Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Paweł Gora
Nr albumu: 219859

Adaptacyjne planowanie ruchu drogowego

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
prof. dra hab. Andrzeja Skowrona
Instytut Matematyki

Wrzesień 2010

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

W pracy przedstawione zostały metody adaptacyjnego planowania ruchu drogowego oparte na algorytmie genetycznym. Ich skuteczność przetestowana została przy użyciu symulatora ruchu drogowego TSF (Traffic Simulation Framework). Opisana została również architektura samego symulatora oraz techniczne aspekty jego implementacji przy użyciu technologii .NET Framework.

Słowa kluczowe

systemy inteligentne, modelowanie złożonych procesów, symulacja ruchu drogowego, automaty komórkowe

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.0 Matematyka, Informatyka:
11.4 Sztuczna inteligencja

Klasyfikacja tematyczna

68T99

Tytuł pracy w języku angielskim

Adaptive planning of vehicular traffic

Podziękowania

Pragnę złożyć serdeczne podziękowania dla Profesora Andrzeja Skowrona, który kierował moją pracą, wspierał mnie i stanowił nieocenioną pomoc zarówno od strony merytorycznej, jak i mentalnej - dopingując mnie, motywując i wspierając w trudnych chwilach, które towarzyszyły mi podczas powstawania pracy.

Dziękuję mojej Rodzinie i Przyjaciołom, którzy wspierali mnie zarówno podczas tworzenia tej pracy, jak i podczas całych studiów na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego.

Serdeczne podziękowania składam również wszystkim innym osobom, które przyczyniły się do powstania mojej pracy, wspierając mnie cennymi wskazówkami, dyskusjami, konstruktywnymi uwagami, okazując zainteresowanie moją pracą i dając poczucie, że to co robię jest ważne. Przede wszystkim byli to uczestnicy seminarium badawczego Zakładu Logiki Matematycznej: dr hab. Hung Son Nguyen, dr Jan Bazan, dr Marcin Szczuka, Marcin Wojnarski, Mateusz Adamczyk, Paweł Betliński, Sebastian Stawicki i Andrzej Janusz.

Dziękuję również moim kolegom ze studiów: Pawłowi Brachowi, Krzysztofowi Choromańskiemu, Alkowi Jankowskiemu i Bartkowi Łosiowi, którzy współtworzyli ze mną system inteligentnej nawigacji UTCS. Stał się on bazą i inspiracją do moich dalszych prac nad programem TSF opisanym w niniejszej pracy.

Wprowadzenie

Ruch drogowy jest bardzo złożonym i trudnym w opisie zjawiskiem. Uczestniczy w nim wielu agentów (kierowców prowadzących pojazdy) podejmujących niezależne decyzje i wykonujących akcje zmierzające do realizacji określonego celu (dotarcie do punktu przeznaczenia) przy użyciu dostępnych środków (sieć drogowa, możliwości pojazdu) i przestrzegających określonych reguł (przepisy ruchu drogowego). Opis jest jeszcze trudniejszy, gdy akcje te są wykonywane w zmieniającym się środowisku (warunki atmosferyczne) oraz gdy ustalone reguły są przez kierowców łamane.

Rozwój technologiczny spowodował, że coraz łatwiej i szybciej produkuje się pojazdy, a ich ilość na drogach zwiększa się w bardzo szybkim tempie (Tablica 1 ilustruje dynamikę wzrostu ilości pojazdów w Polsce). Prowadzi to do dużego problemu cywilizacyjnego jakim stały się korki drogowe. Problem ten jest szczególnie widoczny w dużych aglomeracjach, gdzie duże natężenie ruchu jest powodem zanieczyszczenia środowiska oraz trudności organizacyjnych. Z tego powodu bardzo istotne stały się badania nad zjawiskiem ruchu drogowego i próby matematycznego opisu tego zjawiska.

Tablica 1: Wykaz ilości pojazdów w Polsce w ostatnich latach (na podstawie [GUS])

	2000	2005	2006	2007
Ilość pojazdów (w tys. sztuk)	14106	16816	18035	19472

W pracy [GOR] ująłem najważniejsze otrzymane dotychczas rezultaty związane z modelowaniem ruchu drogowego: metody oparte na analogiach do znanych, dobrze zbadanych zjawisk i równań fizycznych (przepływ cieczy, dynamika Newtona, prawo Maxwella, kinetyczna teoria gazów) oraz na komputerowych symulacjach ruchu drogowego. Okazuje się, że stosując drugie podejście uzyskuje się bardzo dobry opis zjawiska ruchu drogowego, szczególnie w przypadku ruchu po prostych odcinkach i autostradach. Najskuteczniejsze modele symulacji oparte są na modelu Nagela-Schreckenberga ([NS]), który zostanie przedstawiony w rozdziale 1.1. Opiera się na nim również model TSF (opisany w rozdziale 1.2) wykorzystany w symulatorze TSF. Implementacja modelu w postaci programu komputerowego oraz wszystkie przydatne funkcjonalności symulatora zostaną przedstawione w rozdziale 2. Symulator został użyty przeze mnie do testowania metod adaptacyjnego planowania ruchu drogowego. Metoda oparta na algorytmie genetycznym będzie przedstawiona w rozdziale 3.

Zagadnienia konstruowania symulatorów złożonych systemów dynamicznych jak również eksploracji danych generowanych przez takie systemy, np. dla wspomagania adaptacyjnego sterowania obiektami w takich systemach, należą do bardzo aktualnych problemów badanych przez wiele ośrodków na świecie oraz firm komputerowych (np. [OTC], [RPS], [SCATS], [SIE], [TELVENT], [TLC], [TRB], [WU]). Zagadnienia te należą do bardzo trudnych i stanowią one wyzwania dla naukowców (np. [AGG], [AL], [BU], [GS], [HL], [HLL], [MBY], [MG], [PRTB], [RPBM], [TSS], [WI], [WJK]).

Niniejsza praca jest krokiem w kierunku opracowania skalowalnych algorytmów adaptacyjnych dla sterowania ruchem ulicznym, i w szerszym sensie dla adaptacyjnego sterowania złożonymi systemami dynamicznymi z wykorzystaniem symulatorów i w dalszej perspektywie wiedzy dziedzinowej reprezentowanej najczęściej w języku naturalnym (np. informacje o aktualnej sytuacji meteo-

rologicznej lub robotach drogowych).

Badania przedstawione w pracy były częściowo wspierane finansowo przez grant N N516 368334 Ministerstwa Nauki i Szkolnictwa Wyższego.

Rozdział 1

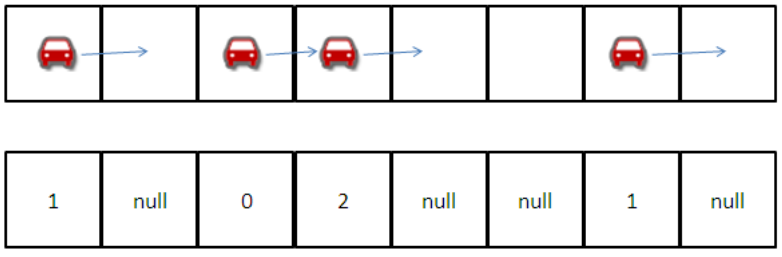
Modele symulacji ruchu drogowego

Współcześnie najskuteczniejsze modele symulacji ruchu drogowego opierają się na teorii automatów komórkowych i modelu Nagela-Schreckenberga, który zostanie opisany w następnym podrozdziale.

1.1. Model Nagela-Schreckenberga

Model Nagela-Schreckenberga (w skrócie: model NaSch) służy do symulacji ruchu pojazdów na prostym odcinku drogi. Jest to probabilityczny automat komórkowy, w którym droga jest reprezentowana jako cykliczna taśma podzielona na komórki, a pojęcia odległości, czasu i prędkości przyjmują wartości dyskretne. Każda komórka może być albo zajęta (znajduje się na niej dokładnie jeden samochód), albo pusta. Ruch na taśmie odbywa się w jednym kierunku i w sposób cykliczny, pojazdy są nierozróżnialne i mają taką samą maksymalną prędkość V_{MAX}. Każdy pojazd w dowolnym momencie posiada dyskretną prędkość ze zbioru $\{0, 1, ..., V_{MAX}\}$. W oryginalnym modelu ([NS]) szerokość pojedynczej komórki była równa $7,5$ metra, a $V_{MAX} = 5$, co odpowiada prędkości $135\frac{km}{h}$.

Ewolucja automatu odbywa się zgodnie z ustalonymi *regułami ruchu*: każdy z pojazdów i przemieszcza się na taśmie do przodu o ilość pól odpowiadającą wartości prędkości V_i obliczonej zgodnie z tą regułą.



Rysunek 1.1: Automat komórkowy w modelu Nagela-Schreckenberga

Reguła ruchu dla każdego pojazdu jest czteroetapowa:

1. $V_i := min(V_{MAX}, V_i + 1)$
2. $V_i := max(V_i, d_i)$, gdzie d_i jest odległością (ilością wolnych komórek) pojazdu i od najbliższego pojazdu przed nim
3. Z prawdopodobieństwem $\mathbf{p} > 0$ $V_i := max(V_i - 1, 0)$

4. Pojazd i przemieszcza się do przodu o V_i komórek.

Kolejne etapy reguły ruchu odpowiadają rzeczywistym sytuacjom, które zachodzą w trakcie jazdy. Krok pierwszy symuluje, że każdy z kierowców stara się jechać najszybciej jak może, bo chce jak najszybciej dotrzeć do celu. Jednocześnie musi zachować bezpieczeństwo na drodze, czyli zmniejsza prędkość w sytuacji, gdy pojazd przed nami jest zbyt blisko (etap drugi). Trzeci krok decyduje o samoistnym powstawaniu korków na drodze [SCH]. Jest to jedyny probabilistyczny element reguły przejścia. Odpowiada on różnym sytuacjom na drodze, w których kierowcy nie zawsze jeżdżą z maksymalną możliwą prędkością (np. z powodu warunków atmosferycznych, nachylenia terenu lub względów psychologicznych). W kroku czwartym pojazd przemieszcza się z wyznaczoną prędkością.

Model NaSch okazał się być bardzo realistyczny dla ruchu drogowego po prostym odcinku, co uzasadniono m.in. w pracy [GOR]. Z tego powodu jest on do dziś podstawą wielu innych modeli symulacji ruchu drogowego.

1.2. Model TSF

Model TSF podobnie jak model NaSch opiera się na teorii automatów komórkowych. Stanowi on jednak istotne rozszerzenie tego modelu, aby umożliwiać przeprowadzanie symulacji ruchu na rzeczywistych sieciach drogowych i rzeczywistych danych. Uwzględnia więc m.in. takie elementy jak:

- skrzyżowania
- rozróżnialność kierowców
- rozróżnialność typów dróg
- światła drogowe
- wielopasmowość dróg

Szczegóły modelu zostały opisane w pracy [GOR]. Tu przytoczymy jedynie najważniejsze jego założenia.

Sieć drogowa w modelu TSF jest reprezentowana jako graf skierowany $G = (V, E)$, gdzie V to zbiór wierzchołków grafu, a $E \subseteq V \times V$ jest zbiorem jego krawędzi. Krawędzie grafu reprezentują odcinki rzeczywistej sieci drogowej i na nich odbywa się ruch - pojazdy przemieszczają się pomiędzy wierzchołkami grafu. Trasy przejazdu pojazdów są zatem ścieżkami w grafie G.

Każda krawędź grafu posiada określoną długość - odległość między wierzchołkami na końcach tej krawędzi. Ponadto każda krawędź należy do określonej kategorii drogi (w oryginalnym modelu z pracy [GOR] występują 4 kategorie dróg). Każdą kategorię cechuje ustalona ilość pasów ruchu, średnia maksymalna prędkość jazdy pojazdów na tej drodze oraz odchylenie standardowe tej średniej prędkości.

W niektórych wierzchołkach grafu zlokalizowane są światła drogowe. Lokalizacja pojedynczej sygnalizacji jest jednoznacznie wyznaczona przez krawędź, na której końcu znajdują się światła.

Odcinki dróg (krawędzie) mogą mieć kilka pasów ruchu. Każdy z pasów ruchu jest reprezentowany jako skończona taśma podzielona na komórki, które będą wchodziły w skład automatu komórkowego. W każdej chwili ewolucji modelu pojedyncza komórka może być pusta lub zajęta przez 1 pojazd. Ewolucja odbywa się w dyskretnym czasie zgodnie z ustalonymi regułami ruchu.

W obrębie danej krawędzi pasy ruchu są numerowane liczbami naturalnymi. W obrębie pojedynczego pasa, komórki również są numerowane. Każda komórka jest więc jednoznacznie wyznaczona przez 3 parametry:

- *edge* - krawędź, na której znajduje się dana komórka
- *lane* - numer pasa ruchu (taśmy) na krawędzi, na której znajduje się komórka
- *cell* - numer komórki na tym pasie (taśmie).

Wszystkie komórki na tej samej krawędzi mają jednakową długość i wszystkie pasy ruchu na tej krawędzi są podzielone na tyle samo komórek o takiej samej długości. Długości komórek na różnych krawędziach grafu mogą się różnić z uwagi na fakt, że łączna długość krawędzi jest wielokrotnością długości jej komórek.

Jeżeli pas ruchu na odcinku reprezentowanym w grafie przez krawędź e ma długość $distance(e)$, to pas ten jest podzielony na $number(e) = \left\lceil \frac{distance(e)}{D} \right\rceil$ komórek, gdzie D - domyślna długość komórki w modelu. Długość tych komórek to $d(e) = \frac{distance(e)}{number(e)}$.

1.2.1. Sygnalizacja świetlna

Bardzo ważnym czynnikiem wpływającym na ruch drogowy jest obecność sygnalizacji świetlnej na skrzyżowaniach.

W modelu TSF sygnalizacja świetlna jest obiektem zlokalizowanym w wierzchołku grafu i charakteryzowanym następującymi atrybutami:

- *id* - identyfikator sygnalizacji
- $position$ - lokalizacja sygnalizacji (krawędź, na końcu której znajduje się sygnalizacja)
- t_{green} - czas trwania fazy światła zielonego
- t_{red} - czas trwania fazy światła czerwonego
- t_{change} - ilość jednostek czasu, po których ma nastąpić zmiana fazy
- $state \in \{GREEN, RED\}$ - aktualny stan/faza sygnalizacji

Wartość atrybutu t_{change} jest liczbą nieujemną i zmniejsza się o 1 w każdym kroku symulacji. Gdy osiąga 0, następuje zmiana fazy sygnalizacji i atrybut ten przyjmuje wartość t_{green} lub t_{red} w zależności od aktualnej fazy.

1.2.2. Pojazdy

Zbiór wszystkich możliwych pojazdów w modelu TSF oznaczamy jako CARS. Pojazd w modelu to obiekt posiadający następujące atrybuty:

- *id* - identyfikator pojazdu
- *edge* - krawędź w grafie, na której znajduje się pojazd
- $distance$ - odległość pojazdu od początku krawędzi, na której się znajduje
- id_{lane} - numer pasa ruchu, na którym jest pojazd
- id_{cell} - numer komórki zajmowanej przez pojazd
- $profile$ - profil kierowcy
- *velocity* - prędkość pojazdu

- *path* - trasa przejazdu
- *start* - ilość jednostek czasu od początku symulacji, po których pojazd rozpoczyna swój ruch.

Jedynym atrybutem, który ma nieoczywistą semantykę jest atrybut $profile$. Każdy kierowca posiada pewną stałą wartość tego atrybutu losowaną z rozkładu normalnego $\mathcal{N}(0,1)$. Wpływa ona na maksymalną prędkość, z jaką ten kierowca chciałby jechać. Szczegóły znajdują się w pracy [GOR].

1.3. Model TSF jako automat komórkowy

Deterministyczny automat komórkowy to sieć komórek, które mogą znajdować się w jednym z możliwych stanów, a stan każdej komórki podlega ewolucji w dyskretnym czasie zgodnie z określoną regułą, zależną jedynie od aktualnego stanu komórki i komórek z jej otoczenia [CA]. Jeżeli reguła ta zależy od zmiennej losowej, to jest to automat *probabilistyczny*.

Model TSF korzysta z nieco uogólnionej definicji automatu komórkowego:

Definicja 1.3.1. ***Automat komórkowy*** *to krotka:*

$$CA =< T, C, N, S, S_0, F >, \tag{1.1}$$

gdzie:

- T *- Przedział czasu, w którym odbywa się ewolucja automatu (*$T = \{0, 1, 2, \ldots, T_{MAX}\}$*, gdzie* $T_{MAX} \in \mathbb{N} \cup \{\infty\}$*)*
- C *- Zbiór komórek*
- $N : C -> \mathcal{P}(C)$ *- Funkcja, która każdej komórce ze zbioru C przyporządkowuje jej otoczenie*
- S *- Zbiór możliwych stanów komórek*
- $S_0 : C -> S$ *- Początkowa konfiguracja komórek (stan komórek w chwili* $t = 0$*)*
- $F : T \times C -> S$ *- Reguła przejścia, taka że* $\forall_{c \in C} \forall_{t \in T}\ c_{t+1} = F(t, c)$*, gdzie* c_t *- stan komórki c w chwili* $t \in T$*.*

Zgodnie z dotychczasowym opisem ruch pojazdów w modelu TSF odbywa się w grafie sieci drogowej $G = (V, E)$, którego krawędzie podzielone są na komórki, a każda komórka może być albo zajęta (jest na niej pojazd), albo pusta. Wszystkie te komórki (oznaczmy je jako $CELLS$) należą również do zbioru komórek automatu komórkowego. Dodatkowo w zbiorze komórek są również komórki reprezentujące sygnalizację świetlną (one również podlegają ewolucji w dyskretnym czasie). Oznaczmy ten zbiór jako $SIGNALS$.

Zatem zbiór komórek C automatu komórkowego TSF to

$$C = CELLS \cup SIGNALS,$$

O ruchu pojedynczego pojazdu decyduje przede wszystkim obraz otoczenia rejestrowany przez jego kierowcę, ale kierowcy mogą również odbierać sygnały dotyczące globalnej sytuacji w ruchu drogowym (może to odpowiadać np. informacjom radiowym, wskazaniom inteligentnych programów nawigacyjnych). Kierowcy reagują również na światła drogowe. Zatem

$$\forall_{c \in CELLS} \quad N(c) = C,$$

Stan komórek reprezentujących sygnalizację świetlną w chwili $t+1$ zależy jedynie od parametrów tej sygnalizacji w chwili t (czyli parametrów takich jak T_{green}, T_{red}, T_{delay}). W modelu TSF zakładamy, że wartości tych parametrów mogą być adaptacyjnie ustawiane tak, aby optymalizować globalny ruch, więc mogą być modyfikowane w zależności od globalnego stanu ruchu. W związku z tym:

$$\forall_{c \in SIGNALS} \quad N(c) = C.$$

Zatem:

$$\forall_{c \in C} \quad N(c) = C.$$

Komórki znajdujące się na krawędziach grafu G mogą być zajmowane tylko przez 1 pojazd lub być puste. Jeżeli w komórce znajduje się pojazd, to stanem tej komórki jest ten pojazd. Jeżeli komórka jest pusta, to jej stanem jest *null*.

W przypadku komórek reprezentujących sygnalizację świetlną, stan komórki jest fazą tej sygnalizacji. Zatem:

$$S = \{null\} \cup CARS \cup \{GREEN, RED\}$$

Zdefiniowanie początkowego stanu automatu polega na podaniu początkowych wartości jego komórkom. W przypadku komórek odpowiadających sygnalizacji świetlnej stanem początkowym może być dowolny element ze zbioru $\{GREEN, RED\}$. Konfiguracja początkowa komórek ze zbioru $CELLS$ to rozmieszczenie pojazdów, które rozpoczynają ruch wraz z początkiem symulacji (dla których atrybut *start* ma wartość 0) na odpowiednich komórkach z $CELLS$. Metoda inicjowania pojazdów rozpoczynających ruch została szczegółowo opisana w pracy [GOR].

Jedyną rzeczą wymagającą wyjaśnienia jest reguła przejścia, czyli funkcja $F : T \times C- > S$, określająca w jaki sposób stan komórki w kroku $t + 1$ zależy od stanu w kroku t:

$$\forall_{c \in C, t \in T} \quad c_{t+1} = F(t, c).$$

Dla komórek ze zbioru $SIGNALS$ regułę przejścia można wyrazić następującym algorytmem:

Algorithm 1 Algorytm przejścia dla sygnalizacji świetlnej s w kroku t

Require: $s \in SIGNALS(G), t \in T$
Ensure: $state(s, t + 1) \in \{GREEN, RED\}$
  if $t_{change}(s) > 0$ **then**
    $t_{change}(s) := t_{change}(s) - 1$
    $state(s, t + 1) := state(s, t)$
    return $state(s, t + 1)$
  else
    if $state(s, t) = RED$ **then**
      $t_{change}(s) := t_{green}(s)$
      $state(s, t + 1) := GREEN$
    else
      $t_{change}(s) := t_{red}(s)$
      $state(s, t + 1) := RED$
    end if
    return $state(s, t + 1)$
  end if

Reguła przejścia dla komórek z $CELLS$ to po prostu reguła poruszania się pojazdów po drodze. W regule tej poza atrybutami pojazdów brane są pod uwagę nastepujące parametry:

- $turnPenalty \in [0, 1]$ - parametr odpowiedzialny za zachowanie pojazdu przed zakręcaniem
- $crossroadPenalty \in [0, 1]$ - parametr odpowiedzialny za zachowanie pojazdu w trakcie przejazdu przez skrzyżowanie (bez skręcania)
- $prob \in [0, 1]$ - parametr odpowiedzialny za losową redukcję prędkości pojazdu.

Regułę ruchu pojazdu można wyrazić następującym algorytmem:

Algorithm 2 Algorytm ruchu pojazdu *car* w kroku t

Require: $G = (V, E)$, $t \in T$, $car \in CARS(t)$, $turnPenalty$, $crossroadPenalty$, $prob$

```
increaseVelocity(car, t);
if stopOnSignal(car, t) then
  reduceVelocityOnSignal(car, t)
else
  if turnOnCrossroad(car, t) then
    reduceVelocity(car, t, turnParameter);
  else
    if crossroad(car, t) then
      reduceVelocity(car, t, crossroadParameter);
    end if
  end if
end if
if shouldChangeLane(car, t) then
  changeLane(car, t);
end if
safeReduceVelocity(car, t)
with probability prob: reduceVelocity(car, t);
makeMove(car, t);
```

Działa on następująco:

- W procedurze $increaseVelocity(car, t)$ pojazd stara się przyspieszyć nie przekraczając maksymalnej prędkości odpowiedniej dla danego kierowcy.
- Jeżeli pojazd dojechałby w tym kroku do skrzyżowania ze światłami i miałby czerwone światło (procedura $stopOnSignal(car, t)$), to w procedurze $reduceVelocityOnSignal(car, t)$ prędkość pojazdu jest redukowana, aby mógł on zatrzymać się przed światłami.
- Jeżeli kierowca nie musi zatrzymywać się na światłach na skrzyżowaniu, ale skręca na skrzyżowaniu ($turnOnCrossroad(car, t)$), to w procedurze $reduceVelocity(car, t, turnPenalty)$ odbywa się odpowiednia redukcja jego prędkości.
- Jeżeli kierowca przejeżdża przez skrzyżowanie, ale nie skręca na nim, to w procedurze $reduceVelocity(car, t, crossroadPenalty)$ jego prędkość jest redukowana z uwzględnieniem parametru $crossroadPenalty$.
- W procedurze $shouldChangeLane(car, t)$ następuje sprawdzenie, czy pojazd jest blisko (w zasięgu przejazdu w 1 kroku) pojazdu, który jest bezpośrednio przed nim i czy kierowca może bezpiecznie zmienić pas ruchu. Jeżeli tak, to w procedurze $changeLane(car, t)$ odbywa się zmiana pasa ruchu.

- Aby zachować bezpieczną odległość pojazdu od pojazdu znajdującego się bezpośrednio przed nim prędkość jest dodatkowo redukowana w procedurze $safeReduceVelocity(car, t)$.
- Z prawdopodobieństwem *prob* prędkość pojazdu ulega dodatkowej redukcji o stała wartość w procedurze $reduceVelocity(car, t)$.
- Pojazd wykonuje ruch w procedurze $makeMove(car, t)$ zgodnie z wyznaczoną prędkością.

Rozdział 2

Symulator TSF

2.1. Architektura symulatora

Symulator Traffic Simulation Framework został napisany w języku C# korzystając z technologii .NET Framework. Jego architektura składa się z następujących elementów:

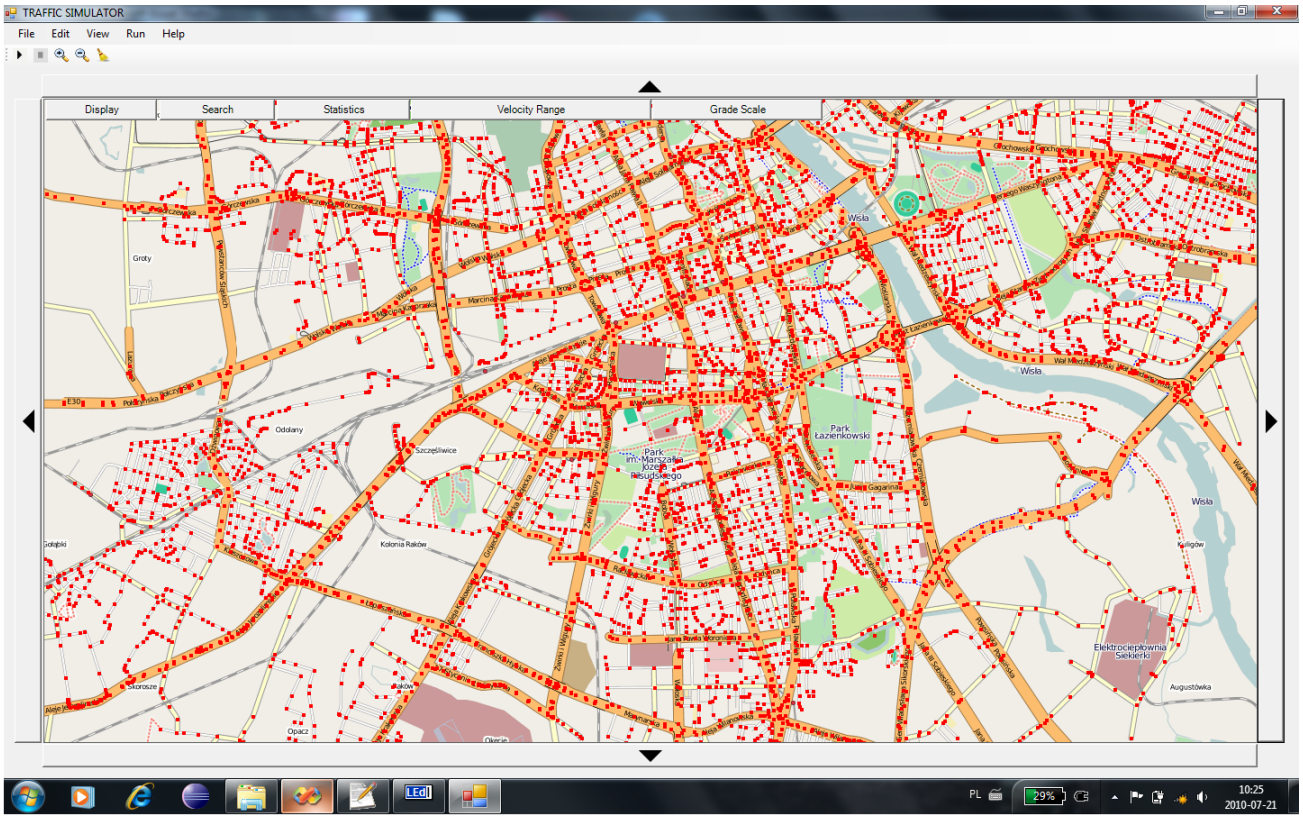
- Graficzny Interfejs Użytkownika (GUI)
- Moduł logiczny
- Moduł symulacji
- Moduł wejścia/wyjścia

W następnych podrozdziałach zostaną opisane wyżej wymienione elementy architektury systemu.

2.1.1. Graficzny Interfejs Użytkownika

Graficzny Interfejs Użytkownika (GUI) został wykonany przy użyciu technologii Windows Forms .NET [WF]. Służy on do wizualizacji przeprowadzanej symulacji oraz łatwej edycji jej parametrów. Wizualizacja wykorzystuje realistyczne mapy drogowe pochodzące z serwisu OpenStreetMap [OSM] (mapy są w nim tworzone i edytowowane przez użytkowników na podstawie zapisów sygnału GPS podczas rzeczywistego przejazdu). W oryginalnej wersji symulatora mapa obejmuje obszar Warszawy w zakresie długości geograficznej (20.830078125, 21.26953125) i szerokości (52.106505190756316, 52.375599176659101). Na mapie mogą być wyświetlane następujące atrybuty związane z ruchem drogowym:

- położenia pojazdów - reprezentowane jako punkty na mapie
- prędkości pojazdów - reprezentowane jako kolory punktów odpowiadających pojazdom
- położenie i stan sygnalizacji świetlnej na skrzyżowaniach
- wierzchołki grafu sieci drogowej
- średnie prędkości jazdy na poszczególnych odcinkach drogi
- odcinki, które są monitorowane podczas symulacji
- obszary, z których odcinki są monitorowane podczas symulacji
- rozkład punktów startowych
- rozkład punktów końcowych



Rysunek 2.1: Graficzny Interfejs Użytkownika

Funkcjonalność GUI GUI daje użytkownikowi następującą funkcjonalność:

- uruchamianie/zatrzymywanie symulacji z odpowiednimi parametrami
- odczyt i zapis informacji związanych z ruchem drogowym: tras przejazdu, sygnalizacji, rozkładu punktów startowych i końcowych, monitorowanych odcinków, monitorowanych obszarów
- edycję domyślnych średnich prędkości przejazdu (i ich odchyleń standardowych) dla poszczególnych typów dróg
- dodawania/usuwanie sygnalizacji świetlnej na skrzyżowaniach, edycję sygnalizacji
- edycję monitorowanych odcinków, monitorowanych obszarów
- edycję parametów symulacji
- edycję rozkładów punktów startowych i końcowych
- wyszukiwanie ulic wg nazwy
- ustawianie zakresu rozróżnianej (kolorystycznie) prędkości pojazdów
- generowanie tras przejazdu

2.1.2. Moduł logiczny

Moduł logiczny systemu TSF stanowią wszystkie struktury danych potrzebne do reprezentacji ruchu drogowego wraz z procedurami umożliwiającymi operacje na tych strukturach. Wśród najważniejszych komponentów tego modułu wymienić można:

- Graf sieci drogowej (zbiór wierzchołków połączonych krawędziami)
- Zbiór sygnalizacji świetlnej (są to obiekty umieszczone w niektórych wierzchołkach grafu sieci drogowej)
- Zbiór pojazdów uczestniczących w ruchu
- Rozkład punktów startowych i końcowych (każdy obszar mapy ma przypisaną pewną wartość **rank** określającą szanse na wylosowanie punktu startowego/końcowego z tego obszaru)
- Zbiór odcinków ulic, które są monitorowane w trakcie symulacji - informacje na temat tych odcinków (np. natężenie ruchu, średnia prędkość) są potem gromadzone i wypisywane w trakcie symulacji

Dla większości z ww komponentów istnieją w module logicznym programu TSF pomocnicze struktury ułatwiające i przyśpieszające operowanie na logice systemu.

2.1.3. Moduł symulacji

Moduł symulacji odpowiada za inicjowanie i przeprowadzanie symulacji ruchu drogowego. Inicjowanie symulacji polega na ustawieniu parametrów potrzebnych do przeprowadzenia symulacji. Wszystkie parametry są wczytywane z plików/ustawiane domyślnie w momencie uruchamiania symulacji, ale też mogą być uaktualniane później przez użytkownika z poziomu GUI.

Poniżej wykaz wszystkich parametrów, od których zależy przebieg symulacji i które można ustawić z poziomu GUI:

- Ilość pojazdów uczestniczących w ruchu od początku
- Czas t (w krokach), po którym cyklicznie rozpoczynają ruch następne pojazdy
- Ilość pojazdów, które rozpoczynają ruch co t kroków
- Czas trwania symulacji (ilość kroków)
- Czas trwania pojedynczego kroku symulacji (w milisekundach)
- Domyślne przyśpieszenie pojazdu w ciągu 1 kroku symulacji (w $\frac{km}{h}$)
- Parametr odpowiedzialny za zachowanie pojazdu podczas wjazdu na skrzyżowanie (poziom redukcji prędkości)
- Parametr odpowiedzialny za zachowanie pojazdu podczas skrętu na skrzyżowaniu (poziom redukcji prędkości)
- Domyślne średnie prędkości (i odchylenie od średnich) dla poszczególnych typów dróg
- Trasy przejazdu
- Rozkłady punktów startowych/końcowych

- Położenie i konfiguracja sygnalizacji świetlnej na skrzyżowaniach

Sama symulacja przebiega w skończonej ilości kroków i stanowi implementację modelu symulacji TSF opisanego w rozdziale 1.2. W każdym kolejnych kroku dla każdego pojazdu przeliczane jest jego nowe położenie i prędkość zgodnie z regułą ruchu opisaną w 1.2. Ponadto w kolejnych krokach symulacji modyfikowany jest stan sygnalizacji świetlnej na skrzyżowaniach, a prezentacja stanu całej symulacji jest również uaktualniana w module GUI. Dodatkowo w niektórych krokach symulacji niektóre dane związane z symulacją mogą być wypisywane do pliku (szczegóły w rozdziale 2.1.4).

Symulacja może być uruchamiana w różnym celu, w zależności od tego, co chcemy osiągnąć. W obecnej wersji programu TSF rozróżnianych jest 5 trybów, w których można uruchomić symulację:

- **Tryb standardowy** - w tym trybie symulator nie wypisuje żadnych dodatkowych informacji do zewnętrznych plików
- **Tryb natężenia ruchu** - w tym trybie do zewnętrznego pliku wypisywane są szeregi czasowe odpowiadające natężeniu ruchu na wybranych, monitorowanych odcinkach. Natężenie ruchu jest tu rozumiane jako:

$$\frac{\text{ilość pojazdów, które przejechały przez odcinek w danym przedziale czasowym}}{\text{długość przedziału czasowego}} \tag{2.1}$$

- **Tryb wykrywania korków** - w tym trybie do zewnętrznego pliku wypisywane są identyfikatory odcinków, na których tworzy się korek. Wypisywanie odbywa się w kolejności powstawania korka na danym odcinku. "Korek na odcinku"jest zdefiniowany jako wystąpienie w ciągu pewnego czasu (domyślnie - 6 minut) symulacji odpowiednio małej średniej prędkości (domyślnie - mniejszej niż $3\frac{km}{h}$) przy odpowiednio dużej ilości samochodów, które wjechały na ten odcinek (domyślnie - 10).
- **Tryb średniej prędkości** - w tym trybie do zewnętrznego pliku wypisywane są położenia i prędkości jazdy pojazdów uczestniczących w ruchu, a także średnie prędkości jazdy przez wybrane, monitorowane odcinki. Średnia prędkość jest w tym przypadku średnią harmoniczną prędkości przejazdu przez dany odcinek wszystkich samochodów, które na niego wjechały w określonym przedziale czasowym. Dla pojedynczego pojazdu prędkość ta wyraża się wzorem:

$$\frac{\text{długość odcinka}}{\text{czas przejazdu przez ten odcinek}} \tag{2.2}$$

- **Tryb optymalizacji sygnalizacji świetlnej** - w tym trybie do zewnętrznego pliku wypisywane są wartości funkcji celu dla kolejnych genotypów reprezentujących konfigurację sygnalizacji świetlnej.

2.1.4. Moduł Wejścia/Wyjścia

Wejście

Wiele danych i parametrów potrzebnych do poprawnego działania programu TSF musi zostać wczytanych z zewnętrznych źródeł przed rozpoczęciem symulacji. Za ich wczytywanie, przetwarzanie i przekazywanie do Modułu Logicznego odpowiada Moduł Wejścia/Wyjścia. W trakcie uruchamiania programu wczytuje on z plików następujące informacje:

- Struktura grafu sieci miejskiej
- Położenie i konfiguracja sygnalizacji świetlnej

- Rozkład punktów startowych/końcowych
- Trasy przejazdu dla pewnej liczby pojazdów (obliczone wcześniej w oparciu o zadany rozkład punktów startowych/końcowych)
- Domyślny zbiór odcinków monitorowanych podczas symulacji

Dodatkowo w trakcie działania symulacji pobierane są z dysku obrazy umożliwiające prezentowanie mapy przez GUI.

Wyjście

W trakcie symulacji ruchu drogowego program TSF może wypisywać informacje związane z symulacją, które mogą się przydać do dalszej analizy tego złożonego procesu. Wypisywane informacje zależą od trybu, w którym uruchamiana jest symulacja.

Tryb natężenia ruchu W tym przypadku format wypisywanych danych ma postać: [Czas, Wierzchołek1, Wierzchołek2, Ilość pojazdów].

Znaczenie poszczególnych atrybutów:

- Czas - czas w minutach od rozpoczęcia symulacji
- Wierzchołek1 - identyfikator wierzchołka, który jest wierzchołkem początkowym badanej krawędzi
- Wierzchołek2 - identyfikator wierzchołka, który jest wierzchołkem końcowym badanej krawędzi
- Ilość pojazdów - ilość pojazdów, która przejechała przez badany odcinek w zadanym przedziale czasowym

Tryb wykrywania korków W tym przypadku format wypisywanych danych ma postać: [Czas, Wierzchołek1_Wierzchołek2].

Znaczenie poszczególnych atrybutów:

- Czas - czas w sekundach od rozpoczęcia symulacji
- Wierzchołek1_Wierzchołek2 - Identyfikatory punktu początkowego i końcowego krawędzi, na której utworzył się korek

Tryb średniej prędkości W typ przypadku wypisywane są dane 2 rodzajów:

- dane pochodzące z pojazdów uczestniczących w ruchu i dotyczące ich położenia
- dane zagregowane dotyczące średnich prędkości na wybranych do monitorowania odcinkach

W pierwszym przypadku format wypisywanych danych ma postać: [Czas, Pojazd, Wierzchołek1, Wierzchołek2, Dystans, Prędkość, Szerokość geograficzna, Długość geograficzna]. Domyślnie takie dane są wypisywane co 10 sekund.

Znaczenie poszczególnych atrybutów:

- Czas - nr kroku symulacji, z którego pochodzi pomiar
- Pojazd - identyfikator pojazdu

- Wierzchołek1 - identyfikator wierzchołka, który jest początkiem odcinka, na którym znajduje się pojazd
- Wierzchołek2 - identyfikator wierzchołka, który jest końcem odcinka, na którym znajduje się pojazd
- Dystans - odległość pojazdu od początku odcinka, na którym się znajduje
- Prędkość - prędkość pojazdu
- Szerokość geograficzna - szerokość geograficzna pojazdu
- Długość geograficzna - długość geograficzna pojazdu

W drugim przypadku format wypisywanych danych ma postać [Czas, Wierzchołek1, Wierzchołek2, Prędkość, Ilość pojazdów]. Znaczenie poszczególnych atrybutów:

- Czas - czas od rozpoczęcia symulacji (w sekundach), w którym wypisywane są dane
- Wierzchołek1 - identyfikator wierzchołka, który jest początkiem monitorowanego odcinka
- Wierzchołek2 - identyfikator wierzchołka, który jest końcem monitorowanego odcinka
- Prędkość - średnia prędkość na danym odcinku w pewnym przedziale czasu
- Ilość pojazdów - ilość pojazdów, które przejechały przez dany odcinek w pewnym przedziale czasu

Rozdział 3

Adaptacyjne sterowanie ruchem ulicznym

3.1. Dotychczasowe badania

Zagadnienie adaptacyjnego sterowania ruchem ulicznym poprzez odpowiednią konfigurację sygnalizacji świetlnej jest bardzo trude i stanowi wielkie wyzwanie dla naukowców. Tym bardziej, że ilość pojazdów uczestniczących w ruchu drogowym stale wzrasta. Badane rozwiązania opierają się m.in. na teorii automatów komórkowych, logiki rozmytej [ChCh], systemów wieloagentowych [TSS] oraz algorytmach ewolucyjnych (w szczególności na algorytmie genetycznym, mrówkowym i rojowym) [CAG], [CYP], [OB], [STHB]. Na ich podstawie powstało wiele profesjonalnych systemów sterowania ruchem ulicznym [OTC], [SCATS], [SIE], [TELVENT], [TRB] zainstalowanych w dużych miastach na całym świecie. Systemy takie są cały czas rozwijane i usprawniane, a zapotrzebowanie na nowe, lepsze rozwiązania jest bardzo duże - gdyż w wielu miastach poziom zakorkowania ulic jest wciąż bardzo wysoki.

W niniejszym rozdziale przedstawię autorskie rozwiązanie wykorzystujące algorytm genetyczny optymalizacji konfiguracji świateł drogowych zaadoptowany do stworzonego i zaimplementowanego przeze mnie modelu TSF 1.2.

3.2. Podstawy teoretyczne

W pracy [GOR] przedstawiłem koncepcję tzw. *nadzorcy procesu*, czyli systemu służącego do adaptacyjnego zarządzania złożonymi procesami. Zaproponowałem w jaki sposób rozszerzyć system TSF do postaci kompleksowego systemu zarządzaniu ruchem ulicznym. Zgodnie z moim pomysłem, system taki miałby składać się z następujących komponentów:

- **Moduł monitorujący** - zbierający i wstępnie przetwarzający niskopoziomowe dane o procesie (np. o ruchu drogowym).
- **Moduł logiczny** - tworzący na podstawie gromadzonych danych *percepcję rzeczywistości* i identyfikujący złożone wzorce czasowo-przestrzenne występujące w gromadzonych danych.
- **Moduł sterujący** - odpowiedzialny za planowanie i podejmowanie odpowiednich decyzji na podstawie wiedzy posiadanej przez moduł logiczny.
- **Moduł wykonawczy** - wprowadzający w życie akcje proponowane przez moduł sterujący.

W przypadku moich badań każdy z tych 4 modułów jest zaimplementowany w programie TSF. Program przeprowadza symulacje rzeczywistego ruchu, więc ma pełną wiedzę na jego temat. W przypadku wykrycia zagrożeń może zaproponować i wykonać akcje mające na celu optymalizację ruchu.

W ogólności zarówno moduł logiczny jak i moduł sterujący mogłyby być wsparte przez *wiedzę dziedzinową*, czyli dodatkowe informacje na temat procesu (pochodzące najczęściej od eksperta). W przypadku ruchu drogowego wiedza dziedzinowa może się przydać np. do właściwego zdefiniowania pojęcia *korka drogowego*, lub ogólniej, wykrywania sytuacji, w których moduł sterujący powinien przeprowadzać akcje. W niniejszej pracy nie będę zajmował się jednak tą ciekawą (i jednocześnie bardzo trudną) tematyką.

Przedstawię natomiast w jaki sposób zaprojektowałem oraz zaimplementowałem moduł sterujący systemu zarządzania ruchem drogowym, którego działanie oparte jest na adaptacyjnym sterowaniu konfiguracją sygnalizacji świetlnej na skrzyżowaniach. Sterowanie to wykorzystuje algorytm genetyczny, którego ideę krotko przedstawię w następnym podrozdziale.

3.2.1. Algorytm genetyczne

Inspiracją algorytmu genetycznego ([MIC], [GOL]) jest rzeczywiste zjawisko ewolucji biologicznej. Idea polega na tym, że poszukiwane rozwiązania (np. optymalna synchronizacja świateł drogowych w mieście) są kodowane w postaci struktury genomu (np. jako wektor genów). Po utworzeniu puli przykładowych genomów, podlegają one reprodukcji zgodnie z zasadami panującymi w biologii, a więc zgodnie z procesem doboru naturalnego: najlepsze osobniki krzyżują się tworząc coraz lepsze (bardziej przystosowane) osobniki, które dalej podlegają mutacji.

Algorithm 3 Algorytm genetyczny

Require: Zbiór możliwych rozwiązań S
Ensure: Odpowiednie rozwiązanie ze zbioru S
  $A := ChooseInitialPopulation(S)$
  $A := Encode(A)$
  repeat
    $B := Select(A)$
    $C := Reproduce(B)$
    $A := Mutate(B)$
  until Istnieje odpowiednio dobry osobnik w zbiorze A
  $G := ChooseTheBestIndividual(A)$
  return $Decode(G)$

Algorytm najpierw wybiera zbiór przykładów ze zbioru wszystkich możliwych rozwiązań (procedura *ChooseInitialPopulation*). Zbiór taki może być kodowany jako zbiór genotypów reprezentowanych np. w postaci *wektora genów*, w którym każdy gen jest liczbą rzeczywistą. Funkcja kodująca powinna być bijekcją, aby dało się jednoznacznie odkodować optymalne rozwiązanie. Kodowanie odbywa się w procedurze *Encode*.

Dalsza część obliczeń wykonywana jest w pętli. Wybierany jest podzbiór całej populacji (genotypów), według ustalonego kryterium (procedura *Select*). Najczęściej istnieje funkcja, która dla danego genotypu ocenia jak dobrym jest on przedstawicielem populacji. Najlepsi przedstawiciele krzyżują się w parach (procedura *Reproduce*), tworząc odpowiednio duży zbiór genotypów potomnych (z każdej pary powstaje nowy genotyp stanowiący ich kombinację). Zbiór nowych genotypów podlega mutacji (*Mutate*), która nieznacznie (losowo) zmienia ich cechy. Jeżeli w tak otrzymanym zbiorze istnieje odpowiedni przedstawiciel populacji (według tej samej funkcji oceny jak selekcja), to przykład mu

odpowiadający jest zwracany przez algorytm (procedura *Decode*). Jeśli nie, ewolucja odbywa się dalej.

3.2.2. Wykorzystanie algorytmu genetycznego do optymalizacji konfiguracji świateł drogowych

W tym rozdziale przedstawię metodę optymalizacji konfiguracji sygnalizacji świetlnej w modelu TSF z wykorzystaniem algorytmu genetycznego.

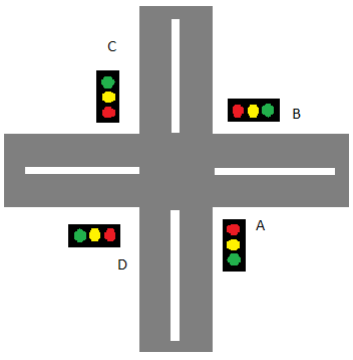
Reprezentacja genotypu

Przypomnijmy, że sygnalizacja świetlna jest obiektem umieszczonym na skrzyżowaniach, czyli w wierzchołkach grafu sieci drogowej. Każdy taki obiekt posiada m.in. atrybuty: t_{green}, t_{red}, t_{change}, *state*. Dla uproszczenia przyjąłem, że wartości t_{green} i t_{red}, czyli czas trwania fazy światła zielonego i czerwonego, są takie same dla każdej sygnalizacji występującej w modelu i wynoszą odpowiednio: 60 sekund i 60 sekund (czyli długość całej pojedynczej fazy, to 120 sekund). Parametrami, które mogą być modyfikowane, aby zoptymalizować globalny ruch pojazdów jest początkowa wartość t_{change} określająca przesunięcie w fazie oraz początkowa wartość *state*. Te 2 parametry można zakodować przy pomocy 1 parametru $timeToGreen \in \{0, 1, \ldots, 119\}$, określającego czas (w sekundach) do najbliższego przejścia pomiędzy fazą światła czerwonego i światła zielonego. Parametr ten w dalszej części pracy będzie oznaczany jako TG.

Przyjąłem dodatkowo założenie, że sygnalizacje w obrębie tego samego skrzyżowania muszą być ze sobą zsynchronizowane, aby zapewnić bezpieczny przejazd pojazdów przez to skrzyżowanie. Oznacza to, że w przypadku skrzyżowania zaprezentowanego na rysunku 3.1 sygnalizacje A i C będą miały taką samą wartość parametru TG. Podobnie taką samą wartość muszą mieć sygnalizacje B i D. Dodatkowo, aby zapewnić bezpieczny przejazd pojazdów przez to skrzyżowanie wartość TG dla sygnalizacji A (i C) musi być przesunięta o 60 względem TG dla B (i D). Mamy zatem warunki:

1. $TG_A = TG_C$
2. $TG_B = TG_D$
3. $TG_A = (TG_B + 60) \bmod 120$

Oznacza to, że wartość TG dla sygnalizacji A,B,C,D jest wyznaczona jednoznacznie przez wartość TG dla dowolnej spośród tych sygnalizacji.



Rysunek 3.1: Automat komórkowy w modelu Nagela-Schreckenberga

Definicja 3.2.1. *Niech $A = \{A_1, A_2, \ldots, A_k\}$ będzie niepustym zbiorem sygnalizacji świetlnych na pewnym skrzyżowaniu w modelu TSF.* ***Reprezentantem*** *zbioru A będziemy nazywać dowolny element tego zbioru i będziemy go oznaczać jako $r(A)$.* ***Reprezentantem*** *dowolnego elementu $A_i \in A$ będziemy nazywać reprezentanta $r(A)$ zbioru A i będziemy go oznaczać jako $r(A_i)$ (zatem $\forall_{A_i \in A} r(A_i) = r(A)$).*

Definicja 3.2.2. *Niech S będzie zbiorem wszystkich sygnalizacji w grafie sieci drogowej.* ***Reprezentacją*** *S będziemy nazywać dowolny jego podzbiór $S' \subseteq S$ spełniający warunki:*

1. $\forall_{A_i \in S} \quad r(A_i) \in S'$
2. *jeśli $A_i \neq r(A_i)$, to $A_i \notin S'$*

W proponowanym algorytmie genetycznym pojedynczy gen będzie właśnie wartością TG dla reprezentanta zbioru sygnalizacji świetlnych na pojedynczym skrzyżowaniu. Genotyp będzie zatem wektorem genów odpowiadających dowolnej, ustalonej reprezentacji zbioru S sygnalizacji w grafie sieci drogowej.

Wybór i kodowanie początkowej populacji

W proponowanym algorytmie genetycznym wybór początkowej populacji odbywa się w procedurze *ChooseInitialPopulation*. Na początku tworzona jest reprezentacja zbioru wszystkich sygnalizacji świetlnych w grafie. Każdej sygnalizacji z tego zbioru odpowiadać będzie 1 gen w wektorze genów, będący reprezentantem tej sygnalizacji. Potem tworzona jest pula ustalonej liczby genotypów, a każdy gen w genotypie przyjmuje losową wartość TG ze zbioru $\{0, 1, \ldots, 119\}$. Każdy taki genom wyznacza jednoznacznie konfigurację sygnalizacji świetlnej i na odwrót: każda konfiguracja sygnalizacji świetlnej jednoznacznie wyznacza genom (pod warunkiem, że sygnalizacje na skrzyżowaniach są zsynchronizowane tak, aby zapewnić bezpieczeństwo przejazdu). Dzięki temu możliwe jest poprawne kodowanie sygnalizacji i dekodowanie genotypu.

Funkcja oceny genotypu

Jednym z kluczowych elementów podczas projektowania algorytmu genetycznego jest wybór odpowiedniej funkcji oceny genotypu. Funkcja ta służy do nadawania wartości genotypom tak, aby dało się wybrać do dalszej ewolucji najlepsze z nich. Występuje również w warunku stopu pętli, w której odbywa się ewolucja.

W naszym przypadku do oceny jakości genotypu potrzebne jest przeprowadzenie symulacji komputerowej przy użyciu programu TSF.

Używane przeze mnie parametry symulacji P zostaną szczegółowo opisane w rozdziale 3.2.3.

Ponadto testowałem 2 funkcje F oceniające jakość symulacji:

- łączny czas jazdy kierowców z prędkością poniżej $20\frac{km}{h}$ - funkcja Time<20
- łączny czas, w którym kierowcy mieli prędkość $0\frac{km}{h}$ - funkcja Time0

Selekcja najlepszych genotypów

W procesie selekcji wybierane są najlepsze genotypy z całej populacji, aby mogły potem wziąć udział w dalszej ewolucji. Selekcja dokonuje się poprzez zaaplikowanie funkcji oceny 3.2.2 do wszystkich genotypów w populacji S i wybranie z całej puli osobników $\sqrt{|S|}$ tych, dla których wartość funkcji oceny jest najlepsza.

Algorithm 4 Funkcja oceny genotypu

Require: Genotyp G reprezentujący konfigurację sygnalizacji świetlnej; Czas trwania symulacji T; Funkcja F oceniająca jakość symulacji; Parametry symulacji P.
Ensure: Wartość funkcji oceny dla genotypu G
  $value = 0$;
  for $i = 1$ to 5 **do**
    Skonfiguruj sygnalizację świetlną zgodnie z genotypem G;
    Przeprowadź symulację przy użyciu programu TSF przy zadanych parametrach P trwającą T kroków;
    $value = value$+F(przeprowadzona symulacja);
    i++;
  end for
  $value = value/5$;
  return $value$;

Krzyżowanie najlepszych genotypów

Po selekcji najlepszych z całej populacji genotypów następuje ich krzyżowanie, celem utworzenia nowej populacji. Procedurę krzyżowania przedstawia algorytm 5.

Algorithm 5 Procedura krzyżowania

Require: Zbiór S najlepszych przedstawicieli populacji
Ensure: Nowa populacja P
  $P = \emptyset$;
  N =rozmiar genotypów w populacji P
  for all G1 in S **do**
    for all G2 in S **do**
      G=nowy genotyp
      for i=1 to N **do**
        L=losowo wybrana liczba ze zbioru $\{0, 1\}$.
        if L==0 **then**
          Umieść w genotypie G na pozycji i gen G1[i]
        else
          Umieść w genotypie G na pozycji i gen G2[i]
        end if
        $P = P \cup \{G\}$;
      end for
    end for
  end for
  return P;

Mutacja

Po utworzeniu nowej populacji osobników należy dokonać ich mutacji. Odbywa się to w procedurze *Mutate*, którą ilustruje algorytm 6.

Algorithm 6 Procedura mutacji

Require: Populacja P, próg mutacji q
Ensure: Zmutowana populacja P'
  $P' = \emptyset$;
  N =rozmiar genotypów w populacji P
  for all G in P **do**
    for i = 1 to N **do**
      L =losowa liczba z przedziału [0, 1]
      if L<q **then**
        $G[i]$ losowa wartość ze zbioru $\{0, 1, \ldots, 119\}$
      end if
      $P' = P' \cup \{G\}$;
    end for
  end for
  return P';

Zatrzymanie ewolucji

Ewolucja populacji musi się w pewnym momencie zakończyć. W swojej pracy przyjąłem, że kryterium stopu będzie przeprowadzenie określonej liczby N kroków ewolucji, czyli zatrzymanie ewolucji w chwili otrzymania N-tej populacji potomnej. W przypadku moich obliczeń wartość N wynosiła 9.

Innymi często stosowanymi kryteriami stopu są:

- Kryterium optymalnej wartości - zatrzymujemy ewolucję w momencie, gdy otrzymamy w populacji osobnika o dostatecznie dobrej wartości funkcji oceny
- Kryterium zbieżności - zatrzymujemy ewolucję w momencie, gdy w kolejnych jej krokach różnica w wartościach funkcji oceny dla najlepszych osobników w populacji jest bardzo mała

Testowanie tych kryteriów w proponowanym algorytmie genetycznym będzie przedmiotem moich badań w przyszłości.

3.2.3. Implementacja algorytmu genetycznego

W tym rozdziale przedstawię w jaki sposób zaimplementowałem algorytm genetyczny opisany powyżej, a więc:

- Jakie dokładnie zostały użyte parametry
- Jak przeprowadzane były eksperymenty

W przeprowadzanych przeze mnie eksperymentach każda symulacja trwała 10 minut (600 sekund), parametry P były standardowymi parametrami symulacji zgodnymi z tabelą 3.1. Populacja początkowa składała się ze 100 osobników, pojedynczy osobnik posiadał natomiast 292 geny w swoim genotypie (jest to ilość skrzyżowań z sygnalizacją świetlną na wczytywanej przez program TSF mapie Warszawy). Algorytm genetyczny był testowany dla 2 różnych funkcji oceny jakości genotypu (3.2.2), a w każdym przypadku warunkiem stopu algorytmu było przeprowadzenie $N = 9$ iteracji ewolucji. W procedurze selekcji wybierane było $\sqrt{100} = 10$ najlepszych osobników, które potem podlegały procedurze krzyżowania, a nowo powstałe osobniki ulegały mutacji zgodnie z procedurą 6 przy parametrze $q = 0.05$ (czyli w każdym genotypie średnio 5% wszystkich genów zmieniało swoją wartość).

Tablica 3.1: Parametry symulacji użyte do przeprowadzenia doświadczeń

Nazwa parametru	Opis parametru	Wartość parametru
NrOfCars	Ilość pojazdów	30000
TimeGap	Czas, po którym nowe pojazdy rozpoczynają ruch	1 sekunda
Step	Czas trwania kroku symulacji	1000 milisekund
NewCars	Ilość pojazdów, które rozpoczynają ruch co TimeGap sekund	20
Steps	Czas trwania symulacji	600 kroków
Accelerate	Przyśpieszenie pojazdów	$10\frac{km}{h}$
Crossroad	Parametr odpowiedzialny za zachowanie pojazdów (redukcję prędkości) na skrzyżowaniu	0.25
Turning	Parametr odpowiedzialny za zachowanie pojazdów (redukcję prędkości) na skrzyżowaniu podczas skrętu	0.5

Tablica 3.2: Tabela przedstawiająca wartości funkcji celu w kolejnych krokach ewolucji algorytmu w przypadku obu badanych funkcji

Krok ewolucji	Time0	Time<20
1	5937535	8381108
2	5897720	8350837
3	5884938	8328534
4	5855378	8292620
5	5840454	8292925
6	5811827	8242777
7	5794758	8228153
8	5780861	8234131
9	5752666	8228039

3.2.4. Wyniki doświadczeń

W drugiej kolumnie Tabeli 3.2 są wartości funkcji oceny dla najlepszych osobników w populacji w kolejnych krokach ewolucji w przypadku gdy funkcja celu jest łączną ilością sekund, przez którą pojazdy stoją w miejscu. Jak widać, w każdym kroku ewolucji wartość funkcji oceny dla najlepszego osobnika zmniejsza się. Ostatecznie w 9-tym kroku ewolucji udaje się uzyskać wynik o 3.11% lepszy niż w pierwszym kroku. Szczegółowe wyniki wartości funkcji celu dla wszystkich kroków ewolucji można znaleźć w załączniku. W trzeciej kolumnie Tabeli 3.2 są z kolei wartości funkcji oceny dla najlepszych osobników w populacji w kolejnych krokach ewolucji w przypadku, gdy funkcja celu jest łączną ilością sekund, w których pojazdy poruszają się z prędkością poniżej $20\frac{km}{h}$. Nie w każdym kroku ewolucji otrzymuje się lepszą wartość funkcji celu, mimo to w kroku 9-tym pojawił się osobnik o najlepszej wartości funkcji celu. Wartość ta była o 1.82% lepsza niż wartość funkcji celu dla najlepszego osobnika w pierwszym kroku ewolucji. Szczegółowe wyniki wartości funkcji celu dla wszystkich kroków ewolucji można znaleźć w załączniku 4.

3.3. Podsumowanie

Zastosowanie algorytmu genetycznego przyniosło polepszenie jakości ruchu drogowego zarówno w przypadku funkcji oceny Time0, jak i w przypadku funkcji oceny Time<20. Poprawa nie jest jednak znacząca (odpowiednio 3.11% i 1.82%). Z powodu dużej złożoności obliczeniowej udało się przeprowadzić jedynie kilka (dokładnie 9) iteracji algorytmu genetycznego. Można się spodziewać, że dłuższy czas ewolucji może przynieść znacznie większe polepszenie jakości ruchu drogowego - tym bardziej, że przeszukiwana przestrzeń rozwiązań (a więc przestrzeń liniowa ${\mathbb{Z}_{120}}^{292}$) jest bardzo duża w porównaniu do ilości genotypów, które zostały zbadane przez algorytm ($9 \cdot 100 = 900$). Można się spodziewać, że przy istotnie większej ilości kroków algorytmu udałoby się uzyskać poprawę rzędzu kilkudziesięciu procent w stosunku do początkowej populacji. Przeprowadzenie większej ilości kroków wymagałoby jednak większej mocy obliczeniowej lub poprawienia skalowalności proponowanego algorytmu. Między innymi tym drugim zagadnieniem bedę zajmował się w swoich przyszłych badaniach w ramach pracy doktorskiej.

Rozdział 4

Podsumowanie

W pracy przedstawiłem metodę adaptacyjnego sterowania ruchem ulicznym poprzez optymalizowanie konfiguracji sygnalizacji świetlnej na skrzyżowaniach przy pomocy algorytmu genetycznego. W algorytmie tym genotypy były wektorami genów, z których każdy reprezentował pojedynczą sygnalizację (będącą reprezentacją zbioru sygnalizacji na pojedynczym skrzyżowaniu). Potrzebne doświadczenia zostały przeprowadzone za pomocą symulacji komputerowej z wykorzystaniem stworzonego przeze mnie programu TSF (Traffic Simulation Framework). Opisana została również szczegółowo funkcjonalność programu oraz opracowany przeze mnie model teoretyczny symulacji.

Przeprowadzone doświadczenia zaowocowały uzyskaniem lepszych genotypów - odpowiadających optymalniejszej konfiguracji sygnalizacji świetlnej. Z uwagi na dużą złożoność obliczeniową (czasową i pamięciową) eksperymentów udało się przeprowadzić stosunkowo niedużą liczbę iteracji algorytmu, w związku z tym otrzymane ulepszenie badanych funkcji oceny jest jedynie kilkuprocentowe. Można się jednak spodziewać, że dysponując większą mocą obliczeniową oraz poprawiając skalowalność proponowanego rozwiązania da się uzyskać większą poprawę wartości funkcji oceny.

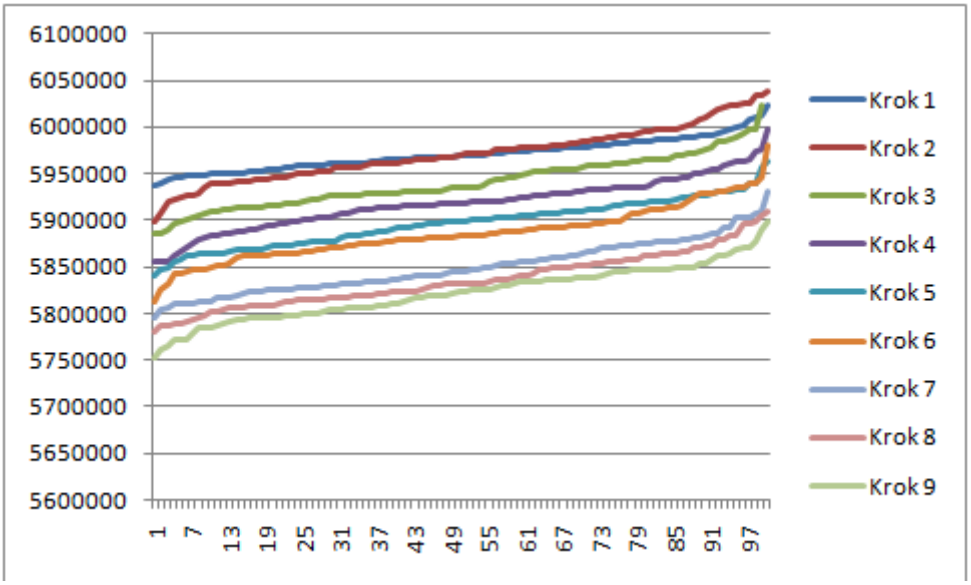
Właśnie poprawa skalowalności algorytmu genetycznego a także zastosowanie innych funkcji oceny (m.in. badanie średniej prędkości jazdy kierowców) oraz innych algorytmów ewolucyjnych (np. algorytmu mrówkowego, rojowego) będzie jednym z obszarów moich dalszych badań nad omawianym problemem w ramach studiów doktoranckich na Wydziale MIMUW.

Dodatek A

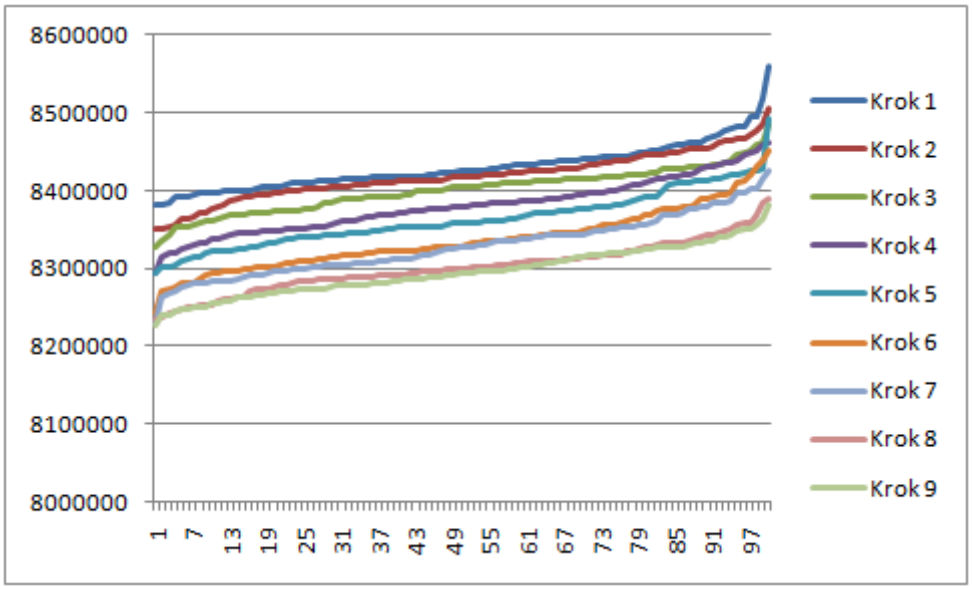
Wyniki doświadczeń przy pomocy programu Traffic Simulation Framework

W niniejszy dodatku przedstawione zostaną szczegółowe wyniki eksperymentów: przebieg algorytmu genetycznego zasymulwanego przy użyciu programu komputerowego TSF. W tabelach A.1-A.3 znajdują się wyniki dla algorytmu, w którym funkcja oceny jakości genotypu to Time0. Wyniki te zobrazowane są też na rysunku A.1.

W tabelach A.4-A.6 znajdują się z kolei wyniki dla algorytmu, w którym funkcja oceny to Time<20. Wyniki te zobrazowane są też na rysunku A.2.



Rysunek A.1: Wartości funkcji celu Time0 dla kolejnych kroków algorytmu genetycznego



Rysunek A.2: Wartości funkcji celu Time<20 dla kolejnych kroków algorytmu genetycznego

Tablica A.1: Tabela przedstawiająca wartości funkcji celu w kolejnych krokach ewolucji algorytmu w przypadku funkcji Time0 - część 1

Gen	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9
1	5947096	5942776	5989125	5871243	5869264	5875619	5841108	5842674	5800712
2	6002009	5957558	5928293	5893851	5899256	5880450	5839236	5818956	5813313
3	5981969	5926514	5930156	5917948	5872086	5846099	5863004	5823675	5803781
4	5951648	6019723	5889313	5950034	5914510	5897177	5860913	5817164	5852456
5	6011114	5968313	5885894	5906335	5868360	5863886	5876651	5823345	5785131
6	5950667	6002016	5925839	5945804	5932280	5910771	5815987	5789615	5848005
7	5989093	5981605	5962089	5913166	5882852	5883084	5803864	5821159	5795908
8	5973916	5990139	5918814	5928828	5887414	5882907	5853950	5787830	5849336
9	5978086	5970557	5969122	5934750	5901911	5928442	5834401	5805555	5844665
10	5976103	5991907	5934237	5903260	5908895	5936233	5831041	5857737	5825118
11	5967370	5996715	5954703	5884076	5858043	5842578	5824504	5830535	5794005
12	5961824	5979092	5957962	5894136	5956901	5864681	5826119	5812468	5814895
13	5964772	6038136	5941188	5912665	5904360	5847118	5882844	5870528	5796283
14	6008645	6015014	5920260	5916403	5899627	5882996	5855457	5829442	5771471
15	5974849	5939288	5928943	5885069	5895947	5843934	5851065	5798602	5808978
16	5976101	5988281	5899807	5890790	5884722	5862321	5873919	5847904	5785737
17	6022325	6004039	5969559	5918516	5868797	5870375	5833402	5857095	5844526
18	5995903	5981071	5985556	5856064	5930097	5848762	5834744	5839753	5772490
19	5991085	5942004	5931810	5929010	5926853	5871135	5892069	5857553	5818986
20	5945657	5942417	5942866	5900455	5932481	5883153	5881922	5806961	5876717

Tablica A.2: Tabela przedstawiająca wartości funkcji celu w kolejnych krokach ewolucji algorytmu w przypadku funkcji Time0 - część 2

Gen	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9
21	5976903	5981143	5996687	5963563	5869678	5891286	5903679	5820015	5817910
22	5984106	5987540	5952465	5915222	5902618	5980377	5863711	5808577	5808761
23	5975849	5957549	5914137	5915173	5900475	5914858	5877086	5831203	5805349
24	5960326	6021237	5972436	5917224	5865213	5847939	5875668	5821750	5824660
25	5960931	5925115	5960706	5959530	5864267	5893464	5842743	5832500	5829903
26	5969921	5984183	5913479	5897404	5848506	5885114	5877554	5816163	5845401
27	5968435	5939756	5992727	5924763	5903516	5894500	5902117	5871952	5899483
28	5954292	5994790	5935338	5932867	5908977	5881091	5813674	5856306	5839147
29	5969231	5977903	5985724	5895880	5877409	5861466	5872116	5824253	5824044
30	5979467	5971048	5899106	5919250	5873119	5879165	5877571	5904852	5789334
31	5945075	5922106	5946639	5915029	5899360	5863350	5794758	5816681	5778926
32	5988283	6034857	5929447	5921980	5891715	5891174	5811912	5833173	5806493
33	5968194	5946265	5942702	5937714	5863195	5875535	5828598	5831918	5811296
34	5964942	5953211	5928041	5913336	5898094	5887021	5931094	5851331	5846795
35	6000621	5999229	5922038	5955174	5891020	5872295	5830835	5824721	5771497
36	5976500	5945347	5902474	5920040	5930273	5945392	5862229	5807996	5890377
37	5961655	5951742	5929642	5920397	5902989	5864476	5819207	5786126	5803536
38	5972015	5950160	5964645	5888298	5917707	5892872	5826174	5820039	5797995
39	5967172	5971850	5963280	5973861	5904938	5893885	5907114	5898620	5849658
40	5984639	5979810	5911190	5927071	5861136	5923741	5839710	5800976	5816196
41	5950647	5939735	5917065	5934570	5877938	5875076	5870278	5852469	5796509
42	5965894	5966116	5959521	5929198	5864280	5883240	5870605	5849778	5752666
43	5959656	5998277	5918418	5898499	5872791	5864513	5880675	5815834	5847049
44	5950582	5977744	5926590	5944959	5840454	5878685	5831384	5873851	5807136
45	5978167	6011092	5978296	5941820	5923077	5910947	5891293	5862416	5805350
46	5947706	5968464	5931464	5916171	5916032	5842905	5844531	5791092	5835884
47	5950866	5993283	5965377	5905916	5924831	5824788	5848017	5850920	5792522
48	5968920	5960280	5914047	5942931	5868259	5939960	5865946	5826659	5846661
49	5957166	5977459	5929855	5914395	5894485	5934529	5857955	5831829	5829774
50	5966542	6033365	5926125	5966257	5917683	5931090	5826799	5909975	5835728
51	5948843	6009040	5913432	5934283	5905750	5891187	5866912	5824335	5849876
52	5997950	5977064	5976358	5856370	5918120	5832331	5823643	5838498	5867766
53	5968320	5920984	5909541	5950278	5919094	5938412	5880062	5866549	5835324
54	5958007	5948537	5934770	5855378	5874667	5930058	5859610	5895599	5796174
55	5952408	5987614	5915129	5903339	5887304	5894511	5826046	5884108	5849003
56	5988789	5946341	5905731	5902081	5921109	5887089	5902231	5815604	5843124
57	5977327	5957688	5909995	5919808	5916792	5867523	5845234	5814825	5803232
58	5992101	5906465	5927539	5879737	5911642	5928501	5827582	5810826	5837564
59	5964593	5957566	5922018	5931306	5886336	5913778	5810034	5833019	5833849
60	5977454	6025214	5984183	5900390	5964003	5892100	5823021	5866010	5862934

Tablica A.3: Tabela przedstawiająca wartości funkcji celu w kolejnych krokach ewolucji algorytmu w przypadku funkcji Time0 - część 3

Gen	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9
61	5992302	5972573	5915584	5935035	5929394	5852173	5805911	5808922	5796653
62	5974013	5952397	6023506	5961825	5872305	5811827	5909067	5878260	5796323
63	5969166	5944211	5896204	5912402	5892119	5898387	5837205	5809246	5869936
64	5987560	5942997	5931017	5997711	5912135	5909802	5872041	5813378	5836760
65	5958656	5962682	5960042	5925077	5896860	5898612	5874994	5804593	5831735
66	5957635	5927487	5974792	5954772	5930238	5889008	5849984	5882694	5861656
67	5951215	5989990	5957440	5976143	5847839	5919388	5815988	5795417	5807182
68	6011732	5962069	5959737	5912191	5901332	5866054	5869956	5780861	5760317
69	5986780	5979062	5906911	5914850	5881568	5878708	5823108	5835848	5847589
70	5954122	5960836	5911302	5890395	5883658	5886196	5810101	5815312	5801726
71	5984093	5957906	5965256	5885969	5909687	5877666	5885747	5853375	5853564
72	5981392	5983825	5953998	5862822	5900745	5854529	5840553	5835384	5787438
73	5991032	5951166	5930152	5963830	5909119	5880497	5853678	5870737	5790211
74	5960245	5897720	5954856	5930575	5882770	5902953	5841284	5864961	5822922
75	5969778	5968095	5884938	5933013	5920236	5860434	5852604	5851088	5764811
76	5973903	5972615	5935969	5935779	5854941	5861354	5847916	5863346	5841149
77	5979748	5942586	5937897	5921864	5907757	5881754	5809873	5861335	5833668
78	5960982	5981742	5996601	5883287	5923592	5887294	5829700	5819527	5825665
78	5948497	5997968	5945967	5943543	5940221	5851425	5885816	5835294	5811411
80	5967254	5939498	5926494	5919628	5907949	5880098	5855073	5840854	5798488
81	5940228	5997865	5970825	5936040	5868005	5888228	5832085	5846004	5819264
82	5959719	5972491	5954594	5874758	5894532	5862930	5833019	5862133	5838247
83	5978762	5996397	5936123	5881239	5865749	5879045	5879860	5792834	5858206
84	5962691	5965856	5952901	5943741	5896165	5876656	5855539	5880281	5833341
85	5937535	5934370	5929955	5926856	5892612	5914728	5830458	5808239	5846870
86	5961432	5976598	5954785	5932144	5906156	5875595	5841111	5801760	5846693
87	5954909	6022783	5950622	5934003	5877315	5898658	5857827	5855997	5784494
88	5986045	5964864	5959213	5865457	5864663	5865407	5848254	5832663	5835385
89	5984698	5960138	5966122	5917942	5907699	5929729	5816897	5813979	5828218
90	5943051	6025999	5932038	5918923	5877996	5896896	5820258	5831953	5819087
91	5985864	5988939	5960005	5898265	5875763	5862483	5825905	5807253	5870180
92	5972168	5957417	5952545	5891198	5917127	5870088	5809693	5849133	5826030
93	5980609	5927782	5928023	5908279	5900595	5890612	5872553	5816349	5833502
94	5964478	5976256	5913093	5951627	5861153	5907691	5812371	5895471	5798746
95	5970691	5975107	5949051	5929322	5925613	5872955	5839102	5848147	5838949
96	5959484	5949702	5923631	5920851	5940447	5869368	5833458	5848575	5818403
97	5982370	5960793	5964850	5926934	5876201	5906200	5827269	5789858	5794440
98	5963367	6023289	5925738	5910038	5912090	5881150	5844471	5865110	5800500
99	5950321	5961816	5916813	5888443	5919700	5933481	5835433	5856496	5864957
100	5951441	5965152	5996715	5910450	5926504	5910769	5860059	5823529	5821355

Tablica A.4: Tabela przedstawiająca wartości funkcji celu w kolejnych krokach ewolucji algorytmu w przypadku funkcji Time<20 - część 1

Gen	Krok 1	Krok 2	Krok 3	Krok 4	Krok 5	Krok 6	Krok 7	Krok 8	Krok 9
1	8469703	8411361	8402845	8369034	8304388	8331868	8342273	8251366	8257624
2	8457308	8410035	8374739	8367482	8353501	8316423	8284185	8250068	8271751
3	8437681	8435834	8392699	8447456	8376666	8350855	8313209	8283253	8247294
4	8400406	8406709	8398554	8349402	8422503	8310396	8337451	8296154	8301828
5	8423786	8372514	8407444	8462666	8320135	8242777	8339681	8290310	8286102
6	8560436	8453537	8369982	8392717	8342380	8345343	8343367	8256700	8248751
7	8445889	8351330	8414983	8396624	8346398	8326516	8341780	8259507	8289011
8	8414209	8405756	8359171	8390566	8371800	8281895	8343448	8296331	8290793
9	8466177	8416435	8412317	8360515	8418028	8319206	8266912	8251051	8293030
10	8402901	8434696	8411415	8346170	8352645	8376229	8282593	8277577	8317568
11	8451964	8444736	8372337	8413440	8292925	8324770	8305012	8234131	8277983
12	8397399	8464796	8409829	8319543	8301930	8314705	8318442	8263886	8272255
13	8405258	8394950	8392610	8344341	8331399	8373761	8291115	8293568	8308642
14	8398845	8410812	8418475	8353957	8391463	8323153	8328801	8389845	8320083
15	8419553	8412724	8393749	8348601	8350837	8322630	8308000	8261037	8310558
16	8430367	8393694	8383591	8408691	8361052	8393608	8290576	8310510	8228039
17	8494551	8364020	8399848	8382163	8339149	8310286	8296962	8292464	8380978
18	8425994	8416487	8353371	8328768	8384241	8335466	8308344	8295952	8251050
19	8517534	8453670	8404524	8348255	8352689	8326774	8281016	8316892	8317400
20	8462152	8403121	8392925	8327193	8365564	8332596	8329021	8298499	8291111

Tablica A.5: Tabela przedstawiająca wartości funkcji celu w kolejnych krokach ewolucji algorytmu w przypadku funkcji Time<20 - część 2

Gen	Krok 1	Krok 2	Krok 3	Krok 4	Krok 5	Krok 6	Krok 7	Krok 8	Krok 9
21	8409404	8467839	8341614	8351785	8373885	8359721	8334313	8289846	8249358
22	8440087	8422742	8388640	8353973	8357279	8310302	8353520	8287544	8255461
23	8408918	8426144	8419498	8422720	8379213	8337868	8367500	8278912	8335201
24	8442980	8464745	8368405	8419032	8342639	8340745	8345753	8298078	8340285
25	8416725	8424926	8404396	8375987	8351415	8305544	8377227	8345249	8265376
26	8418115	8389797	8408827	8438140	8354160	8302124	8360766	8302018	8283906
27	8419437	8433607	8375118	8435285	8327731	8368932	8352966	8357948	8306542
28	8428185	8354465	8409550	8370833	8414533	8389576	8327583	8315110	8306979
29	8432480	8438273	8406615	8404278	8362538	8321703	8356831	8310394	8293653
30	8459791	8411789	8387038	8351791	8333741	8343537	8281147	8308673	8327559
31	8449231	8437969	8431675	8385074	8380135	8344213	8337904	8291362	8297765
32	8398450	8408671	8404744	8343959	8353599	8345745	8290566	8286780	8320862
33	8417633	8391376	8420945	8390165	8325384	8299622	8280290	8310253	8312991
34	8415178	8418305	8483864	8406969	8321179	8311445	8298889	8310752	8351584
35	8414073	8367003	8391108	8377489	8374812	8321907	8351412	8273599	8262735
36	8405604	8466608	8427486	8384834	8310619	8277109	8368026	8301386	8277446
37	8433946	8428081	8370016	8350488	8353336	8286948	8305789	8317572	8355440
38	8409122	8453224	8445030	8384935	8338627	8438182	8342326	8342803	8272838
39	8413161	8370193	8367364	8362356	8354271	8394972	8384749	8303069	8279998
40	8461428	8422250	8374952	8374801	8382667	8337729	8313792	8286756	8282160
41	8407588	8441915	8391788	8382475	8345309	8290489	8299650	8289583	8330021
42	8443058	8413168	8403670	8374716	8492364	8341667	8352230	8307643	8296821
43	8457895	8350837	8328534	8292620	8342828	8379618	8348918	8262368	8326936
44	8437847	8417970	8373644	8397099	8321255	8282029	8286558	8368704	8286686
45	8442407	8446174	8413892	8368567	8379511	8296535	8301695	8274772	8240081
46	8417156	8398399	8359825	8414300	8341743	8392752	8304314	8303743	8336118
47	8397475	8471014	8360331	8384817	8346915	8355686	8375777	8287254	8296356
48	8399196	8401002	8415171	8395555	8330172	8389351	8387962	8321116	8295489
49	8424338	8447064	8432252	8387841	8314762	8326712	8316576	8314166	8268866
50	8481216	8425972	8375194	8338333	8410556	8339409	8324715	8289469	8351268
51	8437412	8437169	8389441	8332954	8428653	8324789	8311048	8270016	8257719
52	8410630	8413714	8410089	8320837	8360725	8271648	8271007	8332221	8278900
53	8418245	8402378	8427754	8407120	8344657	8376831	8396690	8287164	8348227
54	8417581	8427179	8422419	8458199	8347693	8295414	8383489	8302850	8239804
55	8434803	8424855	8368752	8434028	8372556	8344997	8335897	8328519	8289463
56	8391305	8391546	8405884	8380147	8360064	8321202	8424348	8329848	8328318
57	8426361	8413859	8410029	8418186	8381223	8358147	8357576	8355090	8320564
58	8436658	8400604	8437638	8417710	8376769	8350915	8379557	8279807	8326431
59	8480829	8448209	8417273	8414447	8389531	8380651	8307399	8273963	8332140
60	8482474	8401288	8414743	8361667	8335435	8335137	8384483	8252124	8332519

Tablica A.6: Tabela przedstawiająca wartości funkcji celu w kolejnych krokach ewolucji algorytmu w przypadku funkcji Time<20 - część 3

Gen	Krok 1	Krok 2	Krok 3	Krok 4	Krok 5	Krok 6	Krok 7	Krok 8	Krok 9
61	8495461	8454785	8333976	8385805	8323405	8327328	8327848	8242728	8321532
62	8390742	8362541	8378252	8314483	8322189	8420369	8370705	8337737	8278688
63	8441589	8456893	8422351	8431847	8357917	8296340	8312868	8346992	8295982
64	8437497	8404066	8428107	8420020	8424245	8317951	8296898	8300054	8328354
65	8403825	8428441	8414337	8364829	8368143	8303468	8228153	8271992	8289070
66	8424370	8396647	8430401	8399715	8355104	8363990	8317989	8316694	8340884
67	8415406	8413382	8383208	8398907	8412754	8355082	8349200	8310117	8317352
68	8436688	8419503	8434274	8394454	8313962	8301315	8336064	8332735	8255114
69	8432112	8418577	8389492	8377362	8391539	8380229	8274820	8333065	8342035
70	8381540	8400624	8372579	8385218	8352302	8315804	8402337	8284339	8299501
71	8444448	8395021	8372289	8371470	8410157	8328719	8292793	8324233	8313275
72	8442031	8412671	8458704	8347848	8371655	8301407	8284142	8253717	8262788
73	8425431	8376261	8431799	8345409	8312262	8450541	8322158	8340682	8245462
74	8450976	8420919	8356122	8397742	8425327	8280816	8335317	8321297	8270910
75	8431712	8460825	8399794	8324294	8413361	8354745	8342501	8246949	8315243
76	8412586	8476953	8373969	8373866	8372509	8307023	8377894	8294838	8301377
77	8442400	8408139	8435092	8332531	8418999	8272345	8283322	8298650	8304211
78	8418747	8396978	8394471	8379190	8385834	8317948	8296601	8314769	8268956
79	8423847	8424399	8428494	8353223	8360016	8322559	8325825	8350744	8279669
80	8399875	8505776	8367435	8431157	8376184	8334266	8402188	8308994	8294133
81	8422593	8378233	8406498	8344173	8371423	8319906	8303064	8282301	8262946
82	8431879	8405175	8413363	8339050	8410905	8314267	8369912	8285485	8287366
83	8411948	8419114	8413715	8386357	8326732	8413581	8308265	8238230	8248433
84	8461284	8420633	8417562	8403030	8390693	8311322	8355681	8296789	8323732
85	8447650	8430460	8416847	8348665	8346043	8293472	8342338	8306605	8297551
86	8405330	8386784	8388969	8385952	8400121	8396754	8413936	8342406	8273665
87	8381108	8429013	8419664	8376143	8420628	8431154	8289452	8311039	8277479
88	8476214	8381529	8419280	8338465	8357989	8323374	8331797	8289443	8273708
89	8391526	8483455	8450094	8377774	8362506	8346988	8351240	8291032	8322643
90	8472567	8452337	8363765	8436872	8341135	8340387	8299261	8326764	8280923
91	8385125	8412619	8399521	8382527	8341417	8362542	8313038	8243863	8273109
92	8397005	8447636	8400049	8354869	8358660	8377334	8302013	8384140	8274838
93	8394478	8405766	8352212	8362039	8408200	8344653	8262984	8358000	8318809
94	8414930	8432083	8430116	8428571	8414613	8297826	8302916	8333261	8266198
95	8397577	8408929	8461638	8358291	8413022	8300853	8304729	8290934	8324995
96	8417748	8447327	8412132	8366160	8339182	8409490	8398018	8334446	8284232
97	8428004	8447000	8353116	8450306	8302245	8334784	8306219	8303849	8364776
98	8417046	8407827	8392259	8392142	8359566	8293826	8326791	8313083	8270661
99	8454296	8355598	8413816	8388764	8324228	8326401	8339453	8317457	8286596
100	8411616	8422722	8449895	8447162	8336927	8367422	8278677	8301059	8303018

Bibliografia

[AL] Allsop, R.E., *SIGSET: A computer program for calculating traffic signal settings. Traffic Engineering and Control*, 1971, p. 58-60.

[AGG] Aggarwal, Ch. C. (2007) *Data streams. Models and algorithms, Springer, Heidelberg.*

[APK] Abdulhai, B., Pringle, R., Karakoulas G. J., *Reinforcement learning for true adaptive traffic signal control*, J. Transp. Engrg., 129(3), 2003: 278-285

[AWM] W.M.P. van der Aalst, A.J.M.M. Weijters, L. Maruster, *"Workflow Mining: Discovering process models from event logs"*, Knowledge and Data Engineering, IEEE Transactions on Volume 16, Issue 9, Sept. 2004 Page(s): 1128 - 1142.

[BHSS] R. Barlović, T. Huisinga, A. Schadschneider, M. Schreckenberg, (2005) *Adaptive Traffic Light Control in the ChSch Model for City Traffic Traffic and Granular Flow*, 2005, Part 3, p. 331-336.

[BJ] Branke, J. (2001) *Evolutionary optimization in dynamic environments.* Kluwer Academic Publishers Norwell, MA, USA.

[BU] D. Bullock, T. Urbanik *Traffic signal systems: addressing diverse technologies and complex user needs*. In: Transportation in the New Millennium. Transportation Research Board, 2000, 9 p.

[CA] Kułakowski K., *"Automaty komórkowe"*, http://www.ftj.agh.edu.pl/ kulakowski/AC/.

[CAG] Chen H., Abu-Lebdeh G., Goodman E.: *Optimizing Dynamic Roadway Traffic Control with Parallel Genetic Algorithms and Assessment of Computational Advantages*, Proceedings of the ASCE Civil Engineering Conference and Exposition, Baltimore, 2004.

[ChCh] S. Chiu, S. Chand, *Self-Organizing Traffic Control via Fuzzy Logic* In: Proc. 32nd IEEE Conf. on Decision & Control San Antonio, Texas - December, 1993, p. 1897-1902

[CYP] Chen S.W., Yang C.B., Peng Y.H., *Algorithms for the Traffic Light Setting Problem on the Graph Model*, Proc. of the 12th Conference on Artificial Intelligence and Applications, TAAI 2007.

[GOL] Goldberg D., *"Genetic algorithms"*, Addison Wesley, kierunku Matematyka na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego, 2009

[GOR] Gora P., *Modelowanie złożonych procesów na przykładzie symulacji ruchu drogowego*, praca magisterska na kierunku Matematyka na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego, 2009

[GS] G. Gomes, A. Skabardonis, *Effectiveness of adaptive traffic control for arterial signal management*, 2009 California PATH Working Paper UCB-ITS-PWP-2009-2

[GUS] Główny Urząd Statystyczny, *„Rocznik statystyczny Polski 2008”*, http://www.stat.gov.pl.

[HLL] Helbing, D., Leammer, S., Lebacque, J-P. (2005) Self-organized control of irregular or perturbed network traffic. In: C. Deissenberg and R.F. Hartl (Eds.) Optimal Control and Dynamic Games, Advances in Computational Management Science, 2005, Volume 7, Part IV, Springer, 239-274.

[HL] Helbing, D., Leammer, S. (2008) *Self-control of traffic lights and vehicle flows in urban road networks.* Journal of Statistical Mechanics: Theory and Experiment 4: p. 3-34.

[MBY] Minai, A., Braha, D., Bar-Yam, Y. (2008) *Unifying themes in complex systems VI.* Proceedings of the Sixth International Conference on Complex Systems.Springer, Cambridge, MA, USA.

[MG] Mohammadpour, J., Grigoriadis, K.M. (Eds.) (2009) *Efficient Modeling and Control of Large-Scale Systems.* Springer, Heidelberg.

[MIC] Michalewicz Z. *Algorytmy genetyczne + struktury danych = programy ewolucyjne*. Warszawa: WNT, 1996.

[NS] Nagel K., Schreckenberg M., *“A cellular automaton model for freeway traffic”*, Journal de Physique, 1992, strony: 2221-2229.

[OB] de Oliveira, D., Bazzan, A.L.C. (2006) *Traffic lights control with adaptive group formation based on swarm intelligence.* In: M. Dorigo et al. (Eds.): ANTS 2006, LNCS 4150, Springer, Heidelberg, strony 520-521

[OSM] OpenStreetMap, http://www.openstreetmap.org.

[OTC] Organic traffic control, http://www.aifb.uni-karlsruhe.de/EffAlg/Projekt/otcqe/

[PRTB] Prothmann, H., Rochner, F., Tomforde, S., Branke, J., Christian Mueller-Schloer, Ch., Schmeck, H. (2008) *Organic Control of Traffic Lights.* In: C. Rong et al. (Eds.): ATC 2008, LNCS 5060, strony 219-233.

[RPBM] Rochner, F., Prothmann, H., Branke, J., Mueller-Schloer, C., Schmeck, H. (2006): *An organic architecture for traffic light controllers*. In: Hochberger, C., Liskowsky, R. (Eds.) Informatik 2006 - Informatik fuer Menschen. LNI, vol. P-93, Koellen Verlag, strony 120-127.

[RPS] Roess, R.P., Prassas, E.S., McShane, W.S. (2004) *Traffic Engineering, 3rd Edition*, Upper Saddle River: Pearson Prentice Hall.

[SCATS] http://www.scats.com.au/

[SCH] Schadschneider A., *“The Nagel-Schreckenberg model revisited”*, The European Physical Journal B, Volume 10, Issue 3, pp. 573-582 (1999).

[SIE] Siemens Traffic Solutions, `http://www.itssiemens.com/en/u_nav1241.html`

[STHB] Sha'Aban J., Tomlinson A., Heydecker B., Bull L., *Adaptive traffic control using evolutionary algorithms*, Proceedings 9th Meeting of the EURO Working Group on Transportation, Bari, Italy, 2002.

[TBPEGS] Taale, H., Baeck, Th., Preuss, M., Eiben, A. E., de Graaf, J. M., Schippers, C. A. (1998) *Optimizing traffic light controllers by means of evolutionary algorithms*. In: EUFIT'98.

[TELVENT] TELVENT, `http://www.telvent.com/en/business_areas/transportation/our_company/loader.cfm?csModule=security/getfile&pageid=6892`

[TLC] Traffic lights control University of Dresden, `http://www.trafficforum.ethz.ch/trafficlights/`

[TRB] TRB Traffic Signal Systems Committee, `http://www.signalsystems.org.vt.edu/archive.html`

[TSS] Tubaishat, M., Shang, Y., Shi, H. (2007) *Adaptive Traffic Light Control with Wireless Sensor Networks.* In: Proc. IEEE Consumer Communications and Networking Conference, Las Vegas, NV, January 2007.

[WF] Windows Forms, `http://windowsclient.net/`

[WI] Wiering, M. A. (2000). *Multi-agent reinforcement learning for traffic light control.* In Langley, P., editor, *Proceedings of the Seventeenth International Conference on Machine Learning* (ICML'2000), strony 1151-1158.

[WJK] Wiering, M., van Veenen Jilles, J., Koopman, V.A. (2004) *Intelligent Traffic Light Control.* Institute of Information and Computing Sciences, Utrecht University Technical Report UU-CS-2004-029.

[WKM] Washington, S.P., Karlaftis, M.G., Mannering, F.L. (2003) *Statistical and econometric methods for transportation data analysis.* Chapman & Hall/CRC, Boca Raton.

[WU] Wuertz, R.P. (Ed.) (2008) *Organic computing.* Springer, Heidelberg.