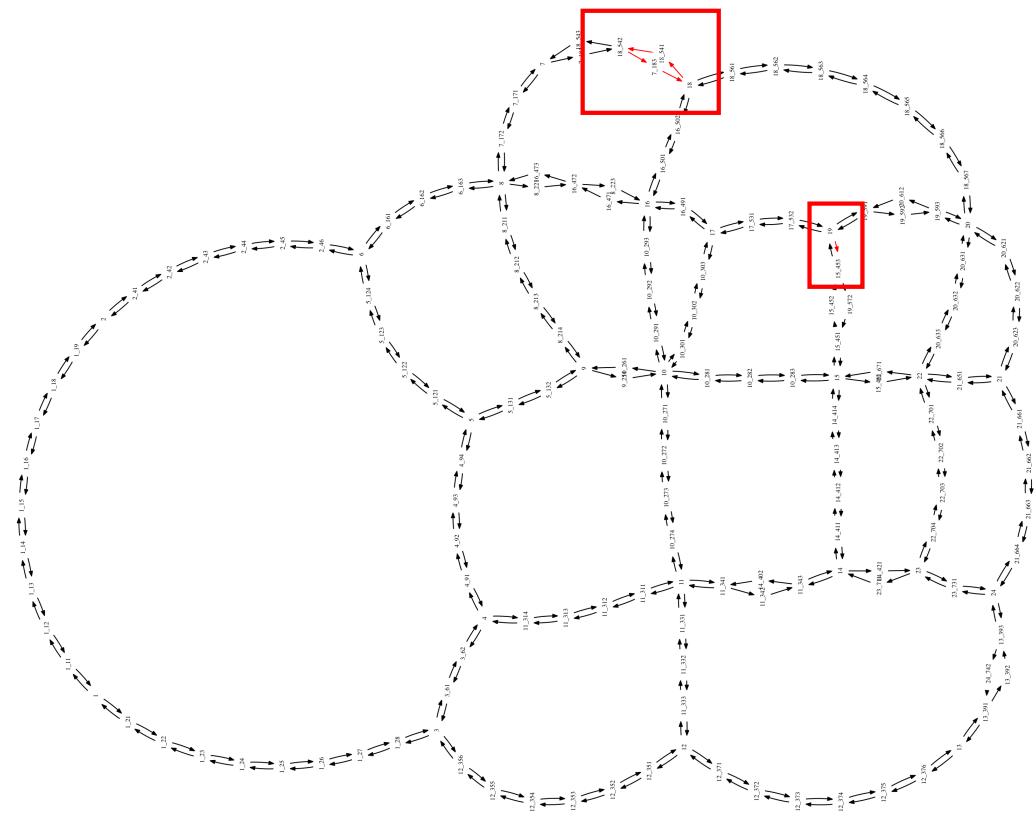
Rys. 5.10: Sieć miasta Sioux Falls, rozwiązańe nr 9

5.2. WYNIKI OPTYMALIZACJI SIECI DROGOWEJ MIASTA SIOUX FALLS

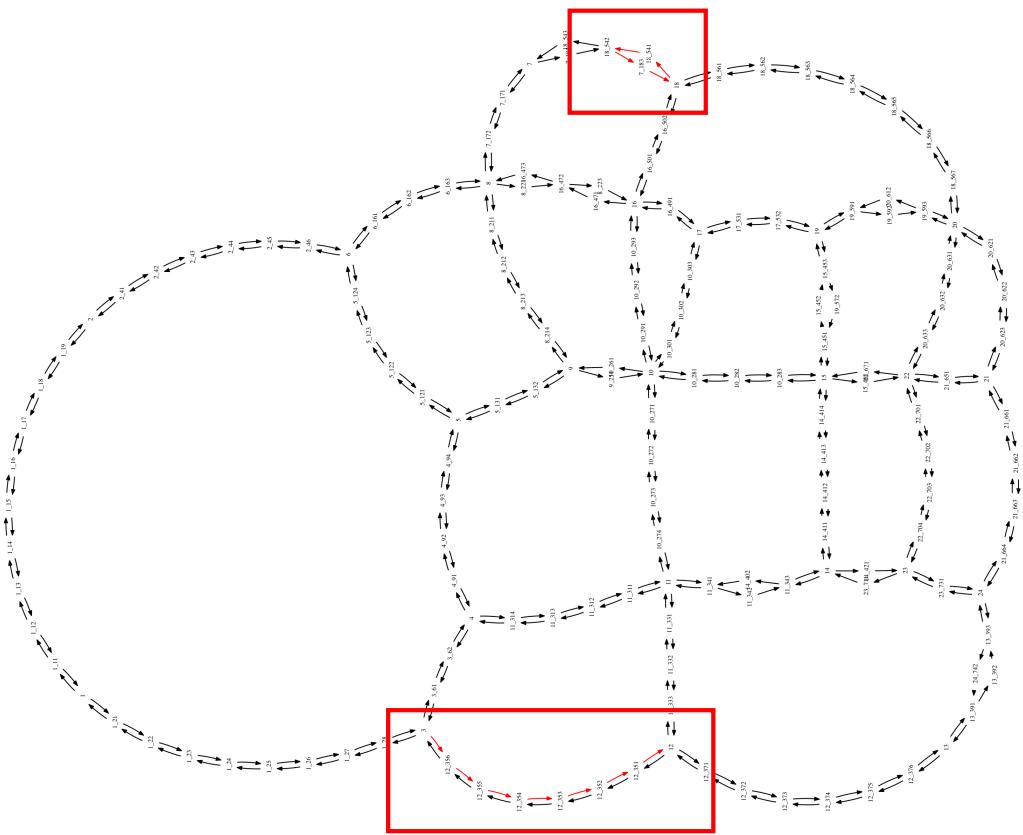




5.2. WYNIKI OPTYMALIZACJI SIECI DROGOWEJ MIASTA SIOUX FALLS

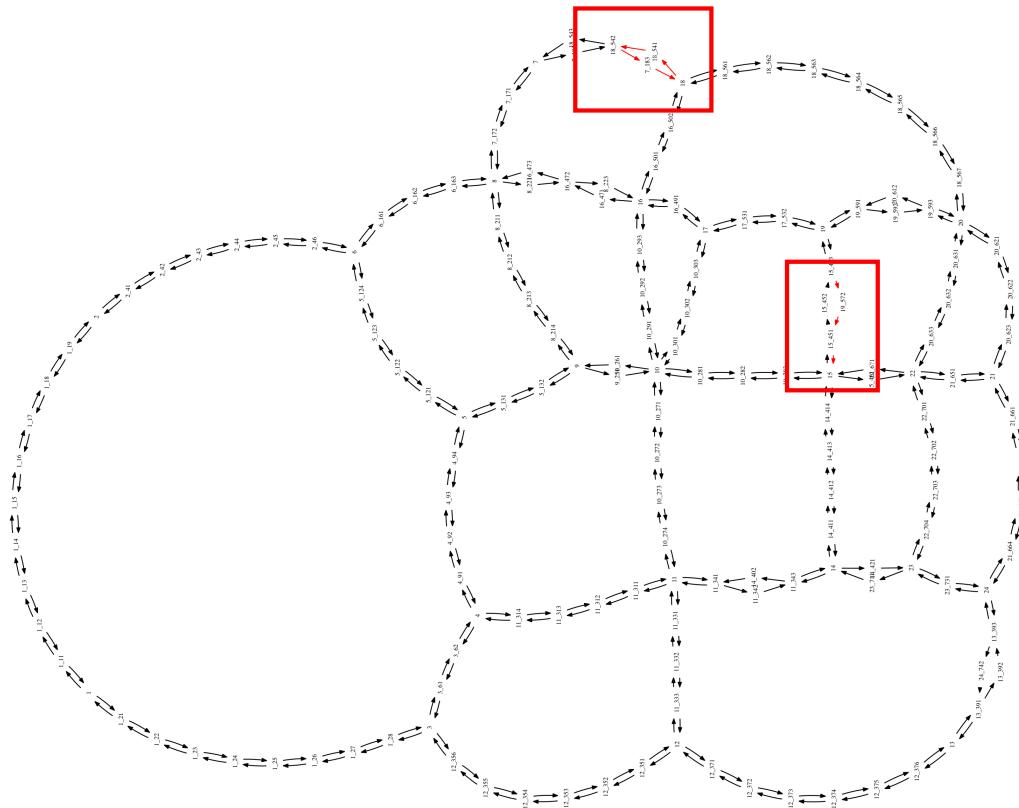


Rys. 5.15: Sieć miasta Sioux Falls, rozwiązańe nr 14

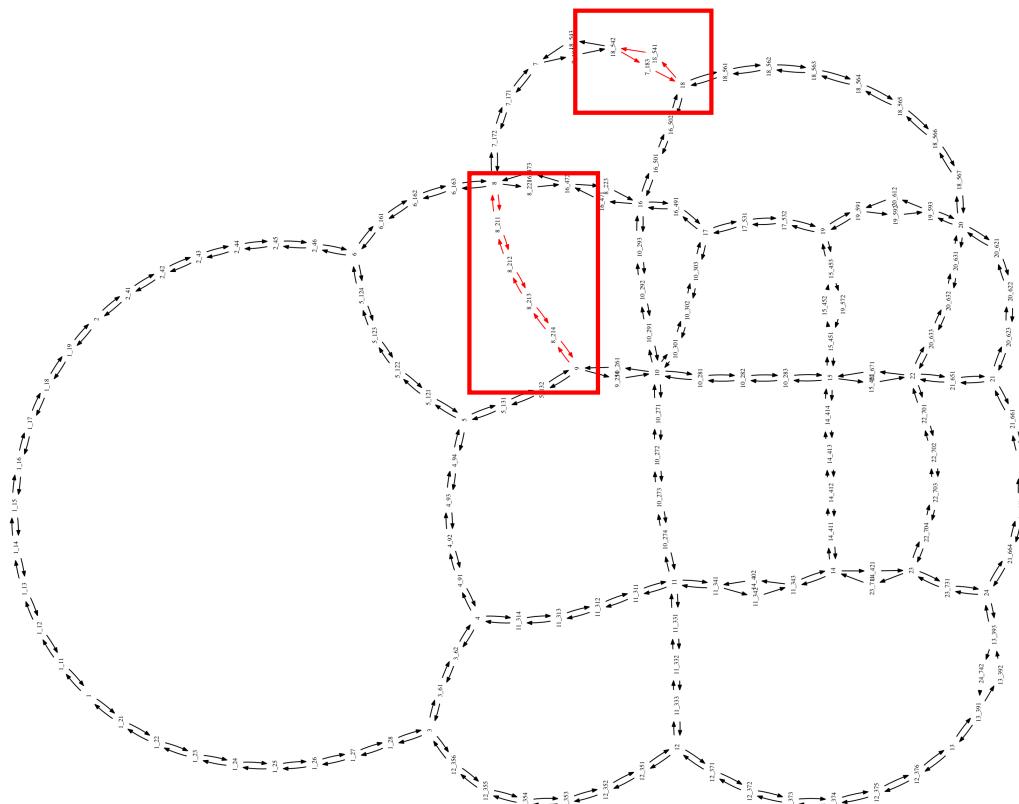


Rys. 5.16: Sieć miasta Sioux Falls, rozwiązańe nr 15

5.2. WYNIKI OPTYMALIZACJI SIECI DROGOWEJ MIASTA SIOUX FALLS



Rys. 5.17: Sieć miasta Sioux Falls, rozwiązańe nr 16



Rys. 5.18: Sieć miasta Sioux Falls, rozwiązańe nr 17

5.3 Analiza uzyskanych wyników sieci drogowych

Pierwszym krokiem, który pomaga w analizie otrzymanych wyników jest ich zgrupowanie. Dość intuicyjnym jest grupowanie w zbiory zamkające podobne obszary (węzły). Wyniki należy interpretować z uwzględnieniem natężenia ruchu panującego w mieście oraz rozkładu budynków. Informacje te zostały przybliżone w podrozdziale 3.3. Natężenie ruchu prezentowane jest na rysunkach 3.7 i 3.8, natomiast rozkład budynków znajduje się na rysunku 3.6.

„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 w mieście. Mowa oczywiście o zamknięciu węzłów między 18 i 18_542 (zaznaczony fragment na rysunku 5.19, obrócony o 90° w lewo). Analizując ten fragment pod kątem rozkładu budynków w mieście, można zauważać duże skupienie w pobliżu tego węzła. Bardzo prawdopodobnym jest tutaj wpływ zamknięcia tych ulic na odległość domostw od najbliższego, możliwego punktu zostawienia samochodu. Podczas zamknięcia tego fragmentu, nastąpiło wydłużenie odcinka pokonywanego bez samochodu dla wielu mieszkańców. Mogło to zmusić ich do wcześniejszego wychodzenia z domu oraz wcześniejszego rozpoczęcia podróży. To z kolei wpłynęło na bardziej równomierne rozłożenie ruchu w godzinach szczytu.

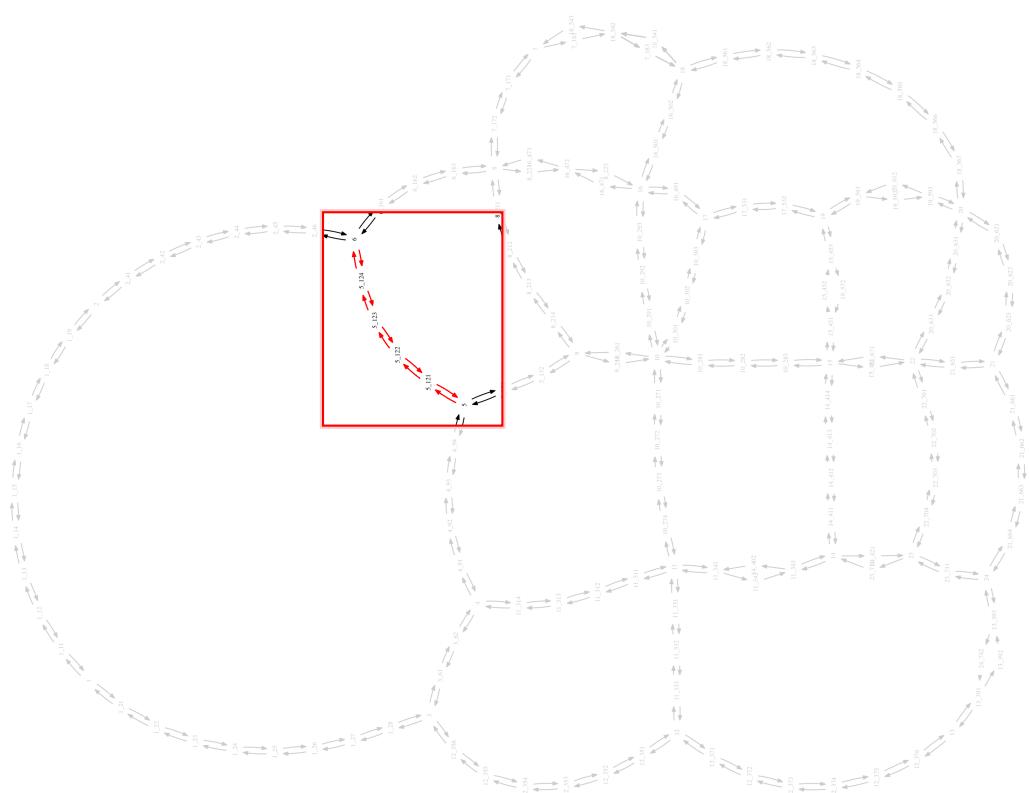
Ciekawym jest jednak obszar wspólny dla wyników prezentowanych na rysunkach 5.6, 5.8, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.17. Jest to obszar znajdujący się pomiędzy wierzchołkami 15 i 19. Analizując położenie tych ulic w mieście, obszar ten znajduje się niedaleko bardzo zakorkowanych ulic, często używanych przez agentów symulacji. Region ten można określić jako centrum miasta. Na pewno wielu agentów dociera tam w ramach pracy lub swoich zainteresowań. Jego usunięcie może znacząco wpływać na wybór drogi, jak również na sposób dojazdu do miejsc znajdujących się w okolicy. Oczywiście, pomimo że w symulatorze węzły te nie były dostępne dla aut, agenci dalej mogli docierać do swoich celów, zostawiając auto w uprzednio wybranym punkcie. Opisywany fragment został przedstawiony na rysunku 5.20.

Drugim obszarem, który został wybrany w przypadku kilku rozwiązań, są ulice pomiędzy skrzyżowaniem (wierzchołkiem) 5 i 6. Został on wybrany przez rozwiązania prezentowane na rysunkach: 5.3, 5.5, 5.7, 5.9, 5.13. Region ten jest jednak oddalony od centrum i prawdopodobnym jest, że zamknięcie drogi wpłynęłoby przede wszystkim negatywnie na dojazd dla wielu osób. Jest to sytuacja analogiczna do tej z analizowanego wcześniej przypadku, przedstawionego na rysunku 5.19. Omawiany fragment prezentowany jest na rysunku 5.21.

Rys. 5.19: Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wszystkich wyników sieci drogowych

5.3. ANALIZA UZYSKANYCH WYNIKÓW SIECI DROGOWYCH

Rys. 5.20: Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wyników sieci drogowych o ID: 5, 7, 9, 10, 11, 12, 13, 14, 16



Rys. 5.21: Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wyników sieci drogowych o ID: 2, 4, 6, 8, 12

5.3. ANALIZA UZYSKANYCH WYNIKÓW SIECI DROGOWYCH

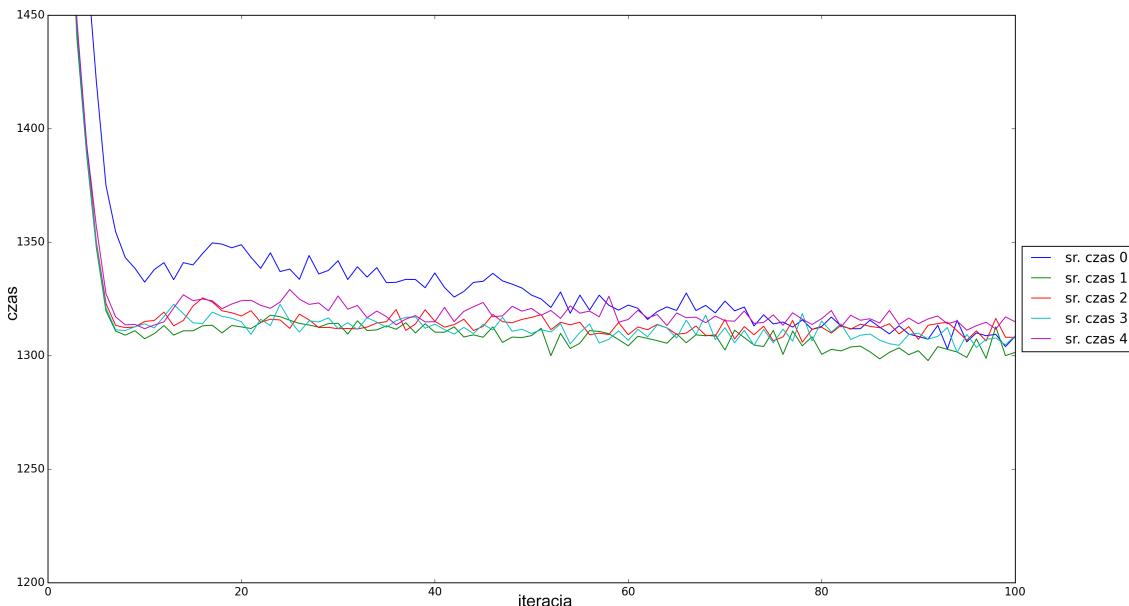
Ostatni obszar, który charakteryzował się wspólnym wynikiem dla więcej niż jednego rozwiązania, to jest dla wyników prezentowanych na rysunkach: 5.4, 5.9, 5.16 znajduje się pomiędzy wierzchołkiem 3 i 12. Zaznaczony fragment przedstawiono na rysunku 5.22, obróconym o 90° w lewo. Wyraźnie jednak region ten nie znajduje się ani na drodze wielu agentów, ani nie jest węzłem często użytkowanym przez agentów. Jego zamknięcie wpłynęło prawdopodobnie na brak gorszej drogi alternatywnej, którą węzeł ten reprezentował w wielu przypadkach.

Rys. 5.22: Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wyników sieci drogowych o ID: 3, 8, 15

Powyższe sugestie mogą zostać użyte w przypadku poszerzonych badań nad danymi obszarami. Decydując, czy faktycznie mają one wpływ na komunikację w mieście trzeba przede wszystkim sprawdzić dokładnie drogi wybierane przez agentów scenariusza. W prezentowanych na rysunkach 5.19 – 5.22 sieciach drogowych zamknięte drogi zwykle nie znajdowały się w kluczowych punktach komunikacyjnych, co przede wszystkim wymusza dogłębniejsze zbadanie algorytmu samej symulacji. 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.

5.4 Analiza zastosowanych ustawień symulacji

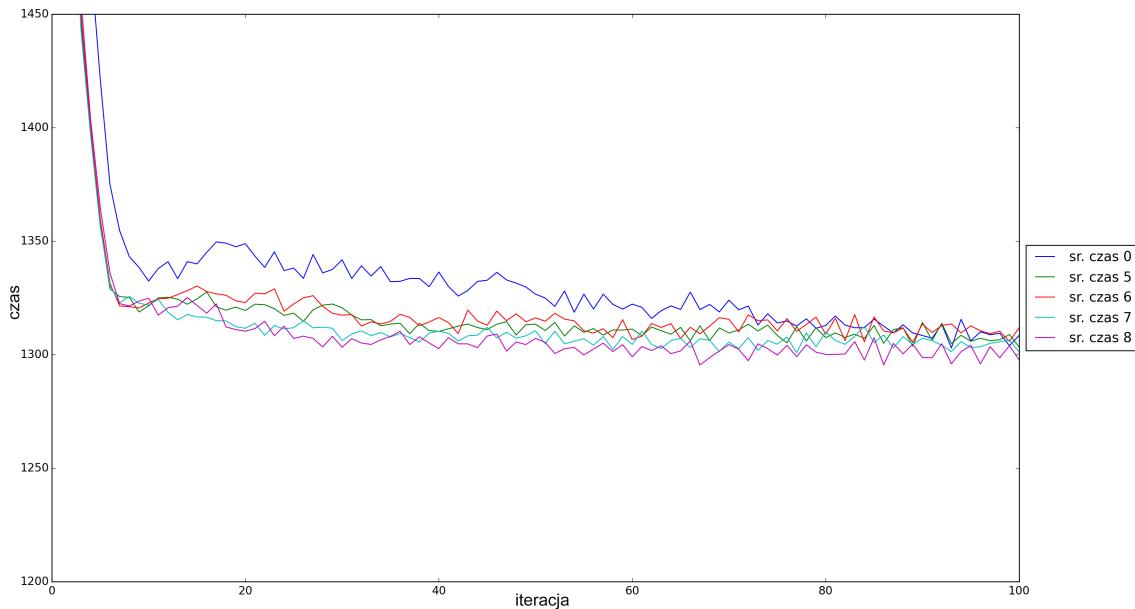
Zgodnie z wcześniejszymi założeniami, podczas poszukiwań bardziej optymalnej sieci drogowej wybrano ustawienie 10 iteracji symulacji do oceny sieci. W celu zweryfikowania wyników, dla najlepszych 17 sieci przeprowadzono ponowną symulację dla 100 iteracji. Wyniki te porównano z wynikami sieci wejściowej. Rysunki 5.23 — 5.27 przedstawiają wyniki tych porównań. Dla poprawienia czytelności wykresy zostały podzielone, prezentując za każdym razem tylko część sieci drogowych. **Legenda wykresów odnosi się do wyniku uzyskanego przez rozwiążanie, które jest przedstawiane w tabeli 5.1.** Numer wyniku sieci odnosi się do numeru identyfikacyjnego sieci. Wyjątek stanowi numer sieci 0, który odnosi się do sieci wejściowej, czyli sieci z dostępnymi wszystkimi ulicami. Rysunek 5.27 przedstawia porównanie najlepszego rozwiązania uzyskanego podczas badań z rozwiążaniem najgorszym oraz z siecią wejściową.



Rys. 5.23: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 1, 2, 3, 4 z ujęciem sieci wejściowej - 0

Najbardziej interesującymi punktami na rysunku 5.23 jest sytuacja podczas pierwszych dwudziestu iteracji i ta zachodząca w końcowych iteracjach, osiemdziesiątej i kolejnych. Przede wszystkim zauważalna jest wyraźna przewaga sieci z zamkniętymi węzłami, występującą w początkowych iteracjach, która stopniowo maleje. Ważnym jest również, że podczas gdy wynik uzyskiwany w kolejnych iteracjach dla sieci wejściowej stopniowo poprawia się, zmiana ta nie jest tak widoczna w przypadku sieci uzyskanych podczas badań. W iteracji setnej, ostatniej sieć z numerem 4 (Rys. 5.5) uzyskuje gorszy wynik niż sieć wejściowa, natomiast wyniki pozostałych sieci (Rys. 5.2 — 5.4) są marginalnie lepsze.

5.4. ANALIZA ZASTOSOWANYCH USTAWIEŃ SYMULACJI

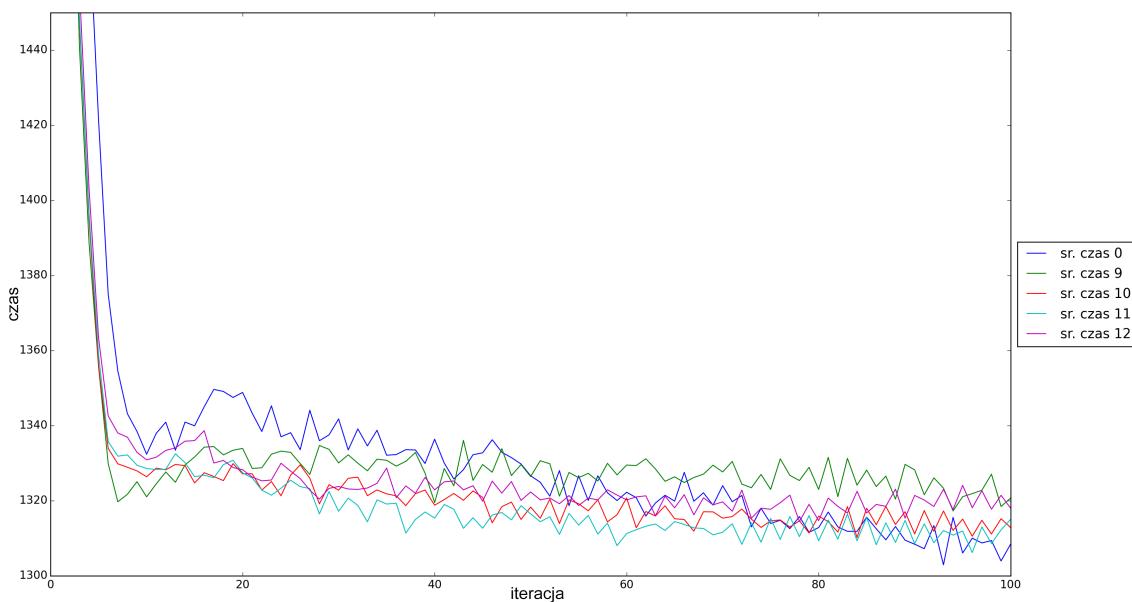


Rys. 5.24: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 5, 6, 7, 8 z ujęciem sieci wejściowej - 0

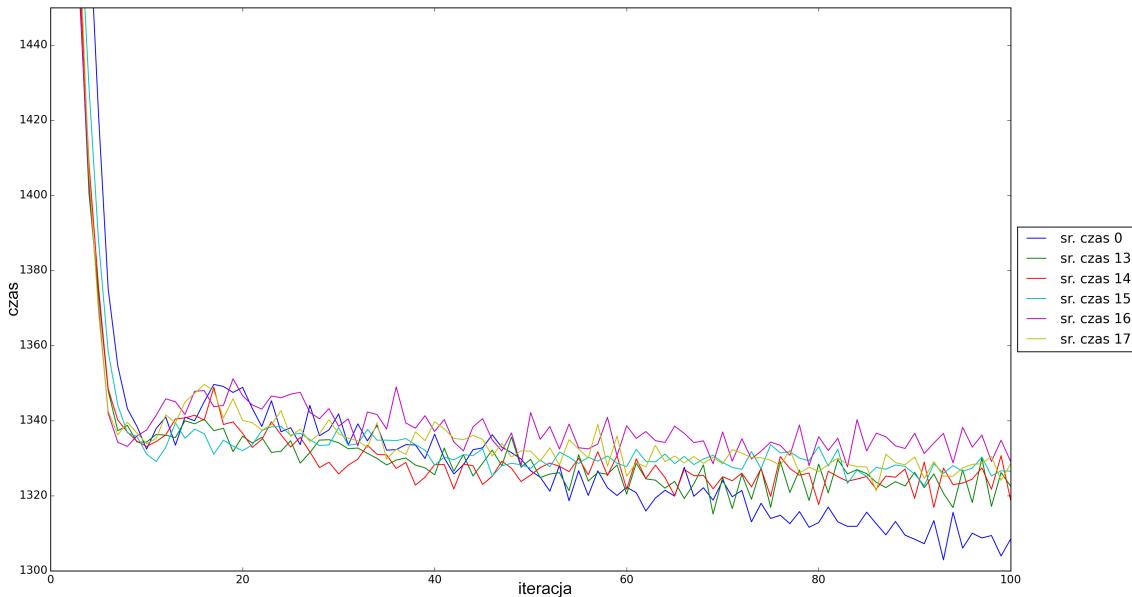
Nie jest zaskoczeniem, że w przypadku danych, pokazanych na rysunku 5.24 sytuacja powtarza się. Początkowo duża przewaga sieci uzyskanych podczas badań zaciera się. Iteracje ponownie nieznacznie wpływają na polepszanie się wyników uzyskanych przez rozwiązania o identyfikatorach: 5, 6, 7, 8 (Rys. 5.6 — 5.9). Wszystkie sieci oprócz sieci o identyfikatorze 6 (Rys. 5.7) w setnej iteracji okazują się jednak lepsze od sieci wejściowej, posiadającej wszystkie dostępne ulice (węzły). Wartym zauważenia faktem jest, że w przypadku dłuższej symulacji kolejność rozwiązań byłaby inna niż obecnie.

W przypadku rozwiązań o identyfikatorach: 9, 10, 11, 12 (Rys. 5.10 — 5.13) wyniki nie są już jednak tak obiecujące. Trend poprzednich wykresów utrzymuje się i w przypadku wykresu na rysunku 5.25 wyraźnie widać, że sieć wejściowa w setnej iteracji uzyskuje lepszy wynik niż uzyskane rozwiązania. Sytuacja jest jeszcze bardziej widoczna dla kolejnych rozwiązań: 13, 14, 15, 16 i 17 (Rys. 5.14 — 5.18), których wyniki jeszcze bardziej różnią się od tych uzyskanych przez sieć wejściową (Rys. 5.26). Wybór wyższego parametru iteracji symulacji *MATSim*, omawianej w podrozdziale 3.2, wpłynąłby zatem znacznie na liczbę otrzymanych sieci wynikowych, zmniejszając tę pulę o połowę.

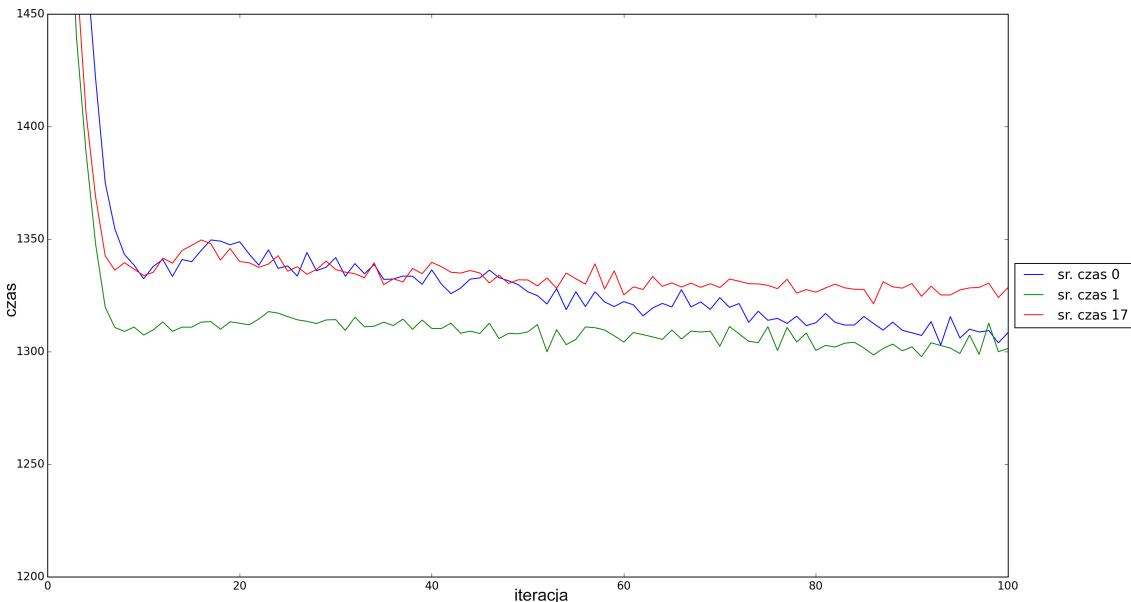
Wykresy prezentowane na rysunkach 5.23 — 5.26 doskonale podsumowuje wykres na rysunku 5.27, gdzie najlepsze rozwiązanie porównywane jest z najgorszym. Sieć wejściowa oraz zmiana wyniku pozwala wyciągnąć wnioski na temat działania samego symulatora.



Rys. 5.25: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 9, 10, 11, 12 z ujęciem sieci wejściowej - 0



Rys. 5.26: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 13, 14, 15, 16, 17 z ujęciem sieci wejściowej - 0



Rys. 5.27: Wykres zmiany średniego czasu przejazdu od iteracji dla sieci najlepszej i najgorszej (1, 17) z ujęciem sieci wejściowej - 0

Symulator *MATSim* zadziałał zgodnie z oczekiwaniemi, 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 powtarzającemu się w każdej iteracji **tym samym** warunkom drogowym, które panują w sieci. Scenariusz symulacji transportu ruchu drogowego jest bowiem pozbawiony losowych wypadków oraz czynników ludzkich. Nie jest to więc sytuacja rzeczywista.

Mając na uwadze ten „idealny” aspekt symulacji, wybory w iteracjach wcześniejszych są mniej idealne, można powiedzieć, bardziej ludzkie. Wykresy zaprezentowane na rysunkach 5.23 — 5.27 wyraźnie pokazują, że dla zmodyfikowanej sieci, znacznie trudniej jest popełnić błąd przy planowaniu trasy, mając ograniczoną wiedzę na temat warunków drogowych. Wymuszenie 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, trzeba bowiem pamiętać, że generalnie większa liczba węzłów oznacza większe możliwości przepływowe 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, zachodzi sytuacja odwrotna. Ich wiedza nie musiałaby sprawdzić się w przypadku zmiany ich planów dnia lub wystąpienia losowych

czynników na drodze.

W powyższym rozdziale przedstawiono wyniki potwierdzające możliwość optymalizacji sieci drogowej przy użyciu algorytmów genetycznych i paradoksu Braessa. Omówiono również konsekwencje użytych parametrów symulacji.

5.4. ANALIZA ZASTOSOWANYCH USTAWIEŃ SYMULACJI

Rozdział 6

Podsumowanie

Wyniki uzyskane w ramach niniejszej pracy dyplomowej potwierdzają możliwość zoptymalizowania sieci drogowej poprzez zamknięcie określonych ulic, przy użyciu algorytmu genetycznego. Dla zadanej sieci drogowej, poprzez optymalizację, uzyskano kilkanaście rozwiązań, które mogą posłużyć jako baza do dalszych badań. Zastosowana metoda oceny rozwiązań, czyli zewnętrzny symulator ruchu drogowego również spełnił oczekiwania projektowe. Trzeba jednak pamiętać, że uzyskane wyniki są czysto teoretyczne, a całość eksperymentu została przeprowadzona przy wielu uproszczeniach.

Pierwszym uproszczeniem jest oczywiście sam symulator. Po pierwsze, w założeniu został zbadany tylko jeden model populacji oraz aktywności dnia (agentów). Model ten przedstawia pewne uśrednienie demograficzne mieszkańców miasta Sioux Falls oraz ich planów komunikacyjnych. Po drugie, w badanym wariantie miasto zostało pozbawione komunikacji miejskiej, co miało na celu zintensyfikowanie ruchu transportu prywatnego. Po trzecie, symulator w sposób nierzeczywisty dokonuje optymalizacji planów dnia agentów, które mogą zostać uznane za nierzeczywiste. Trudnym jest jednak oszacowanie stopnia „sztuczności” dopasowania planu do spodziewanych warunków drogowych.

Drugim uproszczeniem jest model miasta. Z założenia stosowana jest sieć z pominięciem pomniejszych dróg, mało istotnych dla ogółu komunikacji w mieście. Kontynuując, model zakłada jedynie ruch, który jest generowany przez mieszkańców, z pominięciem ruchu napływającego z innych rejonów. Ostatnim są sytuacje losowe, spowodowane zwykle przez czynnik ludzki. Model nie przewiduje awarii, remontów, wypadków, które oczywiście są składnikiem codziennych sytuacji w mieście.

Podczas badań możliwe jest zastosowanie innego kryterium oceny rozwiązań. Zmiana symulatora lub zupełnie inna metoda oceny tworzonych sieci wpłynie bezsprzecznie na otrzymane wyniki.

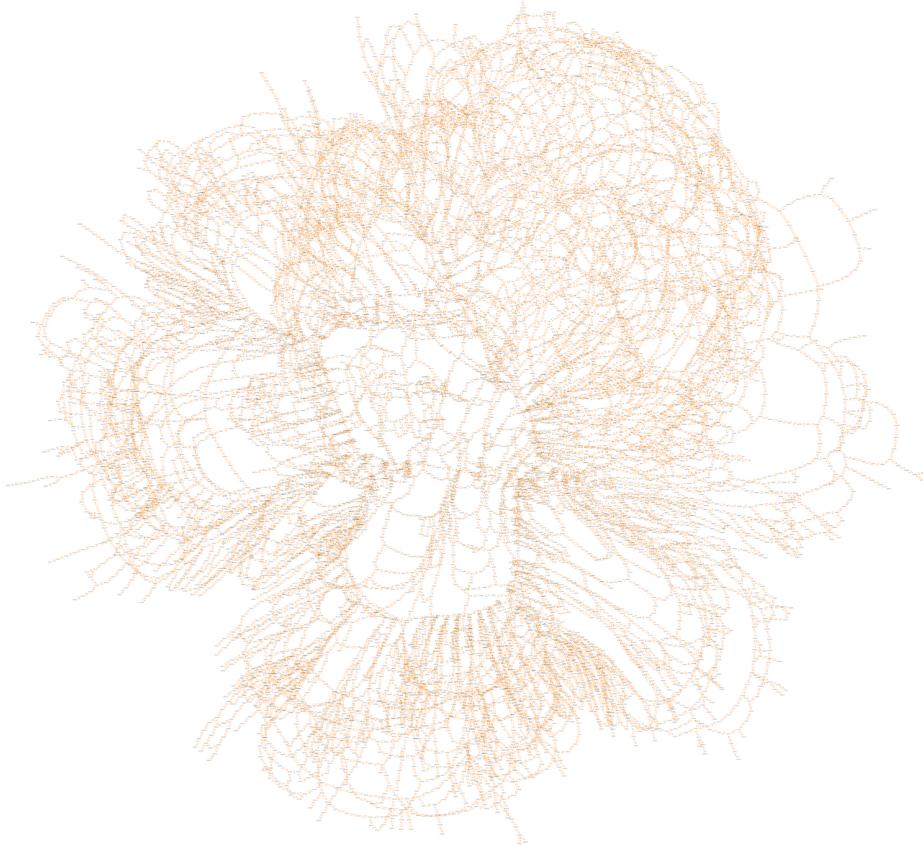
Pamiętając zatem, że prezentowane rozwiązanie miało dostarczyć pewnej sugestii w sprawie usprawnienia ruchu, spełnia ono swoje zadanie. Innowacja nawet tak dobrze

rozwiniętych miast, jak te w USA jest możliwa, niekoniecznie poprzez kosztowne inwestycje w tworzenie nowej infrastruktury.

6.1 Perspektywy dalszych badań w dziedzinie

Dość oczywistym kierunkiem rozwoju może być oczywiście dalsze przeszukiwanie opisanego problemu. Jak już zostało wspomniane wcześniej, ilość możliwych rozwiązań jest bardzo duża (2^{90}), a ponadto, zmiana parametrów samej symulacji, również wpłynie na otrzymywane wyniki.

Ciekawszym wydaje się być jednak przeszukiwanie nowych zbiorów. W przypadku symulatora *MATSim* istnieje pokaźna liczba przykładów opartych o rzeczywiste miasta, na których były prowadzone symulacje. Jeden z dostępnych modeli jest odwzorowaniem miasta Berlin (Rys. 6.1), wykonanym z dużo większą dokładnością niż dla miasta Sioux Falls.



Rys. 6.1: Sieć drogowa miasta Berlin w postaci grafu

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

Spis rysunków

2.1	Schemat przykładowego wyjściowego układu drogowego	10
2.2	Schemat przykładowego uzupełnionego układu drogowego	12
2.3	Ogólny schemat algorytmu genetycznego	15
2.4	Ogólny schemat operacji krzyżowania osobników	17
2.5	Ogólny schemat operacji mutacji osobników	17
2.6	Schemat selekcji osobników metodą koła ruletki	18
2.7	Mapa królewieckich mostów z czasów Eulera	21
2.8	Schemat przekształcenia problemu mostów królewieckich w graf	21
2.9	Przykładowy graf nieskierowany	22
2.10	Przykładowy graf skierowany	23
2.11	Przykładowy graf z oznaczeniem kolejności odwiedzonych węzłów przy przeszukiwaniu metodą DFS	25
2.12	Przykładowy graf z oznaczeniem kolejności odwiedzonych węzłów przy przeszukiwaniu metodą BFS	25
2.13	Przykładowy graf z zaznaczonymi składowymi silnie spójnymi	26
2.14	Fragment sieci drogowej miasta Sioux Falls, Południowa Dakota	27
2.15	Sieć drogowa miasta Sioux Falls w postaci grafu	28
2.16	Graf sieci drogowej miasta z dopasowaną geometrią	28
2.17	Fragment sieci drogowej w postaci tablicy binarnej	29
2.18	Fragment sieci drogowej w postaci grafu	29
3.1	Schemat etapów symulacji <i>MATSim</i>	33
3.2	Graf czasu trwania symulacji <i>MATSim</i> dla stu iteracji	37
3.3	Graf średniego czasu przejazdów agentów dla stu iteracji symulacji	38
3.4	Graf wyników agentów dla stu iteracji symulacji	38
3.5	Graf sieci prezentowanego miasta Sioux Falls	40
3.6	Rozkład budynków na grafie miasta Sioux Falls	40
3.7	Natężenie ruchu w mieście Sioux Falls w godzinach 6.00 - 7.00	41
3.8	Natężenie ruchu w mieście Sioux Falls w godzinach 16.00 - 17.00	41

SPIS RYSUNKÓW

4.1	Diagram wybranych klas pakietu <i>Apache Math Genetics</i> , część 1	44
4.2	Diagram wybranych klas z pakietu <i>Apache Math Genetics</i> , część 2	45
4.3	Dostępne konfiguracje warstwy podstawowej chmury Microsoft Azure . .	46
4.4	Schemat działania aplikacji <i>GRoNO</i>	51
5.1	Wykres zmian najlepszego wyniku iteracji	57
5.2	Sieć miasta Sioux Falls, rozwiązanie nr 1	59
5.3	Sieć miasta Sioux Falls, rozwiązanie nr 2	60
5.4	Sieć miasta Sioux Falls, rozwiązanie nr 3	60
5.5	Sieć miasta Sioux Falls, rozwiązanie nr 4	61
5.6	Sieć miasta Sioux Falls, rozwiązanie nr 5	61
5.7	Sieć miasta Sioux Falls, rozwiązanie nr 6	62
5.8	Sieć miasta Sioux Falls, rozwiązanie nr 7	62
5.9	Sieć miasta Sioux Falls, rozwiązanie nr 8	63
5.10	Sieć miasta Sioux Falls, rozwiązanie nr 9	63
5.11	Sieć miasta Sioux Falls, rozwiązanie nr 10	64
5.12	Sieć miasta Sioux Falls, rozwiązanie nr 11	64
5.13	Sieć miasta Sioux Falls, rozwiązanie nr 12	65
5.14	Sieć miasta Sioux Falls, rozwiązanie nr 13	65
5.15	Sieć miasta Sioux Falls, rozwiązanie nr 14	66
5.16	Sieć miasta Sioux Falls, rozwiązanie nr 15	66
5.17	Sieć miasta Sioux Falls, rozwiązanie nr 16	67
5.18	Sieć miasta Sioux Falls, rozwiązanie nr 17	67
5.19	Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wszystkich wyników sieci drogowych	69
5.20	Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wyników sieci drogowych o ID: 5, 7, 9, 10, 11, 12, 13, 14, 16	70
5.21	Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wyników sieci drogowych o ID: 2, 4, 6, 8, 12	71
5.22	Fragment grafu z zaznaczonym zamkniętym obszarem wspólnie dla wyników sieci drogowych o ID: 3, 8, 15	72
5.23	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 1, 2, 3, 4 z ujęciem sieci wejściowej - 0	73
5.24	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 5, 6, 7, 8 z ujęciem sieci wejściowej - 0	74
5.25	Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 9, 10, 11, 12 z ujęciem sieci wejściowej - 0	75

5.26 Wykres zmiany średniego czasu przejazdu od iteracji dla sieci wynikowych o ID 13, 14, 15, 16, 17 z ujęciem sieci wejściowej - 0	75
5.27 Wykres zmiany średniego czasu przejazdu od iteracji dla sieci najlepszej i najgorszej (1, 17) z ujęciem sieci wejściowej - 0	76
6.1 Sieć drogowa miasta Berlin w postaci grafu	80

SPIS RYSUNKÓW

Spis tabel

5.1 Tabelaryczna reprezentacja otrzymanych najlepszych sieci drogowych . . . 58

Spis listingów

3.1	Przykładowy plik populacji scenariusza zawierający plany dnia agentów .	35
4.1	Plik konfiguracyjny projektu GRoNO	47
5.1	Ustawienia algorytmu genetycznego użytego podczas badań	55

SPIS LISTINGÓW

Bibliografia

- [1] D. Rutkowska, M. Piliński i L. Rutkowski, *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*, PWN, Warszawa 1997.
- [2] Michalewicz Z., *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, Wydawnictwo Naukowo-Techniczne, Warszawa 1999.
- [3] S. T. Wierzchoń, *Sztuczne systemy immunologiczne. Teoria i zastosowania*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001
- [4] Mitchell Melanie, *An Introduction to Genetic Algorithms*, A Bradford Book The MIT Press, 1998.
- [5] Leslie Arthur Keith Bloy, *An investigation into Braess' paradox*, Thesis, 02/2007.
- [6] Dietrich Braess, *Über ein Paradoxon aus der Verkehrsplanung. „Unternehmensforschung”*, 1968 (niem.).
- [7] A. Nagurney, and T. Wakolbinger, *On a Paradox of Traffic Planning, translated from the (1968) original D. Braess paper from German to English by D. Braess*, Transportation Science 39/4, 2005.
- [8] Ric Pas and Shari Principio, *Braess' paradox: Some new insights*, April 1996.
- [9] Richard Steinberg and Willard I. Zangwill, *The Prevalence of Braess' Paradox*, Transportation Science Vol. 17, No. 3, August 1983.
- [10] Łukasz Kowalik, *Algorytmy i struktury danych, grafy*, Wykład, 2003.
- [11] Robin J. Wilson, *Introduction to Graph Theory*, 5th edition, Pearson, 2010.
- [12] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*, North Holland, 5th printing, 1982.
- [13] A. Chakirov, *Enriched Sioux Falls Scenario with Dynamic Demand*, MATSim User Meeting, Zurich/Singapore, June 2013.

BIBLIOGRAFIA

- [14] 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 October 30, 2014.
- [15] Ana L. C. Bazzan and Franziska Klügl, *Reducing the Effects of the Braess Paradox with Information Manipulation*.
- [16] 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.
- [17] Tarjan, R. E., *Depth-first search and linear graph algorithms*, SIAM Journal on Computing, Vol 1, No. 2, 06.1972
- [18] Polskie tłumaczenie paradoksu Braessa, http://pl.wikipedia.org/wiki/Paradoks_Braessa, dostęp 10.04.2015.
- [19] Marek Karabon, Kontr-intuicyjne metody ograniczania korków w miastach, http://www.tnn.pl/k_675_m_3.html, dostęp 10.04.2015.
- [20] Wojciech Szymalski: Prawo Lewisa-Mogridge'a w Warszawie – wprowadzenie, http://www.zm.org.pl/?a=lewis-mogridge-14-00_wprowadzenie, dostęp 10.04.2015.
- [21] Matematyczne twierdzenie grafu silnie spójnego, <http://mathworld.wolfram.com/StronglyConnectedDigraph.html>, dostęp 10.04.2015.
- [22] Strona główna projektu MATSim, <http://matsim.org>, dostęp 10.04.2015.
- [23] Strona główna projektu Apache Commons Math, <http://commons.apache.org/proper/commons-math>, dostęp 10.04.2015.
- [24] Strona główna projektu NetworkX, <http://networkx.github.io>, dostęp 10.04.2015.
- [25] Strona główna systemu Linux Ubuntu, <http://www.ubuntu.com>, dostęp 10.04.2015.
- [26] Strona główna chmury Microsoft Azure, <http://azure.microsoft.com>, dostęp 10.04.2015.
- [27] Max Planck Institute, *Study: Solar and wind energy may stabilize the power grid*, R&D Magazine, 14.09.2012
- [28] M. Pala, S. Baltazar, P. Liu, H. Sellier, B. Hackens, F. Martins, V. Bayot, X. Wallart, L. Desplanque, S. Huant, *Transport Inefficiency in Branched-Out Mesoscopic Networks: An Analog of the Braess Paradox*, Physical Review Letters 108, 06.12.2011.

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. This thesis covers two main fields of presented 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. This provides a wide choice of options for future work or similar experiments. In the thesis, a multiagent simulator, *MATsim* is used. The aspects of the simulation concerning the final results of the project are also described.

The solution is prepared using Java and Python programming languages. 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, as well as the simulation based rank using the multiagent simulator resulted in several 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.

Łódź, dnia 10.04.2015 roku

Michał Siatkowski

ul. Tramwajowa 21 m.10
90-132 Łódź

186865

Wydział Fizyki Technicznej
Informatyki i Matematyki
Stosowanej

Informatyka

II stopnia, studia stacjonarne

Oświadczenie

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedstawiona praca licencjacka / inżynierska / magisterska¹ na temat

Optymalizacja struktury sieci drogowej
Optimization of structures of road networks

została napisana przeze mnie samodzielnie.

Jednocześnie oświadczam, że ww. praca

- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i praw pokrewnych (j.t. Dz. U. z 2006 r. Nr 90, poz. 631, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem dyplomów wyższej uczelni lub tytułów zawodowych.

.....
(Czytelny podpis Studenta)

¹ Niepotrzebne skreślić

Łódź, dnia 10.04.2015 roku

Michał Siatkowski

ul. Tramwajowa 21 m.10
90-132 Łódź

186865

Wydział Fizyki Technicznej
Informatyki i Matematyki
Stosowanej

Informatyka

II stopnia, studia stacjonarne

Oświadczenie

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedstawiona na nośniku CD/DVD praca licencjacka / inżynierska / magisterska¹ na temat

Optymalizacja struktury sieci drogowej
Optimization of structures of road networks

zawiera te same treści, co oceniany przez Opiekuna pracy i Recenzenta wydruk komputerowy.

.....
(Czytelny podpis Studenta)

¹ Niepotrzebne skreślić

