Akademia Górniczo-Hutnicza
im. Stanisława Staszica
w Krakowie

Praca magisterska

Modelowanie i optymalizacja ruchu miejskiego przy użyciu wybranych technik

Bartosz Rybacki

Kierunek: Informatyka
Specjalność: Systemy rozproszone i sieci komputerowe

Nr albumu: 117007

Promotor
dr inż. Jarosław Koźlak

Wydział EAIiE

Kraków 2008

Oświadczenie autora

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.

.....

(Podpis autora)

Spis treści

Spis tablic 4

Spis rysunków 5

Rozdział 1. Wprowadzenie 7

1.1. Motywacje 7

1.2. Temat i Cel pracy 8

1.3. Plan pracy 9

Rozdział 2. Przegląd dziedziny 10

2.1. Modele ruchu 11

2.1.1. Modele makroskopowe 12

2.1.2. Modele mezoskopowe 13

2.1.3. Modele mikroskopowe 13

2.1.4. Mikroskopowe modele sieci dróg 20

2.1.5. Ruch autostradowy i miejski 22

2.2. Sposoby monitorowania stanu ruchu drogowego 23

2.2.1. Monitorowane parametry 23

2.2.2. Źródła danych 24

2.2.3. Integracja informacji zwrotnej 25

2.3. Sygnalizacja świetlna 25

2.4. Koordynacja sterowania sygnalizacją świetlną 26

2.4.1. Synchronizacja zegarowa 27

2.4.2. TRANSYT i SCOOT 28

2.4.3. SOTL 29

2.4.4. RL — Reinforcement Learning 31

2.4.5. Algorytm OptAPO 32

2.5. Inne algorytmy i metody optymalizacji ruchu 34
2.5.1. Znaki o zmiennym tekście — VMS 34
2.5.2. Modyfikacja sieci 36
2.5.3. Priorytety dla komunikacji masowej 36
2.5.4. Podsumowanie 37
Rozdział 3. Koncepcja systemu 38
3.1. Cel projektu 38
3.2. Wymagania i specyfikacja 39
3.2.1. Wymagania funkcjonalne 39
3.2.2. Przyjęte ograniczenia 41
3.3. Model miasta 42
3.4. Model symulacji ruchu 44
3.5. Algorytmy sterowania sygnalizacją świetlną 45
3.5.1. Stała konfiguracja 45
3.5.2. SOTL 46
3.5.3. RL 47
3.6. Algorytmy współpracy sygnalizatorów na skrzyżowaniach 48
3.7. Predykcja wzorców zatłoczenia 48
3.7.1. Wzorzec zatłoczenia 49
3.7.2. Wykrywanie zatoru oraz identyfikacja wzorca problemu 49
3.8. Obserwowane parametry modelu 51
3.9. Podsumowanie koncepcji 52
Rozdział 4. Projekt systemu 54
4.1. Modułowa Architektura 55
4.2. Podstawowe elementy systemu 58
4.2.1. Klasy tworzące sieć dróg 59
4.2.2. Interfejsy modułów 61
4.2.3. Inne interfejsy 67
4.2.4. Jądro a moduły 68
4.3. System symulacji — części składowe 69
4.3.1. Jądro symulacji 70
4.3.2. Moduły rozszerzające 71
4.4. Dynamika systemu symulacji 71
4.4.1. Inicjalizacja części składowych 72
4.4.2. Start symulacji 73
4.4.3. Pętla symulacyjna 74

4.5. Konkretne moduły . . . 77
4.5.1. Moduł fizyczny . . . 77
4.5.2. Moduł decyzyjny . . . 81
4.5.3. Moduł oceny SOTL - Self Organizing Traffic Lights . . . 81
4.5.4. Moduł oceny RL - Reinforcement Learning . . . 82
4.5.5. OptAPO . . . 83
4.5.6. Generator ruchu . . . 85
4.5.7. Router . . . 85

Rozdział 5. Eksperymenty . . . 87

5.1. Topologie skrzyżowań . . . 88
5.1.1. Model skrzyżowania typu T . . . 88
5.1.2. Model skrzyżowania drogi głównej z drogą podrzędną . . . 95
5.2. Prosty model miasta . . . 100
5.2.1. Wpływ natężenia ruchu na średnią prędkość . . . 102
5.2.2. Porównanie do poprzedniej wersji systemu KRAKSIM . . . 105
5.2.3. Modyfikacja sieci drogowej . . . 108
5.3. Różne metody sterowania sygnalizacją . . . 111
5.4. Wpływ systemu informacji o korkach . . . 115
5.4.1. Kratowy model sieci – *manhattan* . . . 116
5.4.2. Uproszczony model miasta . . . 118

Rozdział 6. Podsumowanie . . . 121

Bibliografia . . . 123

Dodatek A. Instrukcja obsługi . . . 125

A.1. Zawartość dysku CD . . . 125
A.2. Instalacja . . . 125
A.3. Program . . . 126
A.3.1. Uruchamianie . . . 126
A.3.2. Obsługa interfejsu graficznego . . . 128
A.3.3. Formaty plików wejściowych . . . 129
A.3.4. Format pliku statystyk . . . 135

Spis tablic

5.1. Model miasta. Długości dróg. 101
5.2. Wpływ natężenia ruchu na średnią prędkość, metoda SOTL. 103
5.3. Wpływ natężenia ruchu na średnią prędkość, metoda RL. 103
5.4. Szczegółowe porównanie wyników eksperymentu dla dwóch wersji systemu. . . . 106
5.5. Porównanie wersji 2006 i 2007; metoda SOTL. 107
5.6. Porównanie wersji 2006 i 2007; metoda RL. 107
5.7. Wpływ nowej drogi w sieci na przebieg symulacji 109
5.8. Wpływ nowej drogi w sieci na średnią prędkość w centralnej części modelu . . . 110
5.9. Schemat ruchu nr 1, dominujące strumienie wschód-zachód. 112
5.10. Schemat ruchu nr 2, dominujące strumienie północ-południe. 113
5.11. Schemat ruchu nr 3, charakter ruchu zmienny, trudno wyróżnić globalny kierunek dominujący. 113
5.12. Porównanie metod sterowania sygnalizacją. 114

Spis rysunków

2.1. Model ruchu — jednowymiarowy automat komórkowy. 15

3.1. Elementy składowe modelu miasta 43
3.2. Komórkowo-turowy charakter symulacji 44

4.1. Ogólna architektura systemu, przykładowa konfiguracja budowy systemu z podsystemów. 54
4.2. Ilustracja pojęć 57
4.3. Koncepcja współdziałania rozszerzeń pewnego elementu 57
4.4. Diagram podstawowych klas jądra systemu 59
4.5. Klasy elementów: węzeł, skrzyżowanie, węzeł we/wy, faza cylku sygnalizacji. . . 60
4.6. Klasy elementów: miasto, połączenie, pas. 61
4.7. Interfejs modułu symulacji. 62
4.8. Interfejs modułu informacji o pojazdach. 63
4.9. Interfejs modułu monitoringu. 64
4.10. Interfejs blokowania pasów. 65
4.11. Interfejs oceny kosztu zmiany sygnalizacji. 66
4.12. Interfejs decyzyjny. 67
4.13. Przykład architektury - interfejs oceny, moduł SOTL 68
4.14. Przykład architektury - interfejs decyzyjny, prosty moduł decyzyjny 69
4.15. Przykład architektury - diagram obiektów 69
4.16. Architektura symulatora 70
4.17. Współpraca części: Inicjalizacja części składowych 72
4.18. Współpraca części: Start symulacji 73
4.19. Współpraca części: Pętla symulacyjna - rozpoczęcie zaplanowanych podróży . . 74
4.20. Współpraca części: Pętla symulacyjna - symulacja tury jazdy samochodów . . . 75
4.21. Współpraca części: Pętla symulacyjna - uaktualnienie oceny pasów 76

4.22. Współpraca części: Pętla symulacyjna - podjęcie decyzji o stanie pasów 76
4.23. Reguły pierwszeństwa na skrzyżowaniu. 79
4.24. Moduł koordynacji pracy skrzyżowań 84

5.1. Skrzyżowanie T: model podstawowy oraz model z pasami do skrętu w lewo. . . . 89
5.2. Skrzyżowanie T: modyfikacje modelu podstawowego polegające na użyciu rampy zjazdowej. 90
5.3. Program sygnalizacji świetlnej na skrzyżowaniu T. 90
5.4. Skrzyżowanie T: ilość przejazdów przez skrzyżowanie 93
5.5. Skrzyżowanie T: przepustowość i pojemność modelu. 94
5.6. Skrzyżowanie T: średnia prędkość i liczba przejazdów. 94
5.7. Skrzyżowanie X: model podstawowy i modyfikacje. 96
5.8. Program sygnalizacji świetlnej na skrzyżowaniu X. 97
5.9. Skrzyżowanie X: ilość przejazdów przez skrzyżowanie. 98
5.10. Skrzyżowanie X: przepustowość i pojemność modelu. 99
5.11. Skrzyżowanie X: średnia prędkość i liczba przejazdów. 99
5.12. Standardowy model miasta 101
5.13. Porównanie wersji 2006 i 2007; metoda SOTL. 104
5.14. Porównanie wersji 2006 i 2007; metoda RL. 105
5.15. Rozszerzony model miasta 109
5.16. Średnia prędkość przejazdu przez konkretne połączenia. 110
5.17. Model manhattan. 112
5.18. System informacji o zatłoczeniach: prędkość i czas symulacji, model manhattan . 116
5.19. System informacji o zatłoczeniach: średni czas trwania podróży i jej długość, model manhattan 117
5.20. System informacji o zatłoczeniach: prędkość i czas symulacji, model miasta . . . 119
5.21. System informacji o zatłoczeniach: średni czas trwania podróży i jej długość, model miasta 119

A.1. Konfiguracja plików wejściowych 128
A.2. Główne okno wizualizacji przebiegu symulacji 129

Rozdział 1

Wprowadzenie

1.1. Motywacje

Modelowanie i symulacja ruchu drogowego ma długie tradycje. Pierwsze próby matematycznego opisania ruchu na autostradach w Stanach Zjednoczonych Ameryki podejmowano w latach 30-tych ubiegłego wieku. Dokładne badania zostały przeprowadzone w latach 50-tych i 60-tych tego samego stulecia[1].

Obecnie, dzięki coraz dokładniejszym modelom symulacji ruchu, można uniknąć wielu błędów konstrukcyjnych przy projektowaniu sieci drogowej. Odpowiednie projekty sieci dróg są przygotowywane jako wirtualne modele. Symulacja ruchu umożliwia zbadanie podstawowych własności wirtualnego modelu w warunkach, w jakich byłby on użytkowany w rzeczywistości. Takie podejście umożliwia wykonanie kilku projektów i wybranie najlepszego.

Innym zastosowaniem dla modeli i symulacji ruchu jest prognozowanie. Taka symulacja działa w trybie ciągłym bazując na aktualnych danych o ruchu drogowym i oferuje

[1] np. publikacja Lightill'a i Witham'a z 1955 r., w 1966 powstało czasopismo naukowe Transportation Science.

prognozę sytuacji w najbliższej przyszłości[2], co umożliwia optymalizację przepustowości sieci drogowej oraz zapobieganie sytuacjom kryzysowym zanim jeszcze nastąpią.

1.2. Temat i Cel pracy

Tematem pracy dyplomowej jest *Modelowanie i optymalizacja ruchu drogowego przy użyciu wybranych technik*. Wybór tematu został zainspirowany zakończeniem prac nad prototypem symulacji ruchu drogowego KRAKSIM.

System KRAKSIM został zbudowany w ramach przedmiotów *Systemy Agentowe* oraz *Technologie Agentowe* w roku 2005/2006 przez studentów: Wojciecha Matyjewicza i Bartosza Rybackiego. Podczas dwusemestralnej pracy nad prototypem udało się opracować modułową koncepcję budowy symulatora/optymalizatora ruchu miejskiego oraz zaimplementować: model sieci drogowej, model ruchu, sterownik sygnalizacji świetlnej, dwie wersje modułu oceny sytuacji na skrzyżowaniu oraz moduł decyzyjny (sterujący sygnalizatorem na podstawie oceny). Optymalizacja polegała na odpowiednim sterowaniu sygnalizacją świetlną. Zaimplementowano w tym celu dwie metody: SOTL oraz RL[3]. Model symulacji systemu KRAKSIM wymagał poprawek, jednak sam system był dobrym punktem wyjściowym dla realizacji tematu pracy magisterskiej.

Autor pracy postawił sobie następujące cele:

- modyfikacja pilotowej wersji oprogramowania w celu zbliżenia zachowania się ruchu drogowego do rzeczywistości,
- poprawa przepustowości wirtualnej sieci drogowej,
- testy porównawcze różnych topologii skrzyżowań,
- opracowanie i implementacja nowych metod sterowania sygnalizacją świetlna,
- porównanie metod sterowania sygnalizacją świetlną,
- analiza wykorzystania znaków o zmiennej treści do optymalizacji ruchu drogowego.

Aby zrealizować stawiane cele, zdecydowano się wybrać następujące techniki: automaty komórkowe (modelowanie ruchu), system agentowy (optymalizacja sygnalizacji), uczenie ze wzmocnieniem (optymalizacja sygnalizacji), algorytmy wyszukiwania ścieżki w grafie (optymalizacja trasy).

Językiem implementacji systemu KRAKSIM jest JAVA.

W pracy przedstawiono stronę teoretyczną wybranych metod optymalizacji sytuacji drogowej, ich implementację w systemie *KRAKSIM* oraz serię eksperymentów, które mają zweryfikować skuteczność wybranych metod.

[2] np.: OLSIM, region Nadrenia-Pólnocna Westwalia, prognozy na 30 oraz 60 minut, strona internetowa z aktualną prognozą: www.autobahn.nrw.de

[3] Self Optimizing Traffic Lights punkt 2.4.3 oraz Reinforcement Learning punkt 2.4.4

1.3. Plan pracy

Praca podzielona jest na pięć części. Pierwszy rozdział zawiera wprowadzenie.

Drugi rozdział stanowi przegląd dziedziny problemu od strony teoretycznej, przedstawia różne sposoby modelowania ruchu, monitorowania jego parametrów oraz przedstawia rozwiązania stosowane w rzeczywistych sieciach drogowych.

Trzeci rozdział to koncepcja systemu. Zawiera on dokładny opis celu, który chciano osiągnąć oraz proponowane rozwiązania, które są stosowane, aby ten cel zrealizować. W tym rozdziale znajdują się: specyfikacja wymagań dla systemu, opis modelu miasta, opis sposobu symulacji ruchu. Na końcu rozdziału opisane są proponowane metody sterowania sygnalizacją świetlną oraz system predykcji zatłoczenia, który ma zapobiegać powstawaniu zatorów.

W czwartym rozdziale znajduje się opis techniczny zrealizowanego systemu informatycznego, który powstał w ramach tej pracy magisterskiej. Ten rozdział wyjaśnia w jaki sposób system został zaprojektowany oraz jak zaimplementowano rozwiązania wybrane w fazie analitycznej. Opisano ogólną architekturę systemu, podział na moduły, podstawowe elementy systemu oraz współdziałanie modułów w trakcie trwania symulacji.

Piąty rozdział zawiera wyniki z eksperymentów oraz ich analizę. Jest to najważniejsza części pracy pisemnej, ponieważ pozwala zweryfikować zachowanie systemu symulacji, gdy stosowane są różne metody optymalizacji. Krótko mówiąc, ten rozdział pozwala ocenić poziom realizacji celów.

Rozdział 2

Przegląd dziedziny

Pierwsze samochody przypominające te, które obecnie spotyka się na drogach, powstały w ostatnim dwudziestoleciu XIX wieku. Od tamtej pory minęło już grubo ponad 100 lat i patrząc z tej perspektywy można stwierdzić, że nastanie ery motoryzacji wywarło na świat tak duży wpływ, jak mało które inne wydarzenie.

Ilość samochodów na drogach rosła lawinowo od samego początku (obecnie w Polsce zarejestrowanych jest ponad 16 milionów pojazdów)[1]. W drugiej połowie XX wieku zaczęto dostrzegać negatywne skutki gwałtownego rozwoju motoryzacji. Wśród nich można wymienić negatywny wpływ rozwoju transportu zarówno na ludzi, jak i środowisko: zwiększony poziom zanieczyszczeń oraz hałasu w pobliżu dróg, intensywna eksploatacja złóż surowców energetycznych to tylko niektóre problemy.

Nieproporcjonalny, w stosunku do długości dróg dla nich przeznaczonych, wzrost ilości samochodów jest problemem również dla samego transportu. Dzisiaj wielokilometrowe korki na autostradach, nieprzejezdne arterie miejskie nie dziwią już specjalnie nikogo. Jest to bardzo niekorzystne z punktu widzenia ekonomicznego, ponieważ transport staje się mniej efektywny i rosną jego koszty.

[1] Publikacja „Transport — Wyniki działalności 2005" dostępna pod adresem internetowym: http://www.stat.gov.pl/cps/rde/xbcr/gus/PUBL_transport_results_2005.pdf

W przeszłości stosowano głównie analityczne podejście do predykcji ruchu drogowego i planowania jego organizacji. Obecnie, ze względu na ogromną złożoność problemu oraz powszechną dostępność dużej mocy obliczeniowej, stosuje się przede wszystkim metody symulacyjne.

Podczas tworzenia praktycznej symulacji należy odpowiednio przypisać wagi do wykluczających się (ze względów efektywności) nawzajem celów: rozdzielczości, dokładności i skali. Rozdzielczość odnosi się do rozmiaru najmniejszych rozpatrywanych jednostek, dokładność do stopnia realizmu modelowania tych jednostek, a skala do rozmiaru problemu.

Obecnie najlepsze rezultaty otrzymywane są w modelach o minimalnej możliwej rozdzielczości, czyli takich, które symulują zachowanie pojedynczych podróżnych[2]. Ponieważ głównymi scenariuszami symulacji są duże aglomeracje, konieczne jest poświęcenie dokładności, aby osiągnąć rozsądne czasy obliczeń.

2.1. Modele ruchu

Modele ruchu dzielimy na statyczne i dynamiczne. Modele statyczne uśredniają wartości oceny stanu ruchu drogowego w czasie. W modelach dynamicznych stan sytuacji drogowej zmienia się w czasie. Modele symulacyjne są dynamiczne. Symulacyjne modele ruchu samochodowego można sklasyfikować ze względu na poziom szczegółowości:

- **Makroskopowe** - wysoki poziom abstrakcji, ruch uogólniony np.: do przepływu cieczy przez rurę;
- **Mezoskopowe** - średni poziom abstrakcji, można wyodrębnić pojedyncze pojazdy, modeluje się wspólne zachowanie, trudno przytoczyć formalną definicję, jest to model o poziomie szczegółowości pomiędzy mikro i makroskopowym;
- **Mikroskopowe** - niski poziom abstrakcji, modeluje się dokładne zachowania pojedynczych pojazdów (często również innych uczestników ruchu, np.: pieszych).

Teoria przepływu jest związana z poszukiwaniem zależności pomiędzy trzema podstawowymi zmiennymi opisującymi ruch: prędkością ν, gęstością ρ, przepustowością j[3]. Pierwszym zadaniem teorii przepływu było znalezienie niezależnego od czasu związku pomiędzy wyżej wymienionymi zmiennymi. Dopiero później wprowadzono opis dyna-

[2] np.: systemy OLSIM oraz TRANSIMS są systemami gdzie najmniejszą jednostką jest pojedynczy pojazd

[3] ilość samochodów przejeżdżających przez przekrój drogi w jednostce czasu

miczny. Do dzisiejszego czasu powstało wiele modeli wykorzystujących podobieństwa ruchu drogowego do zjawisk fizycznych, takich jak na przykład dynamika płynów.

Teoria podążania za poprzednikiem opisuje ruch z bardziej mikroskopowego punktu widzenia. Zachowanie kierowcy jest determinowane przez pojazd znajdujący się bezpośrednio przed nim. Takie założenia prowadzą do równań prędkości zależnych od odległości i prędkości poprzedzającego pojazdu. Do matematycznego modelu tego typu najczęściej wykorzystuje się równania różniczkowe drugiego rzędu. W implementacji najczęściej stosuje się modele oparte na automatach komórkowych.

2.1.1. Modele makroskopowe

Podejście do analizy i symulacji ruchu w skali makro pomija szczegóły dotyczące ruchu pojedynczych pojazdów, a ruch wielopasmowy przybliża się uogólnionym strumieniem. Pojazdy na pasach przedstawia się tylko za pomocą pewnych abstrakcyjnych wartości (np. ilość pojazdów na minutę). W ten sposób model staje się prostszy w budowie. Dzięki uproszczeniom można prowadzić symulację na większych sieciach, jednak traci się pewną szczegółowość.

Typy modeli symulacji makroskopowej:

- Równania różniczkowe kinematyki gazów (np.: Prigogine i Herman[19])
- Równania różniczkowe dynamiki płynów (np.: Lighthill, Witham i Richards[14, 20])

W ruchu drogowym można zaobserwować występowanie fal analogicznych do fal mechanicznych. Niech kilkanaście samochodów czeka na skrzyżowaniu na zapalenie się zielonego światła. Zielone światło zapala się, rusza pierwszy samochód, ale drugi stoi jeszcze czekając aż odjazd pierwszego stworzy odpowiednio dużą lukę, trzeci czeka na odjazd drugiego itd. Równocześnie na koniec tej kolejki mogą przybywać nowe pojazdy. Jeśli ilość samochodów dojeżdżających w jednostce czasu będzie nie mniejsza niż ilość samochodów odjeżdżających, to korek (zarówno jego czoło jak i ogon) zacznie się przesuwać w kierunku przeciwnym do ruchu pojazdów. Oczywiście to nie samochody przesuwają się w tył. Kierowca dołącza się na koniec korka, który „przesuwa się przez niego", po czym może ruszyć dalej.

Podczas analizy tego zjawiska, używając analogii praw fizyki, można wyjść od standardowego równania ciągłości, które jest spełnione tak długo, jak ruch spełnia warunek zachowania masy (żadne pojazdy nie dołączają się i nie opuszczają systemu):

$$\partial_t K + \partial_x Q = 0$$

gdzie x - pozycja, t - czas, K - gęstość/natężenie ruchu, Q - przepływ. Przy dyskretyzacji ze stałymi $\Delta t = 1$ i $\Delta x = 1$ otrzymuje się[6]:

$$K_{t+1}(x) = K_t(x) - (q_t(x + \frac{1}{2}) - q_t(x - \frac{1}{2})) = K_t(x) + q_t(x - \frac{1}{2}) - q_t(x + \frac{1}{2})$$

gdzie $K_t(x)$ to ilość samochodów na odcinku o długości $\Delta x = 1$, a $q_t(x)$ to przepływ przez przekrój w miejscu o współrzędnej x w czasie $\Delta t = 1$.

Powyższe równanie opisuje to, że ilość pojazdów na pasie w chwili $t+1$ (czyli K_{t+1}) jest równa ilości samochodów w chwili t (K_t) plus dopływ pojazdów z tyłu ($q_t(x - \frac{1}{2})$) i odpływ z przodu ($q_t(x + \frac{1}{2})$). W tej pracy nie zdecydowano się na szczegółową analizę modeli makroskopowych.

2.1.2. Modele mezoskopowe

Ten typ symulacji ma średni poziom szczegółowości w porównaniu z modelami mikro i makro. Podobnie jak w modelu mikroskopowym analizuje się pojedyncze pojazdy, jednak ich zachowanie na pasach jest uogólnione.[4] Można wyróżnić przynajmniej dwa typy modeli mezoskopowych. Istnieją modele, w których grupy pojazdów łączy się w pakiety. Inne modele uogólniają zachowanie pojazdu na pasie do wartości czasu przejazdu (wyznaczonego stochastycznie). Traktując pojazd w ten sposób, całkowicie eliminuje się symulację ruchu pojazdu po pasie i analizuje się jedynie zachowanie pojazdów na skrzyżowaniu. Oczywiście problemem pozostaje odpowiednie wyznaczenie czasu przejazdu. Więcej o modelach tego typu można znaleźć w drugim rozdziale publikacji [15], oraz na stronach producentów tych modeli (np.: INTEGRATION[5] DYNASMART[6], DINOSAUR[7], METS[8], TRANSMODELER[9]).

2.1.3. Modele mikroskopowe

Tutaj, przeciwnie do podejścia makroskopowego, skupia się dużą uwagę na odwzorowaniu szczegółów. Modeluje się zachowanie pojedynczych jednostek, w tym przypadku pojazdów. Dokładnie buduje się modele skrzyżowań, wraz ze znakami i sygnałami zmieniającymi zachowanie pojazdów. W ten sposób można zaobserwować pewne lokalne własności, których nie da się sprawdzić w modelu makroskopowym.

[4] modele mezoskopowe buduje się właśnie na bazie modeli mikroskopowych, chociaż trudno jest ustalić sztywne granice dla określenia *mezoskopowe*

[5] INTEGRATION — http://filebox.vt.edu/users/hrakha/Software.htm

[6] DYNASMAR — http://mctrans.ce.ufl.edu/featured/dynasmart/

[7] DINOSAUR (Dynamic Information Network Optimizer for System and User Requirements)

[8] METS (Mesoscopic Event-based Traffic Simulator)

[9] TRANSMODELER — http://www.caliper.com/transmodeler/Simulation.htm

Wzajemne oddziaływania pojedynczych elementów modelu wpływają na stan i zachowanie całego systemu. Modele tego typu są modelami złożonej dynamiki. Nawet drobne losowe zmiany w modelu zbudowanym jako automat komórkowy mogą wpłynąć na stabilność symulowanego systemu (np.: przejście ze stanu płynnego przepływu do turbulencji).

Modele mikroskopowe ruchu można łatwo integrować z narzędziami do zarządzania infrastrukturą sygnalizacji świetlnej UTC[10]. Dane o pojazdach są bezpośrednio pobierane z modelu, a dane o ustawieniach sygnalizacji są przekazywane przez system UTC do modelu symulacji, w taki sam sposób jak do prawdziwego sterownika świateł. Modele mikroskopowe (a w szczególności te zrealizowane jako automaty komórkowe i systemy agentowe) są dobrze przystosowane do symulacji komputerowych dużej skali. Dzięki tym właściwościom modele mikroskopowe, lepiej niż modele makroskopowe, nadają się do integracji z systemami ITS[11]. Warto zaznaczyć, że CORSIM (należący do standardowych narzędzi używanych przez departament komunikacji w USA[12]) używa modelu mikrosymulacji.

Na potrzeby symulacji mikroskopowej należy stworzyć wiele modeli zachowań: model podążania za pojazdem, model zmiany pasów ruchu, model pierwszeństwa przejazdu. Główne zalety modeli mikroskopowych:

- umożliwiają symulację i szczegółową analizę *zachowania* jednostki,
- umożliwiają planowanie trasy pojedynczego pojazdu, co wpływa na rozkład ruchu w sieci komunikacyjnej.

Od roku 1992 powszechne stało się użycie automatów komórkowych[13]. Panowie Nagel i Schreckenberg zaprezentowali model dla ruchu w jednym wymiarze [17], który bardzo dobrze odzwierciedla rzeczywistość. Ich model, z pewnymi modyfikacjami, jest stosowany do dziś, nawet w największych komercyjnych aplikacjach (TRANSIMS[14], OLSIM[15]). Model NaSch uznawany jest za model podstawowy.

Model *Nagla-Schreckenberga* (NaSch[17]). Ten najbardziej rozpowszechniony

[10] Urban Traffic Control — system sterowania ruchem miejskim, głównie sygnalizacją świetlną, np.: SCOOT, SCATS, UTOPIA, PRODYN

[11] Intelligent Transportation System — Inteligentny System Transportu, to ogólnie system transportu (infrastruktura drogowa oraz pojazdy),w którym zastosowano nowoczesne technologie informatyczne w celu poprawy efektywności i bezpieczeństwa tego systemu

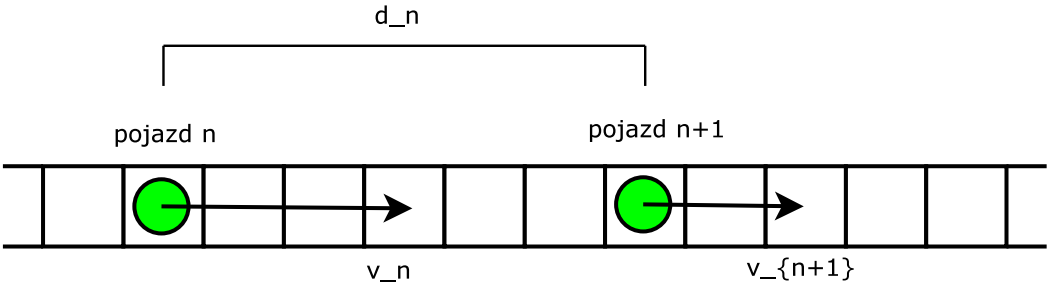
[12] CORSIM — http://ops.fhwa.dot.gov/trafficanalysistools/corsim.htm

[13] Automaty komórkowe — CA, z ang. Cellular Automata

[14] otwarta wersja symulatora TRANSIMS jest dostępna w internecie od 2006-go roku — http://transims-opensource.net/

[15] symulacja ruchu dla regionu Nadrenia-Północna Westfalia przy użyciu OLSIM

w praktyce model, jest również jednym z prostszych. Mimo swojej prostoty pozwala symulować wiele rzeczywistych problemów komunikacyjnych.



Rysunek 2.1. Model ruchu — jednowymiarowy automat komórkowy.

Droga reprezentowana jest jako tablica komórek, które mogą być puste lub zajęte przez dokładnie jeden pojazd n. Pojazd posiada parametr v_n, tj. prędkość z zakresu od 0 do V_{max}. Wszystkie parametry w tym modelu, takie jak odległość i prędkość, są dyskretne. Pozwala to tworzyć bezpośrednie i wydajne symulacje oraz upraszcza obliczenia. Prędkość wyrażana jest w komórkach na jednostkę czasu (na turę). Tura składa się z czterech kroków obliczeniowych, które dobrano tak, aby nie dochodziło do kolizji pojazdów jadących jedną drogą. Pojazdy w opisach numerowane są w kierunku jazdy, pojazd $n+1$ jedzie przed pojazdem n.

I. *Przyspieszanie*: jeśli prędkość v jest mniejsza od V_{max}, a odległość d_n do pojazdu $n+1$ znajdującego się bezpośrednio przed analizowanym pojazdem jest większa niż $v_n + 1$, to prędkość pojazdu n wzrasta o jeden.
II. *Zwalnianie*: jeśli odległość d_n do następnego pojazdu jest mniejsza niż $v_n + 1$ to prędkość jest zmniejszana do wartości $d_n - 1$.
III. *Czynnik losowy*: wartość prędkości v_n pojazdu n jest zmniejszana o 1 z prawdopodobieństwem p.
IV. *Ruch*: mija tura i każdy pojazd przesuwa się o v_n pól do przodu.

Po przekroczeniu pewnej krytycznej gęstości (ilość pojazdów na odcinku drogi) zmniejszają się odległości między pojazdami i zwiększają się między nimi oddziaływania. Jeśli jeden pojazd jest blisko drugiego i zwalnia, to wszystkie pojazdy za nim będą kolejno zwalniały. To jest reakcja łańcuchowa i powoduje powstanie fali zwalniających pojazdów, która przesuwa się odwrotnie do kierunku ruchu. Jest to zjawisko określane przez naukowców jako *„wędrujące korki”*.

Zachowanie algorytmu może być modyfikowane przez odpowiedni dobór parametrów p oraz V_{max}. Parametr p wpływa na zachowanie się modelu w stanie dużego

zagęszczenia ruchu, zmieniając sposób hamowania. Natomiast parametr V_{max} reguluje szybkość i zachowanie się samochodów w stanie swobodnego przepływu.

Przy odpowiednim doborze parametrów ten algorytm bardzo dobrze opisuje ruch o małej gęstości. Jeśli chodzi o rejony o dużym zagęszczeniu, to algorytm niedokładnie odwzorowuje fazy i rodzaje korków, a w szczególności brakuje fazy *ruchu synchronicznego*[16]. Może to być problemem dla długich pasów autostrady (ze zjazdami). W ruchu miejskim, gdzie odcinki są krótkie a światła na skrzyżowaniach mają dominujący wpływ na zachowanie się modelu, brak modelowania fazy ruchu synchronicznego nie jest problemem (patrz porównanie ruchu autostradowego i miejskiego, rozdział 2.1.5). Wadą tej metody jest niedokładna symulacja ruchu swobodnego. Jest to spowodowane dyskretyzacją czasu i przestrzeni. W takim ruchu, przy dużych prędkościach podróży, ważne są ułamkowe zmiany w prędkości pojazdu.

Randomizacja zależna od prędkości — ***VDR*[2]** (ang. *Velocity-dependent randomizations*) to bardziej realistyczny model ruchu drogowego bazujący na założeniach i teorii *Nagla-Schreckenberga*. W tym modelu wprowadzony jest parametr losowy zależny od prędkości $p(v)$. Parametr $p(v)$ jest podobny do parametru p z poprzedniego modelu i opisuje prawdopodobieństwo zmniejszenia prędkości. Obliczanie losowego parametru $p(v)$ następuje przed pierwszym krokiem algorytmu *Nagla-Schreckenberga*. Najprostsza wersja wprowadza rozdział parametru dla pojazdów jadących oraz stojących:

$$p(n) = \begin{cases} p_0 & \text{gdy } v = 0 \\ p & \text{gdy } v > 0 \end{cases}$$

Gdy $p_0 > p$ to jest to tzw. *reguła powolnego startu*. Jeżeli pojazd stoi, to jego prawdopodobieństwo zmniejszenia prędkości ma wartość p_0. Po pierwszym i drugim kroku symulacji zostanie ustalona możliwa prędkość jazdy (maksymalnie 1/s). W kolejnym kroku zastosowany jest czynnik losowy, który w tym wypadku ma wartość p_0. W tym przypadku prędkość może zostać zmniejszona do 0. Jeżeli $p(n)$ ma większą wartość dla pojazdów stojących, to częściej obserwuje się opóźnienia w ruszaniu pojazdu z miejsca.

Głównym osiągnięciem tego modelu jest poprawny opis „szerokich korków". Odpływ samochodów z korka jest mniejszy niż maksymalny przepływ, co powoduje, że taki korek jest stabilny i już się nie powiększa „do przodu" oraz nie zmniejsza się

[16] *Ruch synchroniczny* — wyodrębniana faza ruchu (pomiędzy korkami i ruchem luźnym), w której pojazdy poruszają się ze średnią prędkością mniejszą niż w fazie swobodnego przepływu ruchu, zagęszczenie pojazdów jest o wiele większe niż średnie, natomiast nie dochodzi do zaburzeń ruchu i trwałych korków. Pojazdy ściśle ze sobą oddziaływają co wymusza prawie synchroniczne ruchy. Więcej o stanach w ruchu drogowym można znaleźć w [10, 2, 13].

nienaturalnie szybko. Można obserwować jedynie opisane wcześniej przemieszczanie się korków w tył. Reguła powolnego startu powoduje całkowite rozdzielenie fazy swobodnego ruchu od fazy korków i zanik fazy ruchu synchronicznego.

Model *Światło Stopu* — *BL* (ang. *Brake Light*). Model *BL* łączy regułę powolnego startu z metodą przewidywania prędkości. Kierowcy zmieniają prędkość reagując na światła stopu pojazdów jadących przed nimi. Przykładowy parametr losowy w takim modelu może wyglądać następująco:

$$p(n) = \begin{cases} p_b & \text{gdy } b_{n+1} = 1 \text{ i } t_h < t_s \\ p_0 & \text{gdy } v_n = 0 \\ p_d & \text{w pozostałych przypadkach} \end{cases}$$

b_{n+1} $= 1$, oznacza, że pojazd $n+1$ hamuje,
t_s $= min\{v_n, h\}$, h - parametr określający próg odległości,
t_h $= \frac{d_n}{v_n}$, d_n to odległość pojazdu n od poj. $n+1$, v_n to prędkość pojazdu n

Wartość t_h to czas potrzebny, aby pojazd n osiągnął pozycję aktualnie zajmowaną przez pojazd $n+1$. Parametr t_s to, zależna od prędkości wartość, zapobiegająca reagowaniu na światła stop pojazdu, który jest bardzo daleko.

W powyższym wzorze p_0 określa regułę powolnego startu. Dla p_b parametr $t_h = \frac{d_n}{v_n}$ to iloraz odległości pojazdu n od pojazdu $n+1$ przez prędkość pojazdu n i oznacza czas w jakim pojazd n znajdzie się na pozycji zajmowanej teraz przez pojazd $n+1$. Natomiast $t_s = min\{v_n, h\}$ wprowadza ograniczenie, które zapobiega przed reakcją kierowcy na światła stop samochodu, który jest daleko z przodu. Parametr h umożliwia regulację odległości, od której kierowca może podjąć decyzję o reakcji na światła.

Załóżmy, że h ma dużą wartość. Jeśli kierowca jedzie ostrożnie (powoli) i zachowuje duży odstęp (ten odstęp jest większy od prędkości w jednostkach używanych w symulacji), to $t_h > t_s$ i kierowca nie reaguje na światła, nie reaguje w ogóle na pojazdy przed nim (podobnie jak w *VDR*). Jeżeli kierowca jedzie bardziej agresywnie lub sytuacja jest taka, że prędkość pojazdu jest wysoka, to zależność $t_h < t_s$ jest spełniona i kierowca powinien reagować na światła stopu pojazdu przed nim. Jeżeli zmniejszy się wartość parametru h, to zmniejsza się wartość parametru t_s, a tym samym zmniejsza się odległość, przy której kierowca reaguje na światła (co odpowiada bardziej agresywnemu stylowi jazdy w rzeczywistości).

Ten model, podobnie jak modele *NaSch* czy też *VDR*, ma cztery kroki na turę. Przed tymi krokami liczony jest parametr p. Następne 4 kroki są podobne do tych z algorytmu *Nagla-Schreckenberga*, z tą różnicą, że uwzględnia się tu zapalenie świateł

jeżeli prędkość spada oraz inny jest krok 3 — wprowadzenie czynnika losowego. W kroku trzecim prędkość pojazdu jest zmniejszana o 1 z prawdopodobieństwem p. Jeżeli samochód n zwalnia z powodu hamowania pojazdu $n+1$, to włącza światło stop, natomiast zwalnianie spowodowane wylosowaniem odpowiedniej wartości parametru p uznawane jest za spontaniczne wahania prędkości i nie włącza świateł stopu.

Jak pokazały przeprowadzone badania[9, 11], ten model dobrze oddaje charakterystykę trzech podstawowych faz ruchu:

- faza swobodnego przepływu ruchu,
- faza ruchu synchronicznego,
- faza korków.

Faza ruchu synchronicznego, która nie pojawia się w modelach *NaSch* oraz *VDR*, tutaj wykazuje zachowanie zgodne z charakterystyką uzyskaną empirycznie. Ta faza ruchu, bardzo ważna w analizie złożonych zjawisk przepływu ruchu drogowego, może być pominięta w rozważaniach dotyczących gęstej sieci dróg i skrzyżowań takich jak sieć miejska. Dodatkowo, gdy zastosuje się komórkę o długości kilkukrotnie krótszej od długości pojazdu, to przy małym natężeniu ruchu (swobodny przepływ) można zaobserwować bardzo małe odstępy czasowe[17]. Jest to element rzeczywistości, którego model *NaSch* nie potrafi odtworzyć. Zarówno *BL* jak i *VDR* to modyfikacje modelu *Nagla-Schreckenberga* wprowadzające nowe właściwości oraz więcej parametrów pozwalających lepiej dostroić model, aby lepiej odpowiadał rzeczywistym charakterystykom. Kolejne modele oraz ich porównania opisano w pozycji [13].

Wielopasmowy model *Nagla-Schreckenberga*. Model Nagla-Schreckenberga opisujący ruch na jednym pasie okazał się dobrym punktem wyjścia do budowy modeli opisujących zachowanie pojazdów na jezdni składającej się z wielu pasów. Jezdnię wielopasmową modeluje się jako zestaw równoległych modeli jednopasmowych z dodatkowymi regułami dotyczącymi zmiany pasów. Celem tych reguł jest zbadanie opłacalności zmiany pasa oraz sprawdzenie, czy zmiana ta jest możliwa. Sprawdzenie możliwości zmiany pasa naśladuje zachowanie kierowcy, tj. polega na zmierzeniu dystansów do najbliższych pojazdów z przodu i tyłu na docelowym pasie i odniesieniu ich do aktualnej prędkości.

Reguły dotyczące zmiany pasa są analizowane przed pierwszym krokiem podstawowego algorytmu. Można wyróżnić dwa, opisane poniżej, rodzaje zmiany pasa (pochodzą one od autorów systemu *TRANSIMS*[18, 12]).

[17] odstęp czasowy — czas potrzebny pojazdowi n na osiągnięcie pozycji aktualnie zajmowanej przez pojazd $n+1$

1. Pojazd może *zmieniać pas ruchu w celu wyprzedzenia*. Poniżej (algorytm 1) znajduje się przykład zalgorytmizowanego zestawu reguł dla tego przypadku.

Algorytm 1 Reguły zmiany pasa ruchu w celu wyprzedzania.

if pozycja $x_o(i)$ na sąsiednim pasie jest wolna **then**
  $gap(i) \leftarrow$ odległość do pojazdu z przodu na bieżącym pasie
  $gap_o(i) \leftarrow$ odległość do pojazdu z przodu na pasie docelowym
  $gap_b(i) \leftarrow$ odległość do pojazdu z tyłu na pasie docelowym
  if $(gap(i) < v(i)) \wedge (gap_o(i) > gap(i))$ **then**
    $weight1 \leftarrow 1$
  else
    $weight1 \leftarrow 0$
  end if
  $weight2 \leftarrow v(i) - gap_o(i)$
  $weight3 \leftarrow v_{max} - gap_b(i)$
  if $(weight1 > weight2) \wedge (weight1 > weight3)$ **then**
    zaznacz samochód do zmiany pasa
  end if
end if

Jeśli samochód jest zaznaczony do zmiany pasa, to nie oznacza jeszcze, że pas na pewno zmieni. Wprowadza się stałą p taką, że zmiana pasa przez zaznaczony samochód zachodzi z prawdopodobieństwem p. W przypadku, gdy zmiana pasów może zachodzić w obydwu kierunkach, konieczna jest odpowiednia synchronizacja tych zmian, aby dwa samochody nie zajęły w tym samym kroku tej samej komórki. Uzyskuje się to na przykład poprzez wykonywanie w nieparzystych turach zmiany pasów na te po lewej stronie, a w parzystych na te po prawej.

2. Innym celem zmiany pasa ruchu przez kierującego pojazdem, jest *ustawienie się na odpowiedniej pozycji do przejazdu przez skrzyżowanie*. W niektórych modelach symulacyjnych (m.in. w TRANSIMS) samochody podążają według ściśle ustalonej trasy. Jeśli model uwzględnia takie szczegóły, jak przypisanie pasom ruchu dalszych kierunków, w których jazdę można kontynuować wjeżdżając z tego pasa na skrzyżowanie, to konieczne jest takie sterowanie pojazdami, aby ustawiały się na odpowiednim pasie. Algorytm 2 uzyskano poprzez modyfikację kroku decyzyjnego w algorytmie 1 (przedstawiony w poprzednim podpunkcie).
   W algorytmie nr 2, d^{ch} to stała, którą można interpretować jako odległość od skrzyżowania, przy której kierowca zaczyna rozpatrywać zmianę pasa, a d to

Algorytm 2 Reguły zmiana pasa ruchu w celu ustawienie się na odpowiedniej pozycji do przejazdu przez skrzyżowanie.

if pozycja $x_o(i)$ na sąsiednim pasie jest wolna **then**
  $gap(i) \leftarrow$ odległość do pojazdu z przodu na bieżącym pasie
  $gap_o(i) \leftarrow$ odległość do pojazdu z przodu na pasie docelowym
  $gap_b(i) \leftarrow$ odległość do pojazdu z tyłu na pasie docelowym
  if $(gap(i) < v(i)) \wedge (gap_o(i) > gap(i))$ **then**
    $weight1 \leftarrow 1$
  else
    $weight1 \leftarrow 0$
  end if
  $weight2 \leftarrow v(i) - gap_o(i)$
  $weight3 \leftarrow v_{max} - gap_b(i)$
  $weight4 \leftarrow max[(d^{ch} - d)/(v_{max}), 0]$
  if $(weight1 + weight4 > weight2) \wedge (weight1 + weight4 > weight3)$ **then**
    zaznacz samochód do zmiany pasa
  end if
end if

aktualna odległość od skrzyżowania. Im bliżej skrzyżowania, tym waga $weight4$ jest większa, a tym samym mniej przestrzegane są reguły dotyczące minimalnej odległości do samochodów z przodu i z tyłu na docelowym pasie. W skrajnym przypadku kierowca „wpycha" się na inny pas, gdy tylko będzie miał jedną komórkę wolną.

2.1.4. Mikroskopowe modele sieci dróg

Praktyczny model sieci dróg typowo składa się z następujących modułów:

- Generacja ruchu. Wykorzystuje dane demograficzne i socjologiczne, aby oddać typowe cele podróży mieszkańców w zależności od pory roku (wakacje, pozostała część roku), dnia tygodnia (dni powszednie, dni wolne od pracy), godziny (dojazd do pracy, powrót z pracy, robienie zakupów itp.). W idealnym przypadku, model powinien posiadać dane na temat zachowania każdego indywidualnego mieszkańca, ale wydaje się to niemożliwe do zgromadzenia. Obecnie w modelach wykorzystuje się dane demograficzne, ale o znacznie większej granulacji, przyporządkowuje się zachowania i prawdopodobne cele podróży całym skupiskom ludności. Dane

szczegółowe uzyskuje się na podstawie interpolacji wyników badań ruchu. Badania ruchu, oprócz liczby pojazdów na poszczególnych drogach, dostarczają informacji o celach podróży oraz preferowanych trasach (te dane uzyskuje się przeprowadzając ankiety).

- Wybór trasy. Zajmuje się opracowanej trasy, tj. listy kolejnych ulic przez jakie pojazd musi przejechać w drodze do celu.
- Moduł mikrosymulacji ruchu. Składa się z wielu opisanych modeli jedno- lub wielopasmowego odcinka oraz modeli skrzyżowań, świateł itp. Zajmuje się realizacją przejazdu przez trasy stworzone przez dwa poprzednie moduły.
- Moduł informacji zwrotnej. Dla symulacji, których stan ma odzwierciedlać stan rzeczywisty na drogach, konieczna jest infrastruktura (sieć pętli indukcyjnych) pozwalająca na integrację rzeczywistych danych ze stanem modelu „wirtualnego". Na podstawie wartości natężenia ruchu zmierzonych przez pętle indukcyjne na poszczególnych odcinkach symulacja może dokonywać korekty swojego stanu.

Rozbudowanie modelu do sieci dróg wymaga uwzględnienia budowy jezdni, a także występowania skrzyżowań (ze światłami i bez) oraz parkingów[18].

Poniżej przedstawiony jest model Biham-Middleton-Levine'a[3], który ma znaczenie teoretyczne, a w kolejnych podrozdziałach składowe modeli, które doczekały się praktycznych aplikacji w dużych aglomeracjach miejskich.

Model Biham-Middleton-Levine. Prosty model opisujący ruch drogowy w dwóch wymiarach. Model opisuje kwadratową powierzchnię. Kwadrat jest macierzą komórek, z których każda może zawierać pojazd zmierzający na północ lub na wschód. Model ruchu jest znacznie uproszczony w porównaniu do modelu *NaSch*[19]. Pojazd porusza się do przodu o jedną komórkę, jeżeli jest ona wolna, w innym przypadku zostaje na aktualnej pozycji. Wyróżnia się 3 warianty tego modelu:

1. Pierwszy wariant jest taki, że dynamiką całości sterują światła na skrzyżowaniach. W parzystych turach poruszają się pojazdy zmierzające na północ, w nieparzystych te zmierzające na wschód. W każdej turze pojazd porusza się o jedno pole „do przodu", aż natrafi na koniec przestrzeni lub na inny pojazd. Ten wariant jest całkowicie deterministyczny, jedyne co można dzięki niemu pokazać to problem równoczesnego przepływu ruchu w dwóch kierunkach na jednej płaszczyźnie.

[18] Parking — w przypadku systemów symulacji ruchu ulicznego to jedyny element systemu w którym mogą się pojawiać i znikać pojazdy. Parkingowi w modelu nie zawsze odpowiada parking w modelowanym środowisku (mieście).

[19] krótki opis modelu BML i propozycję bardziej szczegółowego modelu — opisano w artykule *ChSch*[22]; pozycja [4] prezentuje pewne metody optymalizacji pracy sygnalizacji świetlnej

2. W wariancie niedeterministycznym usuwa się światła ze skrzyżowań i wszystkie pojazdy mogą się poruszać cały czas (wtedy, gdy mają miejsce). Jeżeli dwa pojazdy próbują dostać się na to samo miejsce w jednej turze, jeden z nich jest wybierany losowo.
3. Trzecia wersja wprowadza możliwość jednoczesnego zajmowania komórki przez pojazdy północny oraz wschodni. Jeżeli dwa pojazdy próbują dostać się na to samo miejsce w jednej turze, obydwa zajmują to miejsce. Jednak żaden pojazd nie może dostać się na miejsce już zajmowane przez inny, musi się zatrzymać.

2.1.5. Ruch autostradowy i miejski

Tworząc model ruchu należy skupić uwagę przede wszystkich na tych zjawiskach, które wywierają największy wpływ na jego kształt. Aby model pasował do rzeczywistości, muszą one zostać wiernie odzwierciedlone. Zjawiska mniej ważne można w modelu uprościć albo wręcz pominąć.

Używając pojęcia ruch drogowy mamy na myśli ruch samochodowy odbywający się na terenach niezurbanizowanych.

W ogólności charakter ruchu samochodowego jest zdominowany przez najwęższe gardła systemu komunikacyjnego. W obszarze miejskim najwęższym gardłem jest sygnalizacja świetlna na skrzyżowaniach: włączenie światła czerwonego powoduje natychmiastowe uformowanie się korka, natomiast włączenie światła zielonego rozpoczyna ruch samochodów.

Ponieważ sposób przemieszczania się różnych pojazdów od jednego skrzyżowania do następnego jest podobny (zbliżone przyspieszenie podczas ruszania, prędkość podczas jazdy), to można je traktować bardziej ogólnie. Niektóre modele (tzw. modele kolejkowe) nie symulują nawet samego przejazdu samochodu pomiędzy skrzyżowaniami, a jedynie szacują jego czas. Samochód po opuszczeniu skrzyżowania „zasypia” na pewien czas. Po „obudzeniu” jest dodawany do kolejki samochodów oczekujących na przejazd przez kolejne skrzyżowanie.

Poza miastem, na długim odcinku drogi (np. autostradzie) wąskim gardłem jest jej pojemność. Jeśli gęstość ruchu, tj. stosunek samochodów znajdujących się na pewnym odcinku do jego powierzchni, przekroczy pewien punkt krytyczny (charakterystyczny dla odcinka), to najprawdopodobniej zacznie się w tym rejonie tworzyć korek. Można to zaobserwować w miejscach zmniejszania się ilości pasów na drodze. Aby dobrze oddać te zjawiska należy precyzyjnie modelować sposób jazdy samochodów i interakcje pomiędzy nimi.

2.2. Sposoby monitorowania stanu ruchu drogowego

Systemy monitorowania ruchu drogowego są kluczowym elementem w inteligentnych systemach transportowych. Podstawowe zadanie monitoringu to zapewnienie aktualnych danych różnym elementom systemu sterowania. Od jakości danych pomiarowych zależy to, czy uda się w pełni wykorzystać możliwości systemu sterowania ruchem oraz możliwości infrastruktury drogowej.

W zależności od celu systemu nadzoru, monitorowaniu mogą podlegać różne parametry np.: stan zatłoczenia, prędkość, czasy przejazdu, zanieczyszczenie powietrza. W tej pracy opisane są elementy systemu monitoringu, których zadaniem jest dostarczenie informacji o stanie ruchu drogowego do systemu zarządzania ruchem (zarządzanie sygnalizacją świetlną oraz sterowanie trasą pojazdów poprzez znaki o zmiennej treści i systemy radiowe)[20].

2.2.1. Monitorowane parametry

Monitorowane parametry ruchu obejmują:

- natężenie ruchu, mierzone jako wielkość przepływu strumienia,
- gęstość (stopień nasycenia sieci, stan zatłoczenia danej drogi),
- prędkość,
- długość kolejki pojazdów,
- czasy podróży,
- czasy czekania w kolejkach.

Parametry powinny być zapisywane i dostępne jako dane historyczne dla późniejszej analizy. Wartości parametrów mogą być agregowane do przedziałów obserwacji 15-, 30-, 60-minutowych w zależności od dynamiki zmian na badanym obszarze.

Lokalnie należy monitorować parametry takie jak natężenie ruchu, długość kolejki pojazdów lub czas czekania w kolejce. Te parametry mogą być bezpośrednio używane do sterowania sygnalizacją świetlną. Dla prostego sterownika sygnalizacji świetlnej wystarczy tylko informacja o obecności pojazdu (wartość logiczna, prawda/fałsz).

Średnia prędkość, gęstość ruchu i średni czas przejazdu to wartości, które powinny być monitorowane na poziomie pojedynczych dróg (czyli również lokalnie), ale powinny znajdować się w bazie danych dostępnej dla zintegrowanego systemu sterowania ruchem (oraz dla kierowców).

[20] Sposoby monitorowania ruchu opisano na podstawie książki [1], oraz informacji zawartych na stronach systemu http://traffic.houstontranstar.org

Dodatkowo, na poziomie całego miasta, powinny być monitorowane wartości: gęstości, średniej prędkości, czas podróży od źródła do celu (tę da się zmierzyć tylko przez przejechanie danej trasy) oraz ilość pojazdów, które zdołały zrealizować plany podróży.

2.2.2. Źródła danych

Dane z systemu monitoringu można uzyskać na podstawie wartości pochodzących z pomiarów oraz z estymacji parametrów na podstawie danych pomiarowych. Dane pomiarowe uzyskuje się z różnych źródeł:

- detektory i czujniki ruchu drogowego:
  - pętle indukcyjne,
  - wideodetektory,
  - lasery,
  - tuby powietrzne,
- urządzenia nadawczo-odbiorcze montowane w pojazdach[21] (np.: tagi RFID[22]),
- urządzenia telekomunikacyjne pojazdów komunikacji miejskiej,
- systemy identyfikacji i lokalizacji pojazdów (*AVL*[23], *AVI*[24], *GPS*[25]),
- pojazdy testowe[26] i mobilne stacje monitoringu.

Podstawowym źródłem informacji są detektory. Wszystkie detektory umożliwiają zliczanie pojazdów. Detektory laserowe umożliwiają zliczanie pojazdów oraz pomiar prędkości. Tuby powietrze (stare rozwiązanie, działające na zasadzie badania pędu powietrza wywołanego przez przejeżdżający pojazd) mogą zliczać pojazdy oraz podają ich prędkość. Z racji swojej budowy tuby powietrzne są niedokładne. Pętle indukcyjne mają prostą zasadę działania (pętla przewodnika zatopiona w asfalcie, indukowana magnetycznie przez przejeżdżające pojazdy) i dobrze działają jako licznik pojazdów, są one jednak bardzo podatne na uszkodzenia w miarę zużywania się nawierzchni, powstawania kolein oraz wymagają wymiany w przypadku remontów nawierzchni. Pomiar prędkości pojazdów wymaga instalacji dwóch detektorów pętlowych. Wartości takie jak gęstość, natężenie ruchu, długość kolejki i czasy podróży muszą być obliczane

[21] w USA oraz w Niemczech systemy sterowania ruchem wykorzystują informacje z systemu pobierania opłat za autostrady

[22] Radio-frequency identification — radiowy system identyfikacji

[23] Automated Vehicle Location — automatyczna lokalizacja pojazdów

[24] Automated Vehicle Identification — automatyczna identyfikacja pojazdów

[25] globalny system lokalizacji

[26] w Houston można spotkać specjalne służby patrolujące okresowo badany obszar

na podstawie liczby pojazdów oraz ich prędkości na przynajmniej dwóch punktach pomiarowych.

Najczęściej stosowane detektory to pętle indukcyjne oraz, w nowych systemach, wideo-detektory ruchu. Wideodetektory to najczęściej kamery przemysłowe z oprogramowaniem, które umożliwia zliczanie pojazdów oraz ocenę ich prędkości. Umieszczenie kamery pod odpowiednim kątem umożliwia również ocenę długości kolejki przed światłami.

2.2.3. Integracja informacji zwrotnej

Symulatory, które służą do przewidywania stanu dróg w najbliższej przyszłości, wymagają nieustannej korekty stanu modelu do stanu rzeczywistego.

W ramach projektu badawczego w sieci autostrad Nadrenii Północnej Westfalii zainstalowanych zostało kilka tysięcy pętli indukcyjnych dokonujących takich pomiarów jak liczba i średnia prędkość przejeżdżających pojazdów. Tym miejscom na drodze, w których wszystkie pasy są monitorowane przez pętle indukcyjne odpowiadają w modelu tzw. punkty kontrolne. Ogólna idea integracji informacji zwrotnej („dostrajania stanu modelu") polega na mierzeniu różnicy pomiędzy ilością pojazdów przejeżdżających w modelu przez punkt kontrolny, a ilością pojazdów przejeżdżających przez pętle indukcyjne mu odpowiadające i uwzględnieniu tej różnicy w stanie modelu.

2.3. Sygnalizacja świetlna

Sygnalizacja świetlna jest metodą kierowania ruchem stosowaną w celu oddzielenia od siebie w czasie kolizyjnych strumieni ruchu. Takie oddzielenie różnych grup ruchu jest możliwe tylko dzięki temu, że strumienie kolizyjne są zatrzymywane. Powoduje to straty czasowe. Jednak w gęstej zabudowie miejskiej, gdzie oddzielenie różnych kierunków przestrzennie jest niemożliwe, sygnalizacja świetlna jest najlepszym rozwiązaniem.

Powszechnie stosuje się 3 sygnały: czerwony — stop, zielony — jedź oraz żółty (lub pomarańczowy) oznaczający, że niedługo nastąpi zmiana. Główną funkcją sygnału żółtego jest czasowe rozdzielenie kolejnych faz, co zmniejsza niebezpieczeństwo wystąpienia kolizji.

Sygnalizację świetlną można podzielić według planu fazowego[27] na cykliczną, acykliczną i akomodacyjną. Sygnalizacja cykliczna może mieć plan z ustalonymi lub zmiennymi długościami faz.

[27] podziału można dokonać wg. różnych kryteriów, ten sposób podziału zaproponowano na podstawie referatu: http://www.drogowiec.pb.bialystok.pl/referaty/sygnalizacja.htm, 2007

Sygnalizacja ze stałymi czasami ma stałą sekwencję kolejnych faz, ze sztywno ustaloną długością cyklu i poszczególnych sygnałów. Jedno skrzyżowanie może mieć kilka planów przygotowanych na różne pory dnia lub na specjalne wydarzenia (ustawiane ręcznie).

Sygnalizacja akomodacyjna wykorzystuje plan, w którym czas trwania każdej fazy jest zmienny (fazy można również pomijać) i zależny od aktualnych warunków ruchu. Stan ruchu określany jest za pomocą danych zbieranych z detektorów. Dla bezpieczeństwa czasy trwania faz mają ustalone wartości graniczne min i max, umożliwia to pracę nawet w przypadku awarii detektorów.

Sygnalizacja acykliczna nie ma ustalonego stałego cyklu, a fazy i czasy ich trwania są ustalane na bieżąco, na podstawie warunków ruchu. Program w tym przypadku jest całkowicie dynamiczny, więc dobrze dostosowuje się do aktualnych warunków na drodze. Jednak ta dynamika może utrudniać koordynację sąsiednich skrzyżowań.

Sygnalizacja wzbudzana ma ustalony stan wyjściowy, w którym pracuje według ustalonego programu, a po wzbudzeniu przełącza się na pewien czas w stan realizujący funkcję nie objęte w cyklu standardowym, po czym powraca do stanu wyjściowego.

Sterowniki sygnalizacji na skrzyżowaniach mogą być odosobnione lub skoordynowane z sąsiednimi (omówione w punkcie 2.4).

2.4. Koordynacja sterowania sygnalizacją świetlną

Nawet jeżeli sygnalizacja świetlna jest skonfigurowana optymalnie w otoczeniu lokalnym, to cały system może nie być w stanie optymalnym. Dzieje się tak, ponieważ sygnalizacja świetlna[28] na jednym skrzyżowaniu zmienia warunki środowiska sygnalizatorów na wszystkich sąsiadujących skrzyżowaniach. Wpływ działania sygnalizacji na jednym skrzyżowaniu na sąsiednie skrzyżowania może być pozytywny i negatywny. Aby poprawić przepustowość infrastruktury drogowej należy koordynować pracę skrzyżowań.

W przypadku dobrze zaprojektowanej sygnalizacji adaptacyjnej, gdzie program fazowy zmienia się w zależności od wielu kryteriów, można zaobserwować zjawisko samo-organizacji się sterowania sygnalizacją świetlną. Jednak przy ustalonych długościach cyklu należy w tym celu zaprojektować sposób koordynacji skrzyżowań.

Koordynacja może być statyczna — ustawiona na stałe (skrzyżowania są zsynchronizowane wg. zegarów) lub dynamiczna. W tym drugim przypadku stosuje się

[28] w tej pracy używane zamiennie sformułowania: *sygnalizacja świetlna*, *sygnalizator* oraz *skrzyżowanie* — oznaczają sygnalizację świetlną do sterowania ruchem na skrzyżowaniu

rozwiązania sterowania centralnego oraz rozproszonego. Metody statyczne są ustalane z reguły dla określonych wartości natężenia ruchu (które ciągle się zmieniają). Duża zmienność warunków ruchu drogowego sprawia, że nie ma jednorazowych rozwiązań optymalnych. System sterowania musi się adaptować do aktualnej sytuacji. Wybrane rozwiązania są opisane w dalszej części tego rozdziału. Podział sterowania ze względu na sposób koordynacji:

- autonomiczne,
- koordynowane statycznie,
  - synchronizacja zegarowa (np.: zielona fala w jednym kierunku),
  - priorytety ruchu dla głównej arterii,
- koordynowane dynamicznie,
  - centralne (metody programowania dynamicznego, globalny problem optymalizacji),
  - rozproszone (systemy agentowe, teoria gier).

2.4.1. Synchronizacja zegarowa

Jedną z metod koordynacji pracy sygnalizacji świetlnej na skrzyżowaniach jest synchronizacja zegarów. Jeżeli fazy programu sygnalizacji mają stałe długości, a odległości pomiędzy kolejnymi skrzyżowaniami są niewielkie[29], to wprowadzając przesunięcie fazy zielonego światła na kolejnych skrzyżowaniach uzyskuje się *zieloną falę*. Gdy tylko na skrzyżowaniu czerwone światło zmienia się na zielone, pojazd zatrzymany na tym skrzyżowaniu będzie miał zielone na następnym skrzyżowaniu. Przesunięcie w fazie musi być równe średniemu czasowi przejazdu przez odcinek drogi dzielący dwa kolejne skrzyżowania.

W zależności od efektu, który chce się uzyskać, można zsynchronizować zieloną falę o dużej szybkości w jedną stronę (wtedy ruch w kierunku przeciwnym nie jest synchronizowany) lub po odpowiednim dostosowaniu limitów prędkości oraz skróceniu (wydłużeniu) fazy sygnalizacji można uzyskać dwukierunkową zieloną falę. Wbrew pozorom synchronizacja tylko w jednym kierunku może czasami dać o wiele lepsze rezultaty niż ogólna optymalizacja przepustowości Odpowiednie ograniczenie ruchu do centrum miasta i przyspieszenie ruchu wyjeżdżającego może znacząco poprawić

[29] odległość jest niewielka, jeżeli jej przejechanie zajmuje mniej czasu niż długość programu sygnalizacji; jeżeli odległość jest duża to trudno uzyskać w ten sposób skuteczną synchronizację, ponieważ niewielkie zmiany średniej prędkości mogą powodować duże różnice w czasie dojazdu do następnej sygnalizacji

sytuację w centrum, a na kierowcach wjeżdżających wymusza poszukanie innej drogi np.: obwodnicy, gdy celem podróży jest drugi koniec miasta.

W przypadku dróg skrzyżowanych na wzorze kratownicy, podejmowane są próby synchronizacji kierunków prostopadłych (por. model BML opisany na stronie 21 oraz w pozycji[3]).

Synchronizacja zegarowa jest zestrojona optymalnie tylko dla określonych sytuacji. Nie bierze pod uwagę aktualnego stanu ruchu. Dlatego zazwyczaj potrzeba przynajmniej czterech planów synchronizacji dla jednego systemu. Plany porannego szczytu, południowy, popołudniowego szczytu oraz wieczorny/nocny.

Systemem do opracowania planów synchronizacji zegarowej jest TRANSYT[30] rozwijany początkowo przez wydział badań Rządu Brytyjskiego: *„Transport and Road Research Laboratory (TRRL)"*.

2.4.2. TRANSYT i SCOOT

Pierwszym podejściem do poprawy stanu systemu sieci dróg jest optymalizacja przepustowości. Inżynier ruchu zazwyczaj stara się zoptymalizować przepustowość stosując do tego celu diagramy czas-odległość. W przypadku skomplikowanych sieci w centrum miast w godzinach szczytu większa przepustowość w kierunku x wcale nie oznacza lepszej sytuacji dla tego kierunku. Duża gęstość ruchu, długie kolejki przed światłami tworzą skomplikowany system, w którym sama przepustowość traci na znaczeniu (pojazdy skręcające mogą zablokować ruch na wprost, tym samym uzyskana przepustowość nie jest wykorzystywana). Innymi celami optymalizacji są minimalizacja długości kolejek (ten cel jest brany pod uwagę przez systemy TRANSYT oraz SCOOT) oraz minimalizacja ilości zatrzymań podczas przejazdu przez miasto.

Systemy TRANSYT oraz SCOOT[31]używają do obliczeń cyklicznych profili przepływu (CFP — cyclic flow profile). CFP jest miarą ilości pojazdów przejeżdżających przez pewien odcinek drogi w trakcie trwania całego cyklu sygnalizacji świetlnej. TRANSYT używa wartości średnich dla okresu 1h (dane historyczne), natomiast SCOOT wykorzystuje dane odświeżane co 4 sekundy (dane aktualne, analizowane na bieżąco). CFP to profil, który pokazuje charakterystykę ruchu dla danego cyklu. Na podstawie tych profili, można oszacować długości kolejek na następnym skrzyżowaniu (za punktem w którym profil został zmierzony). Długość kolejek oraz szybkość ich

[30] TRANSYT — więcej informacji o systemie TRANSYT można znaleźć na stronie producenta *http://mctrans.ce.ufl.edu/featured/TRANSYT-7F*

[31] SCOOT — strona producenta *http://www.scoot-utc.com*

wzrostu i rozładowania są wszystkimi informacjami potrzebnymi do procesu optymalizacji.

Według autorów pracy[21] przygotowanie dobrych profili przepływu dla średniej wielkości sieci miejskiej (tylko główne skrzyżowania, od 30 do 40 węzłów) oraz opracowanie optymalnych sygnałów zajmuje około jednego roku. Dlatego opracowano metodę SCOOT, która w odróżnieniu od metody TRANSYT, działa na podstawie bieżących, ciągle aktualizowanych danych oraz zapewnia stopniową optymalizację systemu (jest to więc system adaptacyjny). Warto zauważyć, że system SCOOT jest ewolucją systemu TRANSYT.

Przed każdą zmianą fazy SCOOT oblicza, na podstawie aktualnych danych, czy lepiej jest wydłużyć, czy skrócić aktualną fazę o maksymalnie 4 s. Następnie obliczana jest funkcja kosztu na drogach sąsiednich dla tego skrzyżowania pod warunkiem zmiany wartości przesunięcia cyklu o 4 sekundy wcześniej lub później. Podobnie czasy cykli dla grup skrzyżowań mogą być zmieniane o kilka sekund co kilka minut. Zazwyczaj SCOOT przeprowadza 10000 sesji optymalizacji na godzinę dla sieci złożonej z 1000 węzłów.

Dzięki temu, że optymalizator używa wartości ważonych podczas przetwarzania danych, można uzyskać optymalizację ze ścieżkami priorytetowymi. Ważne jest również to, że w przypadku awarii części detektorów, plany sygnalizacji w rejonie awarii wracają stopniowo do ustalonego programu (poprawki nie są obliczane przez system w kolejnych sesjach). Sprawia to, że w sytuacji krytycznej system nie wywoła chaosu w ruchu drogowym.

Efektywność tego systemu została potwierdzona przez badania przeprowadzone w Glasgow, Coventry, Worcester, Southampton i w Londynie[21]. Aktualna wersja systemu pozwala na sterowanie priorytetowe ruchu komunikacji miejskiej.

2.4.3. SOTL

Self-Orginizing Traffic Lights[8] — samo-organizująca się sygnalizacja świetlna. Zasada działania tej metody polega na adaptacyjnym procesie samo-organizacji pracy sygnalizatorów. Termin samo-organizacja jest używany w kontekście różnych dziedzin nauki z różnym znaczeniem. W tym przypadku system, który jest samo-organizujący się to taki, w którym elementy współdziałają w celu osiągnięcia pewnego globalnego stanu. Ten stan jest osiągany dynamicznie w procesie współdziałania pomiędzy elementami systemu. Na uwagę zasługuje fakt, że nie używa się tutaj żadnej metody komunikacji pomiędzy elementami systemu.

Sotl-request. Zasada pracy każdego skrzyżowania jest bardzo prosta. Kontroler sygnalizacji utrzymuje informacje o liczbie pojazdów c na pasie, na którym świeci się aktualnie czerwone światło, oraz o czasie t, jaki upłynął od zapalenia czerwonego światła. Gdy wartość $c * t$ przekroczy górny próg θ, następuje *wymuszenie* zmiany świateł na zielone na danym pasie (co pociąga za sobą zmianę fazy pracy świateł dla danego skrzyżowania). Jeżeli więcej pojazdów czeka na pasie, to światło szybciej zmieni się na zielone. Na pasie, na którym jest mniej pojazdów, światło dłużej pozostanie czerwone. Z tych prostych reguł wynika proporcjonalny przydział czasu dla kierunków o różnym natężeniu ruchu. Drugą zaletą wynikającą ze stosowania tej metody jest łączenie pojazdów w grupy. Ruch konwojów w porównaniu do ruchu pojedynczych pojazdów znacząco poprawia przepływ. Dzieje się tak dzięki temu, że pomiędzy konwojami tworzą się duże puste przestrzenie, które mogą być wykorzystywane przez ruch poprzeczny.

Ponieważ ta metoda nie ma określonego programu sygnalizacji, zmiany są wymuszane tylko przez przejeżdżające pojazdy. W zależności od wartości θ, przy dużym ruchu światła mogą zmieniać się zbyt często.

Sotl-phase. Aby zapobiec zbyt częstym zmianom wprowadza się: zegar (bieżąca wartość zegara to ϕ_i), program fazowy z czasami minimalnymi ϕ_{min} oraz dodatkowy warunek zmiany koloru świateł: $\phi_i \geqslant \phi_{min}$, tzn. nie można zmienić świateł, jeśli upłynęło mniej czasu niż ϕ_{min}.

Sotl-platoon. Trzecia metoda wprowadza dwa dodatkowe ograniczenia w porównaniu do *sotl-phase*. Czerwone światło nie jest zmieniane na zielone, jeżeli na prostopadłej ulicy przynajmniej jeden pojazd zbliża się do skrzyżowania i jest w odległości mniejszej niż ω od skrzyżowania. Dzięki temu konwoje nie są dzielone. W warunkach ruchu o dużej gęstości to ograniczenie zaburzyłoby płynność ruchu, gdyż duże konwoje blokowałyby ruch z prostopadłych kierunków. Drugie ograniczenie zostało wprowadzone, aby temu zapobiec. Gdy więcej niż μ pojazdów zbliża się do skrzyżowania, nie bierze się pod uwagę pierwszego ograniczenia.

System testowano dla bardzo prostego, prostopadłego modelu, z ruchem jednostronnym. Można się spodziewać, że dla bardziej skomplikowanych układów, system będzie wymagał dostrojenia i być może dodania dodatkowych parametrów (co w zasadzie zmienia tę metodę w inną). Prostota systemu jest czynnikiem, który pozwala na jego szybkie zaimplementowanie. Fakt, że wystarczy pomiar ilości samochodów, oraz że system nie wymaga komunikacji pomiędzy sygnalizatorami sprawia, że system można z powodzeniem zainstalować na rzeczywistym skrzyżowaniu.

Zachowanie systemu wyznaczone jest przez reguły bazujące na aktualnych danych, a to sprawia, że system ciągle adaptuje się do bieżącej sytuacji. W przypadku dobrego zestrojenia parametrów, system uzyskuje optymalne (lub prawie optymalne) rezultaty dla różnych gęstości ruchu. Jest duża zaleta metod adaptacyjnych w porównaniu do metod optymalizacji *off-line*, gdzie parametry wyliczone są wcześniej dla ogólnego przypadku. Przykładem może być optymalna *zielona fala*, której skuteczność jest zależna od średniej prędkości pojazdów. Średnia prędkość zmienia się w zależności od gęstości ruchu, więc zielona fala ustawiona dla dużej gęstości będzie za wolna dla ruchu luźnego i odwrotnie.

2.4.4. RL — Reinforcement Learning

Algorytm[23], opisany w poniższym podpunkcie, realizuje strategię uczenia ze wzmocnieniem.

Problemem jest wybór optymalnej konfiguracji świateł w danej chwili. Konfigurację świateł skrzyżowania j oznacza się przez A_j. Konfigurację można wybierać z predefiniowanego zbioru lub tworzyć dynamicznie na podstawie grup ruchu, które nie mają toru kolizyjnego.

Aby określić cel agentów-sterowników świateł konieczne jest wprowadzenie funkcji kosztu. Jeśli samochód musi w danej turze czekać, to można przyjąć 1 jako wartość kosztu: $R(s,s) = 1$ [32], a wartość 0 jeśli może jechać: $R(s,s') \neq 0$, $s \neq s'$. Wartość tej funkcji jest zależna od zmian stanu pojazdu. Podczas symulacji, w każdej chwili samochód znajduje się w pewnym stanie $s = [p,d]$, gdzie p jest pozycją (komórką), w której samochód się znajduje, a d celem podróży. W prostych słowach koszt R to kara nakładana na samochód za to, że w turze nie wykonał ruchu.

W celu optymalizacji używa się dwóch funkcji, które szacują ilość kary nałożonej na samochód do końca jego podróży. Pierwsza z nich $Q(s,l)$ szacuje ilość kary nałożonej na samochód będący w stanie s, gdy znane jest światło l na skrzyżowaniu, do którego zmierza. Druga $V(s)$ szacuje tą samą karę [33], ale już bez znajomości koloru światła.

Definiuje się $Q(s,l)$ dla samochodu będącego w stanie s zbliżającego się do światła (lub stojącego przed światłem) o kolorze l $(red, green)$ jako:

$$Q(s,l) = \sum_{s'} P(l,s,s')(R(s,s') + \gamma V(s'))$$

[32] $R(pozycja_A, pozycja_B)$ koszt generowany poprzez pojazd przemieszczający się z $pozycja_A$ do $pozycja_B$

[33] w szczególności jest to średni czas oczekiwania pojazdu (zmierzającego do celu d) na pozycji p przed światłami

gdzie $P(l, s, s')$ jest prawdopodobieństwem tego, że zapalone jest światło l oraz samochód przejdzie ze stanu s do s'. Należy zauważyć, że s' może także być równe s, ponieważ nawet przy zielonym świetle samochód może zostać w miejscu. Używa się $\gamma < 1$ jako czynnika powodującego preferowanie kosztów krótkoterminowych (efektów podjęcia decyzji) nad długoterminowymi. Drugą z funkcji $V(s)$ definiuje się jako:

$$V(s) = P(green|s)Q(s, green) + P(red|s)Q(s, red)$$

gdzie $P(l|s)$ z $l \in \{red, green\}$ to prawdopodobieństwo tego, że światło (na najbliższym skrzyżowaniu) jest czerwone/zielone, kiedy samochód jest w stanie s. Z definicji $V(s) = 0$, jeśli s jest docelowym stanem samochodu.

Dla wszystkich samochodów czas dojazdu do celu będzie większy w przypadku, gdy na najbliższym skrzyżowaniu pali się światło czerwone, niż w przypadku, gdy pali się zielone. Dlatego też każdy samochód preferuje zapalenie światła zielonego, a jako intensywność tej preferencji można użyć różnicy czasu dojazdu przy czerwonym i zielonym świetle. Niech j będzie skrzyżowaniem. Światła składają się z pojedynczych sygnalizatorów: $i \in L_j$. Należy więc dążyć do wybrania takiej konfiguracji świateł A_j na skrzyżowaniu j, aby zmaksymalizować sumę:

$$\sum_{i \in A_j} \sum_{s \in queue_i} Q(s, red) - Q(s, green)$$

gdzie $queue_i$ to kolejka samochodów stojących przed sygnalizatorem i. Pisząc kolejka mamy na myśli te samochody, które *stoją* jeden za drugim przed sygnalizatorem świetlnym.

Na początku jako wartości Q i V można dla wszystkich stanów przyjąć 0. W trakcie działania symulacji prowadzone jest zliczanie zdarzeń $green|s$, $red|s$ dla każdego stanu (komórki) s, przejść $green, s, s'$ oraz red, s, s' dla każdej pary stanów s i s'. Po każdej turze bada się jakie zdarzenia w niej zaszły i zwiększa liczniki z nimi związane. Na podstawie wartości tych liczników można wyznaczyć prawdopodobieństwa: $P(l, s, s'), P(l|s)$.

Autorzy zbudowali prosty symulator *Green Light District*[34], w celu przetestowania zaproponowanej metody. Wyniki testów porównawczych prezentują prace [24, 23].

2.4.5. Algorytm OptAPO

Algorytm OptAPO — Optimal Asynchronous Partial Overlay[16, 7], podobnie jak *SOTL* opisany w rozdziale 2.4.3, wymaga informacji o ilości pojazdów. Dodatkowo

[34] GLD – http://www.sf.net/projects/stoplicht, 2006

potrzebna jest pewna wiedza o topologii skrzyżowań. Idea tej metody (rozproszony protokół mediacji) polega na utworzeniu grup współpracujących agentów-skrzyżowań. Wtedy w topologii miejskiej tworzą się kanały lub arterie, dla których ustalony jest jeden główny kierunek ruchu.

Każdy agent wykonuje podstawowe kroki algorytmu, w celu poprawienia lokalnego rozwiązania, przy czym agent może zmienić lokalne rozwiązanie tylko wtedy, gdy jego zmiana poprawia wartość globalnego rozwiązania.

W prezentowanym rozwiązaniu jeden agent ma przypisaną jedną zmienną. Agent przechodzi przez 3 fazy: inicjalizacja, sprawdzenie widoku, mediacja. W fazie inicjalizacji następuje ustawienie początkowych wartości: aktualna wartość (d_i) oraz nazwa (x_i) zmiennej przypisanej do agenta, priorytet (p_i), dziedzina (D_i), relacje (C_i), listy znanych agentów *good_list* oraz *agent_view*.

Podczas drugiej fazy agent i oblicza koszt F_i w podgrafie relacji *good_list* używając widoku *agent_view*. Jeżeli aktualny koszt jest większy od kosztu rozwiązania optymalnego ($F_i > F_i^*$), wtedy agent i przeprowadza aktywną lub pasywną mediację (po czym F_i jest ponownie przeliczona).

Jeżeli priorytet agenta i jest niższy niż priorytet innego agenta w podsystemie to przejdzie on w stan pasywny podczas mediacji, w przeciwnym przypadku ustawi tymczasową flagę m_i w stan aktywny. Gdy flaga mediacji jest w stanie aktywnym agent może być mediatorem, jeżeli nie ma innego agenta z aktywną flagą mediacji i wyższym priorytetem. Jeżeli zmiana lokalnej wartości spowoduje, że lokalny koszt osiągnie wartość optymalną, to agent zmienia tę wartość, ale nie zaczyna sesji mediacji. Jeżeli flaga mediacji jest ustawiona w stan mediacji pasywnej, wtedy agent zaczyna mediację pasywną.

W fazie mediacji agent, który otrzymuje żądanie mediacji, oblicza wartości lub wysyła wiadomość „czekaj". Obliczenie polega na przejrzeniu wszystkich elementów z dziedziny i etykietowanie ich nazwami agentów, dla których koszt relacji f_i jest większy niż f_i^*. Po obliczeniu tych wartości są one rozsyłane. Mediator przeprowadza poszukiwanie metodą ograniczeń i rozgałęzień (ang. branch-and-bound).

Wykorzystanie OptAPO do problemu koordynacji sygnalizacji świetlnej. W tym przypadku każdy agent ma przypisaną dokładnie jedna zmienną i jest nią kierunek synchronizacji. Dziedziną tej zmiennej jest dwuwartościowy zbiór $D = \{NS/SN, EW/WE\}$, a zmienna może przyjąć tylko jedną z tych wartości. Koordynacja w jednym kierunku sprawia, że agent-sterownik ustala swoje fazy świateł zgodnie z sąsiadami ulokowanymi na wybranym kierunku. Jeżeli czasy planu sygnalizacji usta-

lone są tak, aby dać priorytet kierunkowi NS/SN, to nie możliwa jest równoczesna koordynacja programu w kierunku WE/EW.

Funkcja kosztu liczona jest na podstawie liczby pojazdów dojeżdżających do skrzyżowania oraz kierunku synchronizacji. Koszt dla danego agenta zależy od kierunku w którym ruch ma największe natężenie. Agent liczy koszt jako sumę kosztów z agentami sąsiadującymi w tym konkretnym kierunku (z większym natężeniem ruchu). Funkcja kosztu pomiędzy dwoma agentami obliczana jest w następujący sposób:

- kierunki synchronizacji są takie same oraz jest to synchronizacja w kierunku większego natężenia ruchu — koszt wynosi 0 $f(x_i, x_j) = 0$.
- agent i używa planu sygnalizacji z priorytetem w kierunku, w którym ruch ma większe natężenie, ale plan agenta j nie jest zsynchronizowany w tym kierunku — koszt jest obliczany jako stosunek liczby pojazdów jadących z x_j do x_i do liczby wszystkich pojazdów dojeżdżających do skrzyżowania x_i.
- agent i używa planu sygnalizacji z priorytetem w kierunku innym niż kierunek ruchu o największym natężeniu — koszt jest podwojoną wartością kosztu obliczonego jak w poprzednim przypadku.

Celem optymalizacji problemu jest koordynacja ruchu w celu zminimalizowania globalnego kosztu.

2.5. Inne algorytmy i metody optymalizacji ruchu

W poniższej sekcji opisane są dwie kolejne metody optymalizacji ruchu w mieście. Jedną z nich jest informowanie kierowców o lokalnej sytuacji drogowej za pomocą znaków o zmiennym tekście (VMS — Variable Message Sign). Druga metoda polega na przydzielaniu priorytetów dla komunikacji masowej poprzez wydzielanie pasów ruchu oraz specjalne sterowanie sygnalizacją świetlną.

2.5.1. Znaki o zmiennym tekście — VMS

Jedną z metod poprawy pojemności sieci drogowej jest kontrola ruchu przy pomocy znaków o zmiennym tekście. Takie znaki mogą wyświetlać informację o aktualnym ograniczeniu prędkości (zmienne ograniczenie prędkości na autostradzie może zwiększyć jej ogólną przepustowość i zapobiec tworzeniu się korków), o optymalnej prędkości, która zapobiegnie utworzeniu się korka lub pozwoli kierowcy trafić na zielone światło na następnym skrzyżowaniu. Znaki o zmiennym tekście mogą być zastosowane do

sugerowania najkrótszych[35] tras w sieci drogowej oraz do informowania kierowców o wypadkach i zdarzeniach losowych w najbliższej okolicy.

Podstawowym warunkiem, który decyduje o skuteczności tej metody sterowania ruchem, jest jej dokładność, przewidywalność oraz akceptacja u kierowców. Jeżeli chodzi o akceptację u kierowców, to nie ma na to wpływu żaden element systemu, a jedynie sposób jego użycia. Należy zadbać o to, aby znaki *VMS* nigdy nie były używane do celów innych niż informacje o ruchu (w przeciwnym przypadku kierowcy będą je ignorowali). Wyświetlanie danych nieaktualnych lub nieprawidłowych może zakłócić sytuację drogową. Jeżeli znaki będą kierowały kierowców na „lepszą" trasę, która od 15 minut jest zatłoczona, to spowoduje to całkowite zablokowanie tej okolicy (jako skutek uboczny zmniejszy akceptację kierowców dla tej metody).

Przykład 2.1. Na drodze X wzrasta ilość pojazdów i zatłoczenie. Przed skrzyżowaniem, z którego można wjechać na tę drogę, stoi znak o zmiennym tekście (oczywiście musi stać w odpowiedniej odległości, aby umożliwić kierowcom zmianę trasy przed skrzyżowaniem). Na takim znaku wyświetlone zostaną aktualne informacje o czasie przejazdu (przez drogi odchodzące z najbliższego skrzyżowania) oraz sugestie objazdu. Jeżeli część kierowców zastosuje się do tej informacji, to taki system może zapobiec zakorkowaniu się najbliższej okolicy drogi X.

Przykład 2.2. Znak VMS pokazujący ograniczenie prędkości na autostradzie[5]. Jeżeli gęstość strumienia ruchu jest niewielka (na drodze panuje mały ruch), to znak pokazuje maksymalną prędkość dozwoloną prawnie na tym odcinku. Zapewnia to największą przepustowość w tych warunkach. Gdy gęstość strumienia ruchu rośnie, to system, bazując na aktualnych danych o stanie ruchu, zmienia limit prędkości na niższy (na podstawie eksperymentu symulacyjnego lub danych obliczonym za pomocą algorytmu optymalizacji). Zmniejszenie limitu prędkości sprawia, że można bezpiecznie uzyskać o wiele większą gęstość (ponieważ odstępy pomiędzy pojazdami mogą być mniejsze oraz oddziaływania zmian prędkości pojazdu na zachowanie kierowcy pojazdu za nim jest mniejsze), czyli również większą przepustowość.

[35] jako kryterium długości można przyjąć pewną funkcję kosztu, a znaki mogą sugerować trasy które mają najkrótszy czas lub najniższy koszt przejazdu

2.5.2. Modyfikacja sieci

Modyfikacja sieci jest najtrudniejszym i najbardziej kosztownym rozwiązaniem, równocześnie może przynieść duże korzyści. Fizyczna przebudowa sieci drogowej może znacząco poprawić przepustowość. Na przykład zbudowanie bezkolizyjnego lewoskrętu na skrzyżowaniu[36] zwiększy przepustowość na wszystkich kierunkach, które do tej pory musiały być zatrzymane, aby przepuścić ruch w lewo. Dobudowanie nowej drogi do już istniejącej sieci może odciążyć fragment sieci.

Z uwagi na wysokie koszty tych modyfikacji, za każdym razem należy przeprowadzić gruntowną analizę proponowanego rozwiązania. Projektowana droga zajmuje dodatkowe tereny, które często nie są własnością budującego (trzeba je wykupić) i mogą być niedostępne dla celów budowy, dlatego trudno jest zautomatyzować planowanie.

Oprócz dodawania i usuwania gałęzi w sieci drogowej można zmieniać kierunek ruchu dla wybranych pasów drogi. To rozwiązanie może być w pewnym stopniu zautomatyzowane. Zmiana kierunków ma sens wtedy, gdy natężenie ruchu zmienia się w czasie i w jedną stronę jest wyraźnie większe, przy czym o różnych porach inny kierunek jest dominujący.

Przykład 2.3. Trasa prowadząca do dzielnicy biznesowej w mieście ma 3 pasy. Rano, gdy większość ruchu odbywa się w kierunku tej dzielnicy, droga ma 2 pasy w kierunku tej dzielnicy i tylko jeden w kierunku przeciwnym. Po południu, gdy więcej pojazdów zaczyna wyjeżdżać z tego regionu niż do niego wjeżdżać, środkowy pas zostaje na chwilę zamknięty i kierunek na tym pasie zmienia się na przeciwny.

2.5.3. Priorytety dla komunikacji masowej

Głównym celem wprowadzenia priorytetów dla komunikacji masowej jest poprawa punktualności tego transportu, zmniejszenie jego szkodliwości dla środowiska oraz poprawa czasów przejazdu. Jeżeli równocześnie udaje się świadczyć ten sam poziom usług używając mniejszej liczby pojazdów w ruchu, to zmniejsza się zatłoczenie. Dodatkowym efektem zwiększenia punktualności i szybkości komunikacji miejskiej jest wzrost atrakcyjności tej formy komunikacji. Jeżeli pewien procent podróżujących przenosi się z własnych pojazdów do pojazdów komunikacji masowej, to zatłoczenie miasta spada. Przy zastosowaniu tego typu rozwiązań należy dobrze ocenić zyski dla komunikacji masowej w porównaniu do kosztów, jakie wskutek tego ponoszą pojazdy

[36] patrz. rozdział 3 na stronie 91 – eksperyment ze skrzyżowaniem typu T

użytkowników indywidualnych (wymuszenie zielonego światła dla autobusu/tramwaju, powoduje wydłużenie światła czerwonego na innych pasach).

Podobnym rozwiązaniem jest stosowanie pasów, na których priorytet mają pojazdy komunikacji masowej lub pasów przeznaczonych wyłącznie dla pojazdów tego typu. W Houston w Texasie wyznaczane są specjalne pasy dla pojazdów komunikacji masowej. Definicją takiego pojazdu jest tam pojazd wiozący co najmniej trzy osoby. Jeżeli 10% procent użytkowników zacznie łączyć się w grupy 3 osobowe, to zatłoczenie zmniejszy się o ok. 20–30%.

Przykładem zastosowania rozwiązań tego typu są pasy HOV w systemie Houston TranStar[37].

2.5.4. Podsumowanie

Ograniczenia czasowe oraz objętościowe dla tej pracy nie pozwalają na zaimplementowanie i przetestowanie wszystkich prezentowanych tutaj rozwiązań. Również ten opis nie jest pełny. Przedstawiony został tylko zarys metod optymalizacji oraz opisane zostały te, o których sądzę, że z powodzeniem można je zaimplementować w prezentowanym modelu *KRAKSIM*.

[37] Witryna internetowa systemu Houston TranStar — http://www.houstontranstar.org/, Informacje o pasach dla pojazdów komunikacji zbiorowej w systemie Houston TranStar — http://www.ridemetro.org/TransportationServices/Carpool_Vanpool_Services/HOV_system.asp

Rozdział 3

Koncepcja systemu

Rozdział „Koncepcja systemu” opisuje wymagania stawiane projektowi, przybliża koncepcję jego wykonania oraz pewne ograniczenia przyjęte, aby umożliwić wykonanie projektu w założonym czasie. Rozdział powstał na bazie analizy potrzeb związanych z modelowaniem i optymalizacją ruchu ulicznego.

3.1. Cel projektu

Celem tego projektu jest rozwój systemu do modelowania i optymalizacji ruchu miejskiego. Istniejący system *KRAKSIM* rozwijano w celu zwiększenia stopnia odzwierciedlenia rzeczywistości, a dla podsystemu optymalizacji opracowano nowe metody i algorytmy optymalizacyjne.

Główny nacisk został położony na sterowanie sygnalizacja świetlną. Podjęte zostały próby zaimplementowania dynamicznego systemu koordynacji pracy sąsiadujących skrzyżowań.

W celu uniknięcia korków zaproponowano system ostrzegania, który współpracuje z systemem przewidywania korków na podstawie ich wzorców. Ostrzeżenia mogą być przekazywane uczestnikom ruchu przy użyciu znaków o zmiennym tekście — *VMS*,

lub za pomocą specjalnych kanałów radiowych — TMC[1]. Aby umożliwić wykorzystanie znaków VMS, zaimplementowano model kierowcy, który umożliwia zmianę trasy w trakcie przejazdu.

Ponieważ system musi dawać możliwość analizy przebiegu symulacji, zaplanowano zbiór parametrów, które są obserwowane oraz sposób ich wyznaczania.

3.2. Wymagania i specyfikacja

3.2.1. Wymagania funkcjonalne

Poniżej przedstawione są podstawowe wymagania z perspektywy funkcjonalności systemu.

1. System ma umożliwić przeprowadzenie realistycznej symulacji ruchu:
   - symulacja powinna być przeprowadzana na możliwie niskim poziomie, aby umożliwić analizę oddziaływań pomiędzy pojazdami oraz lokalne zatłoczenia - mikrosymulacja,
   - czas w symulacji może być dyskretny — symulacja przeprowadzana jest w systemie turowym,
   - symulacja musi posiadać konfigurowalne parametry odpowiadające rozdzielczości czasu oraz przestrzeni (na etapie kompilacji aplikacji), w przypadku symulacji na poziomie pojedynczych pojazdów, jednostka długości przestrzeni powinna odpowiadać średniej długości pojazdu,
   - jednostka czasu powinna odpowiadać jednej sekundzie, jako jednostka odległości proponuje się 7,5 metra[2],
   - w ruchu uwzględnione będą tylko pojazdy drogowe jednego typu.
2. System ma umożliwić zbudowanie sieci drogowej, odpowiadającej rzeczywistej sieci dróg wybranego miasta:
   - w systemie występują elementy takie jak: droga, skrzyżowanie, sygnalizacja świetlna, parking (tu zaczynają się i kończą podróże),
   - droga jest krawędzią grafu, skrzyżowanie i parking są węzłami, a sygnalizacja świetlna to dodatkowy element działający we współpracy ze skrzyżowaniem,

[1] Kanały radiowe nadające specjalne komunikaty o ruchu drogowym to TMC (z ang. Traffic Message Channel). Kanały TMC z powodzeniem są stosowane w wielu krajach Europy i USA. W Polsce GDDKiA (http://www.gddkia.gov.pl/) prowadzi prace nad uruchomieniem kanałów TMC, które będą przesyłane z sygnałem RDS istniejących komercyjnych stacji radiowych.

[2] odpowiada to minimalnemu modelowi NaSch, minimalny to znaczy taki, którego uproszczenie powoduje znaczną utratę realizmu odwzorowania symulowanej rzeczywistości

- droga może być dwukierunkowa,
- na jednym kierunku jest tylko jeden główny pas ruchu,
- ruch na skrzyżowaniach odbywa się zgodnie z zasadami prawa o ruchu drogowym,
- można zbudować skrzyżowania równorzędne oraz skrzyżowania z wyróżnioną drogą główną, drogę główną wyróżnia się przez odpowiednie ustalenie priorytetów w macierzy pierwszeństwa,
- sygnalizacja świetlna działa zgodnie z ustalonym planem,
- możliwe jest zaprojektowanie sygnalizacji z kolizyjnymi strumieniami ruchu (wtedy ważne są zasady prawa o ruchu drogowym, np.: reguła „prawej dłoni"), oraz sygnalizacji bezkolizyjnej,
- parking, z punktu widzenia systemu może być węzłem takim, jak skrzyżowanie,

3. System powinien tak sterować pracą sygnalizacji świetlnej, aby optymalizować parametry ruchu:
   - system musi posiadać możliwość zmiany parametrów sterownika sygnalizacji świetlnej w trakcie trwania symulacji,
   - system musi implementować przynajmniej jeden algorytm sterowania sygnalizacją świetlną, który dostosowuje parametry sygnalizacji świetlnej do aktualnych warunków ruchu drogowego w taki sposób, że obserwowalne charakterystyki ruchu ulegają poprawie (przykładowe charakterystyki: średnia prędkość, liczba zatrzymań, liczba pojazdów na sekundę).
4. Formaty opisu mapy i ruchu powinny być w miarę proste oraz uniwersalne:
   - format map powinien być na tyle elastyczny, aby można było symulować jak największą liczbę sytuacji drogowych z użyciem skrzyżowań o różnorodnej budowie,
   - opis trasy zawiera jedynie węzeł startowy oraz końcowy i informację o czasie w jakim pojazdy powinny rozpoczynać podróż, węzły pośrednie zostaną wyznaczone przez generator tras,
   - opis trasy może opcjonalnie zawierać węzły pośrednie podróży,
   - plan podróży określa również liczbę pojazdów, które pojadą tą samą trasą, oraz rozkład prawdopodobieństwa z jakim te podróże mają być generowane,
5. System powinien zapewniać możliwość prezentacji przebiegu symulacji:
   - moduł statystyk powinien zbierać informacje o pojazdach oraz o drogach,
   - powinny istnieć statystyki zarówno o charakterze globalnym jak i szczegółowe,
   - moduł wizualizacji powinien w czytelny sposób prezentować przebieg symulacji,
   - system powinien przedstawić w graficzny sposób sieć dróg modelu miasta,

- wizualizacja przebiegu symulacji powinna uwzględniać pojedyncze pojazdy i sygnalizację świetlną,

6. Wymagania implementacyjne/architektury:
   - system powinien mieć modułową architekturę, tak aby można było budować pełny model symulacji używając różnych modułów, np.: zaimplementować różne algorytmy sterowania światłami jako oddzielne moduły A i B i móc wymieniać je poprzez konfigurację,
   - podsystemy symulacji i optymalizacji powinny być wyraźnie oddzielone, tak aby można było zmienić implementację symulacji (np.: zamienić model ruchu lub dodać nowe algorytmy sterowania światłami) oraz dodawać nowe metody optymalizacji sterowania ruchem,
   - moduł symulacji — jądro systemu — musi działać niezależnie, tak aby można go było wykorzystywać do budowy kolejnych modułów lub innych projektów,
   - moduł symulacji musi dawać możliwość sterowania pracą skrzyżowania poprzez blokowanie i odblokowywanie pewnych pasów (interfejs dla symulatora świateł drogowych),
   - model symulacji powinien posiadać własności, które w przyszłości umożliwią zrównoleglenie obliczeń,
   - konfiguracja modułów systemu może wyglądać jak budowanie rozwiązania z gotowych elementów.

Przykład budowy systemu z modułów: moduły modelu miasta (A) oraz generacji ruchu (B) będą podstawą do budowy całego systemu, moduł mikro-symulacji (C) będzie wykorzystywał moduły A i B, moduł informacyjny (D) będzie pobierał dane z symulacji C, moduł algorytmu sterowania sygnalizacją (E) będzie ustalał parametry sygnalizacji świetlnej na podstawie danych z modułu D, moduł decyzyjny (F) wpływa na symulację C używając parametrów obliczonych w module E, opcjonalnie moduł wizualizacji zaprezentuje przebieg symulacji użytkownikowi na podstawie danych z modułów A i D.

3.2.2. Przyjęte ograniczenia

Biorąc pod uwagę czas przeznaczony na wykonanie projektu oraz niezbędną do realizacji celów funkcjonalność wprowadzono pewne ograniczenia. W ramach tej pracy zostaną zaimplementowane oraz przetestowane tylko wybrane metody optymalizacji.

Największym ograniczeniem modelu jest istnienie dróg o tylko jednym pasie w jednym kierunku. Pas ten może się rozwidlać przed samym skrzyżowaniem, ale na całej

długości drogi może biec tylko jeden pas główny. Ograniczenie wynika z tego, że wprowadzenie dróg z ruchem wielopasmowym wymusza implementację reguł wyprzedzania i komplikuje sam model ruchu.

Kolejnym problemem jest taka implementacja symulacji pracy skrzyżowania, która w sposób realistyczny odzwierciedlałaby zachowanie kierowców na złożonych skrzyżowaniach (z wieloma pasami wlotowymi i wieloma wylotowymi, rozległe skrzyżowania w kształcie ronda, z wysepką). Wymaga to zaimplementowania dodatkowych reguł wjazdu na skrzyżowanie, oraz reguł jego opuszczenia. Na takich skrzyżowaniach występują skomplikowane systemy sygnalizacji świetlnej. To wszystko sprawia, że można wyodrębnić wiele rodzajów takich skrzyżowań. Zadanie to jest na tyle skomplikowane, że ze względu na ograniczenia czasowe nie będzie realizowane. Oczywiście do pewnego stopnia można symulować złożone skrzyżowania poprzez łączenie kilku prostych.

W tym modelu na pewnej drodze wlotowej do skrzyżowania może istnieć co najwyżej jeden pas służący do skrętu (lub jazdy na wprost) w określonym kierunku. Łącznie droga może mieć 3 pasy: do skrętu w lewo, do skrętu w prawo i do jazdy na wprost.

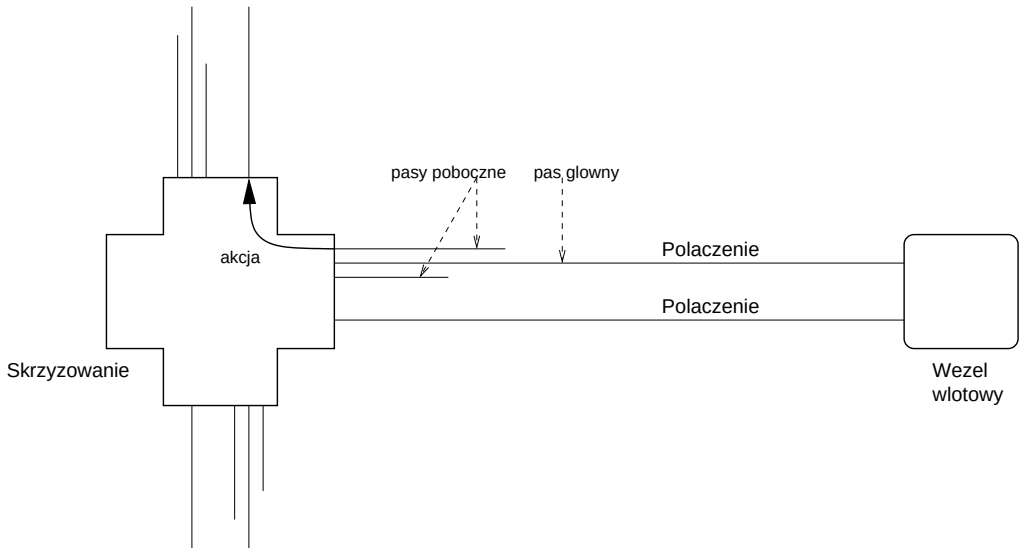
Skrzyżowania są bardzo ważnym elementem w symulacji ruchu ulicznego. Zakłada się wykonanie uniwersalnego modelu skrzyżowania stosując macierz pierwszeństwa. Macierz pierwszeństwa umożliwi określenie zasad pierwszeństwa przejazdu, które panują na skrzyżowaniu. Ten sposób opisu skrzyżowania jest skomplikowany, ale daje duże możliwości konfiguracji.

3.3. Model miasta

Jak napisano wcześniej, system ma mieć uniwersalną architekturę modułową. Uniwersalność jest oczywiście możliwa tylko do pewnego stopnia - konieczne jest istnienie pewnych pojęć i zasad wspólnych dla wszystkich modułów. W naszym przypadku takim szkieletem łączącym moduły jest *model miasta* przedstawiony w tym punkcie.

Po analizie wyróżnione zostały elementy wchodzących w skład sieci dróg, które są ważne z punktu widzenia symulatora i optymalizatora. Elementy te są przedstawione na rysunku 3.1.

Poniżej znajdują się krótkie definicje tych elementów oraz innych ważnych pojęć:
miasto całość sieci dróg,
węzeł pojęcie abstrakcyjne: węzeł wlotowy albo skrzyżowanie,
węzeł wlotowy miejsce, przez które samochody wkraczają i opuszczają model,



Rysunek 3.1. Elementy składowe modelu miasta

skrzyżowanie węzeł sieci dróg, w którym samochody mogą przejeżdżać z jednego połączenia na inne,

połączenie część drogi przeznaczona do ruchu w określonym kierunku, łączy dwa węzły; składa się z pasów, z których jeden zwany pasem głównym ciągnie się od początku do końca; pozostałe pasy to tzw. pasy poboczne,

połączenie wchodzące (z punktu widzenia węzła) połączenie, z którego samochody wjeżdżają do węzła,

połączenie wychodzące (z punktu widzenia węzła) połączenie, przez które samochody opuszczają węzeł,

pas ruchu patrz połączenie,

akcja przejazd jaki może wykonać samochód przez skrzyżowanie, łączy pewien pas połączenia wchodzącego z połączeniem wychodzącym, dla akcji definiuje się zbiór tzw. pasów nadrzędnych,

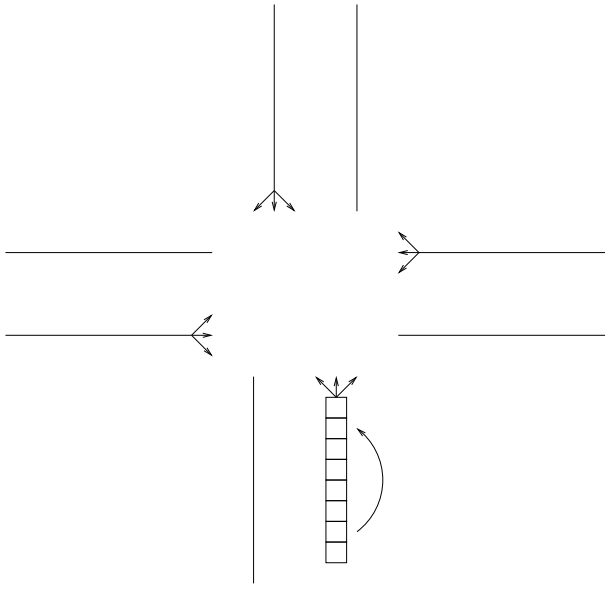
pas nadrzędny (względem akcji) pas wchodzący do skrzyżowania (pas połączenia wchodzącego), któremu należy ustąpić pierwszeństwo,

sygnalizacja świetlna element sterujący, który związany jest z jednym skrzyżowaniem i odpowiada za sterowanie blokadą pasów należących do połączenia wchodzącego do danego skrzyżowania.

3.4. Model symulacji ruchu

Symulacja ruchu na odcinku. Model symulacji ruchu będzie implementacją mikroskopowego modelu symulacji zbudowanego w oparciu o automaty komórkowe. Jest to model NaSch[17] (zastosowany w systemie TRANSIMS), który został już opisany w rozdziale 2.1.3 na stronie 14.

Symulacja ma rozdzielczość czasową 1 sekundy i przestrzenną 7,5 metra.



Rysunek 3.2. Komórkowo-turowy charakter symulacji. Przykładowe skrzyżowanie czterech dróg, z każdego pasa pojazdy mogą skręcać w prawo, w lewo oraz jechać na wprost. Na rysunku na jednym z pasów wyróżniono komórki. Symulacja ma charakter nieciągły. Symulacja ma rozdzielczość czasową 1 sekundy i przestrzenną 7,5 metra.

Symulacja przejazdu przez skrzyżowanie. Skrzyżowania obsługiwane są przez zestaw reguł wzorowany na tym, który zaproponowali autorzy systemu TRANSIMS. W pewnym sensie naśladuje zachowanie kierowcy mającego zamiar przejechać przez skrzyżowanie.

Zanim samochód x wykona akcję, sprawdzane są wszystkie pasy nadrzędne dla tej akcji. Dla każdego z pasów wykonywane są następujące kroki algorytmu:

I. Sprawdza się, czy dany pas nadrzędny (nadrzędny względem pasa, na którym znajduje się pojazd x, próbujący wykonać akcję) jest zablokowany[3]. Jeżeli pas

[3] Pas jest zablokowany wtedy, gdy sygnalizacja świetlna nadaje sygnał czerwony

jest zablokowany, to znaczy , to przechodzi się do następnego pasa nadrzędnego, w przeciwnym wypadku do następnego kroku.

II. Odnajduje się pierwszy (najbliższy skrzyżowaniu) samochód y na pasie, jeśli taki nie istnieje, to przechodzi się do następnego pasa nadrzędnego.

III. Wyznacza się odległość czasową samochodu y z kroku II od skrzyżowania (odległość podzielona przez aktualna prędkość). Jeśli ta odległość jest mniejsza niż z góry ustalona wartość graniczna (w projekcie TRANSIMS równa 3), to istnieje niebezpieczeństwo kolizji, więc samochód x nie może wykonać akcji i kończy się przeglądanie pasów nadrzędnych. W przeciwnym wypadku przechodzi się do następnego pasa nadrzędnego.

Jeśli przegląd pasów nadrzędnych nie został zakończony wcześniej w kroku III, to znaczy, że samochód może wykonać akcję (przejechać przez skrzyżowanie). Należy zwrócić uwagę na możliwość powstania zakleszczenia na skrzyżowaniu. Może zaistnieć taka sytuacja, w której żaden samochód nie może ruszyć, ponieważ musi ustąpić pierwszeństwa innemu (prosty przykład takiej sytuacji: 4 samochody chcące ruszyć na wprost na równorzędnym skrzyżowaniu z czterema odnogami). W modelu takie sytuacje są wykrywane przez algorytm wykrywania cyklu w grafie i rozwiązywane poprzez wylosowanie samochodu, który „wymusi" pierwszeństwo.

Taki zestaw reguł umożliwia przejazd przez skrzyżowania o dowolnym układzie drogi z pierwszeństwem przejazdu. Zasady pierwszeństwa ustalone są w konfiguracji topologii skrzyżowania. Symulacja sygnalizacji świetlnej na skrzyżowaniu wymaga tylko odpowiedniego blokowania i odblokowywania pasów w zależności od koloru światła.

3.5. Algorytmy sterowania sygnalizacją świetlną

W systemie używane będą 3 rodzaje sterowników sygnalizacji świetlnej: podstawowy sterownik cykliczny z fazami o ustalonej długości, sterownik acykliczny używający metody SOTL oraz sterownik acykliczny używający metody RL.

3.5.1. Stała konfiguracja

Jest to najprostszy sterownik sygnalizacji świetlnej. Działa on w oparciu o stałą konfigurację przygotowaną przez użytkownika. Do działania tego sterownika wystarczy zegar oraz wspomniana konfiguracja planu fazowego.

Konfiguracja planu fazowego opisuje wszystkie fazy oraz ich sekwencję. Pełny opis

jednej fazy sygnalizacji to kolory wszystkich świateł na danym skrzyżowaniu oraz czas trwania tej fazy.

Sygnalizacja umożliwia otworzenie drogi dla wielu strumieni ruchu jednocześnie. Jeżeli są to strumienie kolizyjne to pojazdy stosują się do reguł przejazdu przez skrzyżowanie podanych w podpunkcie 3.4 na stronie 44.

3.5.2. SOTL

Metoda SOTL (patrz rozdział 2.4.3 na stronie 29) wykorzystuje bardzo prostą zasadę: dla każdego pasa, wyznaczana jest wartość K_i (jest ona zarazem oceną pasa dla modułu decyzyjnego), która jest iloczynem ilości pojazdów na tym pasie przez ilość tur (jednostek czasu). Te wartości są liczone tylko wtedy, gdy świeci się światło czerwone.

W projekcie użyta jest metoda SOTL-phase, z tym, że minimalny czas palenia się świateł liczony jest na podstawie ilości pojazdów czekających w *strefie*. Ten minimalny czas trwania sygnału zielonego pozwala lepiej wykorzystać sygnalizację świetlną. Jeżeli w strefie czeka tylko jeden pojazd, to światło zielone zapali się na kilka sekund. Jeżeli w strefie czeka 20 pojazdów, to światło zielone zapali się na tak długo, aby 20 pojazdów mogło przejechać (wartość oszacowana z uwzględnieniem opóźnienia reakcji kierowców w ruszaniu z miejsca).

Strefa jest elementem wyznaczonym przez układ dwóch pętli indukcyjnych, jedna wejściowa i druga wyjściowa. Dzięki temu sterownik sygnalizacji posiada dane o ilości pojazdów czekających przed skrzyżowaniem (oczywiście dotyczy to tylko strefy). Długość strefy jest konfigurowalna.

Kolejną ważną zmianą w stosunku do algorytmu oryginalnego jest wprowadzenie możliwości przedłużenia bieżącej fazy na głównym kierunku ruchu, jeżeli na innych pasach nie ma pojazdów (zapobiega to niepotrzebnemu przełączaniu świateł).

Wartość parametru θ ustalono na długość strefy (w jednostkach długości, co odpowiada ilości pojazdów, jakie się zmieszczą w strefie) pomniejszoną o długość trwania żółtego światła. Taka wartość umożliwia zmianę światła na zielone, zanim pojazd dojedzie do skrzyżowania (ale tylko gdy nie ma ruchu z pozostałych kierunków), co poprawia działanie tej metody przy niewielkim natężeniu ruchu.

Konfigurowalne parametry modułu. Na etapie kompilacji lub uruchamiania można skonfigurować takie parametry tego modułu oceny:

- *zoneLength* - długość strefy,
- *carStartDelay, carMaxVelocity* - służą do szacowania minimalnej długości świecenia

się zielonego światła; pierwszy z nich to ilość czasu jaka upływa pomiędzy ruszeniem kolejnych samochodów stojących w korku, a drugi to maksymalna prędkość samochodu,

- *threshold*(θ) - próg, którego przekroczenie pozwala na branie pod uwagę wartości z danego pasa,
- *minimumGreen* - minimalny czas trwania zielonego światła, jeżeli jest tylko jeden pojazd, lub nie ma pojazdów w strefie (jeżeli nie jest stosowane przedłużanie fazy aktualnej)

3.5.3. RL

Metoda RL (opisana w rozdziale 2.4.4 na stronie 31) zastosowana zostanie w sposób, w jaki została opisana w rozdziale analizy rozwiązań. W ramach konfiguracji i przystosowania do modelu symulacji przeprowadzono niewielkie zmiany.

W celu lepszej adaptacji warto co pewną liczbę tur zmniejszać liczniki, tak aby zwiększyć wkład tego, co stało się w ostatnich turach, w wyznaczane prawdopodobieństwo. W tej implementacji co pewną (konfigurowalną) liczbę tur wszystkie liczniki dzielone są przez 2.

Minimalny czas świecenia się zielonego światła jest wyznaczany bardzo podobnie jak w przypadku algorytmu SOTL. Szacuje się jaki czas jest potrzebny samochodom z kolejki (i tylko im) do przejechania przez skrzyżowanie.

Ponieważ pobieranie do obliczeń danych statystycznych dotyczących całej drogi jest złożone obliczeniowo i trudne do zaimplementowania w rzeczywistej sieci drogowej, w tej metodzie, podobnie jak i w metodzie SOTL, wprowadzona została *strefa* przed skrzyżowaniem.

Konfigurowalne parametry modułu. Na etapie kompilacji lub uruchamiania można skonfigurować takie parametry tego modułu oceny:

- *zoneLength* - długość strefy,
- *discount* - czynnik γ ze wzorów,
- *halvePeriod* - co ile tur następuje dwukrotne zmniejszenie liczników (niedodatnia wartość powoduje, że nigdy nie będą zmniejszane),
- *carStartDelay, carMaxVelocity* - służą do szacowania minimalnej długości świecenia się zielonego światła; pierwszy z nich to ilość czasu jaka upływa pomiędzy ruszeniem kolejnych samochodów stojących w korku, a drugi to maksymalna prędkość samochodu.

3.6. Algorytmy współpracy sygnalizatorów na skrzyżowaniach

Do synchronizacji sygnalizacji świetlnej na sąsiednich skrzyżowaniach wybrano algorytm OptAPO (opisany w rozdziale 2.4.5 na stronie 32). Ta metoda wymaga istnienia planów sygnalizacji z priorytetem dla jednego kierunku. Aby możliwa była synchronizacja ze skrzyżowaniami w różnych kierunkach, sterownik sygnalizacji musi posiadać kilka planów.

W tej implementacji ograniczono się do wyboru kierunków północ-południe albo wschód-zachód. Oznacza to, że skrzyżowanie, które ma być uwzględnione w procesie synchronizacji, powinno posiadać dwa plany sygnalizacji.

Algorytm mediacji uruchamiany jest periodycznie. Plan sygnalizacji nie może być zmieniony w trakcie trwania cyklu, tylko po jego zakończeniu. Dlatego domyślnym ustawieniem okresu działania algorytmu jest czas, jaki jest potrzebny na pełną realizację jednego cyklu świateł (dla najdłuższego cyklu w modelu).

Na potrzeby tego algorytmu musi zostać wprowadzona implementacja systemu agentowego. Każdy sterownik sygnalizacji ma odpowiadającego mu agenta. Implementacja systemu agentowego powinna umożliwić komunikację agentów ze sobą. Po zakończonej mediacji, każdy agent ustawi wartość swojej zmiennej dotyczącej kierunku synchronizacji na odpowiadającym mu sterowniku skrzyżowania.

3.7. Predykcja wzorców zatłoczenia

Korek drogowy – zator na drodze, który spowodowany jest najczęściej przez awarię, wypadek czy nadmierny ruch drogowy. W tym przypadku rozważa się głównie korki spowodowane przez nadmierny ruch. W dalszych częściach pracy podjęta zostanie próba opisania systemu, który umożliwi przekierowanie nadmiernego ruchu i rozładowanie korków oraz ich zapobieganie. Do sterowania ruchem można użyć różnych metod przesyłania informacji, w tym wspomnianych wcześniej znaków o zmiennej treści.

W rozdziale 2.5.1 przedstawiono ideę zmniejszenia zatłoczenia poprzez sugerowanie kierowcom optymalnej trasy, bazując na aktualnych danych. Gdyby udało się opracować sposób na przewidywanie, jaki będzie rozkład natężenia ruchu w ciągu 30 najbliższych minut, można by zapobiegać korkom, a nie tylko je rozładowywać. Ta sekcja jest propozycją systemu oceny, przechowywania oraz wykorzystywania informacji o wzorcach korków.

3.7.1. Wzorzec zatłoczenia

System przechowywania informacji o wzorcach korków to prosta struktura bazodanowa, w której zapisane są *pary* lub *grupy dróg*, które wzajemnie na siebie wpływają. W szczególności interesujące są informacje, które pozwolą ocenić czy zakorkowanie ulicy A spowoduje zakorkowanie ulicy B.

Taka struktura może być stworzona jako wynik działania algorytmów z dziedziny *teorii grafów*. Jeżeli potraktujemy sieć dróg jako graf, to mając daną średnią przepustowość dróg, można ocenić jak zaburzony zostanie przepływ w grafie, gdy jedna z dróg zostanie zamknięta (zamknięcie ma przybliżyć sytuację zatoru). Inną metodą tworzenia informacji o zależnościach pomiędzy drogami jest zbieranie danych statystycznych z wielu przebiegów symulacji dla całego modelu miasta z różnymi warunkami początkowym.

Jeden wpis w bazie danych to prognoza zakorkowania ulicy B, gdy wiemy, że na ulicy A już występuje korek. W takiej strukturze danych każdy wpis powinien posiadać ocenę trafności dla prognozy. Wszystkie prognozy o trafności większej niż zadana tworzą *wzorzec zatłoczenia*. Odpowiednia ocena trafności oraz sposób wyboru tych informacji mogą być zrealizowane na różne sposoby. Idealny byłby system, który mógłby ocenić prawdopodobieństwo zagrożenia zatorem oraz wygenerować alternatywną trasę (która nie spowoduje zatoru w nowym miejscu). Do zrealizowanie tego typu systemu można użyć technik z dziedziny *sztucznej inteligencji*.

3.7.2. Wykrywanie zatoru oraz identyfikacja wzorca problemu

Jeżeli system będzie mógł wykryć i oznaczyć korek drogowy oraz ocenić jego klasę, to można zbudować system łączenia korków w grupy (wzorce korków) oraz system przewidywania sytuacji kryzysowej. W modelu mikro-symulacji systemu *KRAKSIM* wyróżnia się 4 klasy ruchu:

Szybki i luźny, gdy zagęszczenie pojazdów na drodze jest niskie - na poziomie 0-15%, a średnie prędkości bliskie maksymalnej dla danego fragmentu drogi (czyli średnia prędkości jest większa niż 75% maks. prędkości). W takiej sytuacji skierowanie dodatkowego ruchu na tą drogę lub wydłużenie czasu trwania sygnału stop na najbliższym skrzyżowaniu nie powinno spowodować zagrożenia korkiem. Nie trzeba kierować pojazdów zamierzających jechać tą drogą na inne trasy.

Płynny i gęsty, gdy zagęszczenie pojazdów na drodze jest bardzo duże - na poziomie 16-30%, a średnie prędkości w dalszym ciągu są wysokie - powyżej 50-60% maksymalnej prędkości dla danego fragmentu drogi. W tym stanie droga osiąga

swoją maksymalną przepustowość i zwiększenie liczby pojazdów lub zwiększenie czasu oczekiwania na czerwonym świetle może spowodować zakorkowanie się drogi. Ponieważ zagęszczenie pojazdów jest duże, to zachowanie jednego kierowcy ma bezpośrednio wpływ na zachowanie kierowcy pojazdu jadącego za nim, a propagacja takiego zdarzenia (np. hamowanie) jest bardzo szybka. Można przekazać kierowcom informację, że na tej trasie panuje duży ruch, natomiast nie ma potrzeby proponowania nowej trasy.

Zatłoczony, gdy zagęszczenie przekracza 30%, a średnia prędkość jest nieduża - poniżej 50% maksymalnej.

Korek, zagęszczenie pojazdów na drodze jest maksymalne, ok 50%, a średnia prędkość bardzo niska, ponieważ pojazdy głównie stoją. Można oszacować że przeciążenie drogi zaczyna się gdy prędkości spadają poniżej 30 procent prędkości maksymalnej.

System ewaluacyjny ocenia klasę ruchu na podstawie danych z monitoringu. Po określeniu klasy ruchu, dane (zawierające identyfikator drogi, klasę ruchu oraz czas obserwacji) zostają zapisane w bazie danych. Konieczne jest zaprojektowanie metod eksploracji tych danych w celu utworzenia wzorców zagrożenia.

Dane o stanie ulic z kilku obserwacji (np.: o czasie T [4], $T+1$ i $T+2$) pozwalają na zaobserwowanie zmian sytuacji drogowej w pewnym otoczeniu.

Klasy ruchu są pewną abstrakcją oceny gęstości ruchu i średniej prędkości. Dzięki ich wyodrębnieniu można uprościć system decyzyjny oraz struktury danych służące do przechowywania wzorców. Jeżeli dany odcinek drogi A ma klasę *zatłoczony*, to należy przeprowadzić ocenę zagrożenia. Ocena polega na wyszukaniu w bazie wzorców danych o klasach ruchu na drogach łączących się z drogą A, gdy ruch na A ma klasę *zatłoczony*. Jeżeli można znaleźć sytuację podobną do aktualnej, to na podstawie danych z następnego odczytu wnioskuje się jakie jest zagrożenie i jak zmieni się ruch na drogach łączących się z A (jest to predykcja na podstawie danych historycznych). Gdy według danych z predykcji klasa ruchu na drodze A zmienia się na *korek*, to należy podjąć działania zapobiegawcze.

System decyzyjny może ocenić, gdzie skierować ruch na podstawie klas ruchu na sąsiednich drogach. Jeżeli na drodze B ruch ma klasę *luźny* lub *płynny*, to można skierować tam ruch, w innym wypadku istnieje ryzyko stworzenia zatłoczenia, jeżeli ruch zostanie skierowany na drogę B. Na podstawie decyzji wykonywane są odpowiednie działania.

[4] T oznacza czas w którym została określona klasa ruchu, $T+1$ to czas następnej obserwacji

Kierowanie ruchem jest możliwe jedynie poprzez przekazywanie kierowcom informacji o aktualnym stanie ruchu i ewentualnie informacji o proponowanym objeździe. Takie informacje mogą być nadawane globalnie dla wszystkich kierowców, a zainteresowani wybiorą te, które ich interesują. W praktyce jest to możliwe do wykonania przy użyciu *stacji radiowych* (kanały TMC-RDS). Kolejną możliwością dotarcia do kierowcy z sugestią od systemu decyzyjnego jest wykorzystanie *znaków o zmiennym tekście* (idea opisana w punkcie 2.5.1 na stronie 34). Takie znaki mogą wyświetlać przewidywany czas przejazdu, ostrzeżenia o korku lub informacje o proponowanym objeździe. W ten sposób informacje są kierowane bezpośrednio do zainteresowanych i mogą mieć większą skuteczność.

Aby system można było zaimplementować, należy opracować konkretną definicję wzorca problemu oraz zaprojektować struktury danych, umożliwiające przechowywanie informacji o sytuacji drogowej. Te struktury danych powinny dawać możliwość analizy danych w celu predykcji zmian sytuacji drogowej. Ważnymi elementami są: podsystem predykcji oraz podsystem decyzyjny. Podsystem decyzyjny współpracy z podsystemem wykonawczym, który na podstawie decyzji steruje systemami informacyjnymi.

Ograniczenia tej pracy sprawiły, że to rozwiązanie nie zostało wybrane do pełnej realizacji. Zaprojektowany został globalny system informacyjny, który zapewnia kierowcom bieżące dane o średnim czasie przejazdu przez konkretne drogi (jest to implementacja idei TMC). Na podstawie tej informacji kierowcy sami ustalają objazdy. Dzięki temu można ocenić wpływ aktualnych informacji oraz dynamicznego wyznaczania objazdów na ruch miejski.

3.8. Obserwowane parametry modelu

Jednym z celów modelowania ruchu jest analiza jego parametrów. Projektowany system będzie oferował możliwość obserwacji parametrów lokalnych oraz globalnych. Parametry mogą określać stan modelu w danej chwili oraz wartości średnie uzyskane po zakończeniu symulacji.

Głównym narzędziem służącym do uzyskania informacji o podstawowych parametrach modelu są pętle indukcyjne instalowane na początku i na końcu pasa ruchu. Dzięki takiemu położeniu pętli, możliwe jest obliczenie liczby pojazdów znajdujących się na danym pasie oraz liczby pojazdów, które przejechały tym pasem od rozpoczęcia symulacji. Te dane powinny być zapisywane po każdej turze. Same pętle dają również możliwość oceny prędkości przejeżdżających pojazdów. Analiza ilości przejazdów tura

po turze umożliwia wyznaczenie ilości pojazdów przejeżdżających dany odcinek drogi na minutę, a to pozwala na zaobserwowanie zmian w dynamice systemu.

Ważnym źródłem danych szczegółowych w modelu wirtualnym są kierowcy. Kierowca rejestruje czas wjazdu na konkretny pas oraz czas wyjazdu z niego, dzięki temu można dokładnie ocenić czas przejazdu oraz średnią prędkość na pasie. Dane są przechowywane dla każdego pasa ruchu oddzielnie oraz są sumowane lub uśredniane po przejeździe każdego kierowcy (nie planuje się przechowywania indywidualnych danych dla pojedynczego kierowcy, chociaż jest to technicznie możliwe). Tak liczona jest średnia prędkość na pasie, oraz średni czas przejazdu przez pas.

W podobny sposób uzyskuje się dane dotyczące podróży, przy czym tutaj kierowca rejestruje czas rozpoczęcia i zakończenia podróży. Zsumowanie czasów wszystkich podróży i podzielenie tej sumy przez liczbę podróży daje w wyniku globalny parametr określający średnią czas przejazdu w modelu. Każdy kierowca rejestruje również długość przejechanej trasy. To umożliwia uzyskanie średniej prędkości oraz średniej długości trasy dla całego modelu.

Wszystkie wyniki pomiarów są zapisywane w plikach wynikowych i dostępne do analizy po zakończeniu przebiegu symulacji.

3.9. Podsumowanie koncepcji

Projektowany system symulacji będzie zbudowany przy użyciu automatów komórkowych. Wybrano metodę mikroskopową NaSch, która wystarczająco dobrze oddaje rzeczywistość.

Dane z mikrosymulacji gromadzone są przez moduł monitoringu oraz udostępniane modułowi oceny. Moduł oceny wyznacza — na podstawie tych danych — pewne wartości, które będą wejściem dla metod optymalizacji.

Głównym problemem optymalizacji jest sterowanie ruchem w celu poprawy charakterystyk ruchu (głównie czas przejazdu i średnia prędkość). Sterowanie ruchem jest możliwe przez sygnalizacją świetlną. Sygnalizatory będą miały możliwość zmiany swojej konfiguracji na podstawie lokalnych danych (metody Rl oraz SOTL) albo na podstawie informacji uzyskanych z sąsiednich skrzyżowań. Do tego celu należy zaimplementować system agentowy. Każdemu skrzyżowaniu odpowiada agent. Agent skrzyżowania komunikuje się z sąsiadującymi agentami zgodnie z zasadami narzuconymi przez algorytm mediacji (OptAPO). Po wybraniu rozwiązania każdy agent skrzyżowania wprowadza zmiany w ustawieniach sterownika sygnalizacji.

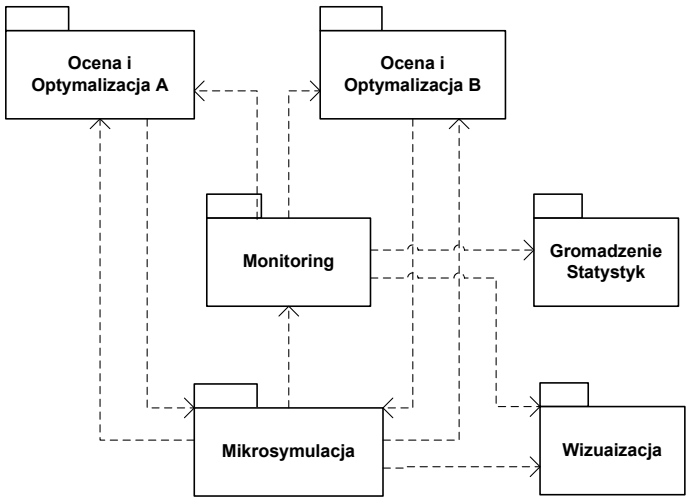
Inny sposób sterowania ruchem to wpływanie bezpośrednio na uczestników ruchu

drogowego. Poprzez przedstawianie kierowcom aktualnych informacji o stanie ruchu, można ich skłonić do zmiany trasy. Informacje przedstawiane są za pomocą znaków o zmiennej treści (opcją są wiadomości radiowe). Moduł oceny oblicza średni czas przejazdu dla konkretnych dróg w modelu. Na podstawie czasów przejazdu system sterowania znakami o zmiennej treści ustawia odpowiednie wiadomości na każdym ze znaków. Treścią wiadomości jest optymalna prędkość na odcinku drogi za znakiem, a w przypadku zatoru informacja ostrzegawcza o korku (wraz z szacowanym czasem przejazdu) oraz proponowany objazd. Kierowcy analizują treść znaków (np. poprzez porównanie czasu przejazdu do wartości średnich) i podejmują decyzje o zmianie trasy.

Rozdział 4

Projekt systemu

Rozdział „Projekt systemu” opisuje wszystkie elementy zrealizowanego systemu informatycznego. Ten rozdział opisuje również ogólną architekturę systemu oraz sposób implementacji rozwiązań wybranych w fazie analitycznej (i opisanych w rozdziale 3 „Koncepcja systemu”).



Rysunek 4.1. Ogólna architektura systemu, przykładowa konfiguracja budowy systemu z podsystemów.

W systemie musi równocześnie współpracować wiele podsystemów np.: podsystem symulacji, podsystem monitoringu, podsystem sterowania sygnalizacją świetlną, podsystem optymalizacji sygnalizacji świetlnej, podsystem gromadzenia danych statystycznych. Przykładowy system mógłby wyglądać tak jak pokazano na rys. 4.1.

Aby umożliwić wykorzystanie podsystemów, których implementację można zmieniać, zaproponowano budowę systemu z modułów. Pierwszy punkt tego rozdziału opisuje zasadę projektowania modułów oraz sposób budowy całego systemu z tych modułów.

W dalszej części rozdziału opisano podstawowe elementy systemu symulacji (takie jak: miasto, droga, węzeł, jądro) oraz moduły. Opis modułu sprowadza się do przedstawienia jego zadań oraz wyjaśnienia szczegółów jego budowy. Interfejsy modułów zilustrowano odpowiednimi diagramami UML.

Później przedstawiona jest dynamika systemu symulacji, czyli przebieg całej pętli symulacji (gdzie zaznaczono współpracę modułów). W ostatnim punkcie rozdziału opisano konkretne implementacje modułów. W większości są to implementacje metod modelowania i optymalizacji ruchu opisanych w rozdziale 3(str. 38).

4.1. Modułowa Architektura

Opracowano koncepcję umożliwiającą tworzenie dowolnych modułów, o ile tylko pasują one do wcześniej zaprezentowanej (patrz. 3.3, str. 42) reprezentacji sieci dróg w mieście.

Centralnym składnikiem tej koncepcji jest *jądro*. W jądrze tworzy się z elementów przedstawionych na rysunku 3.1 topologię sieci dróg. Elementy w jądrze nie posiadają żadnej innej funkcjonalności niż ta pozwalająca na odkrywanie tej topologii.

Do elementów można przyporządkowywać tzw. *rozszerzenia*. Rozszerzenie dodaje do elementu nowe dane i potrafi realizować nowe operacje. Rzeczywisty element sieci dróg jest więc w konkretnym symulatorze reprezentowany przez element jądra i wszystkie rozszerzenia do niego przyporządkowane. Funkcjonalność takiego elementu jest wypadkową funkcjonalności wszystkich rozszerzeń. Przykładowo, do każdego pasa może zostać przyporządkowane rozszerzenie do symulacji samochodów po nim jadących oraz rozszerzenie, które zlicza samochody przejeżdżające tym pasem. Ostatecznie możemy zasymulować ruch na każdym pasie i sprawdzić ilość samochodów, które do danej chwili nim przejechały.

Współpracę pomiędzy rozszerzeniami różnych elementów najlepiej oprzeć na interfejsach. Dla danej funkcjonalności (np. symulacji ruchu samochodów) dobrze jest

zdefiniować jakie operacje ma oferować rozszerzenie połączenia, jakie rozszerzenie skrzyżowania, jakie rozszerzenie węzła wlotowego itd. Jeśli rozszerzenia należące do innej funkcjonalności będą wykorzystywały tylko operacje zdefiniowane w interfejsach, to łatwo uzyskuje się wymienność rozszerzeń.

Zestaw rozszerzeń realizujących konkretną funkcjonalność przyporządkowanych do elementów jądra to *moduł*. Załóżmy, że sieć dróg składa się z dwóch skrzyżowań: X, Y i połączenia A. W jądrze będą istniały:

- skrzyżowanie X,
- skrzyżowanie Y,
- połączenie A.

Moduł symulacyjny będzie się więc składał z:

- rozszerzenia symulacyjnego skrzyżowania X,
- rozszerzenia symulacyjnego skrzyżowania Y,
- rozszerzenia symulacyjnego połączenia A.

Co istotne, powiązania oddające topologię miasta znajdują się w jądrze. Rozszerzenia nie muszą ich duplikować - dzięki temu, że każde rozszerzenie jest przyporządkowane pewnemu elementowi jądra, to może wykorzystać informacje dotyczące powiązań w nim zawarte.

Podobnie jak w wypadku pojedynczych rozszerzeń, warto także współpracę pomiędzy modułami oprzeć na interfejsach. Jeśli zbierzemy razem interfejsy oferowane przez każdy typ elementu, to uzyskamy interfejs modułu. Pisząc jaśniej, interfejs modułu składa się z: interfejsu rozszerzenia skrzyżowania, interfejsu rozszerzenia węzła wlotowego, interfejsu rozszerzenia połączenia itd.

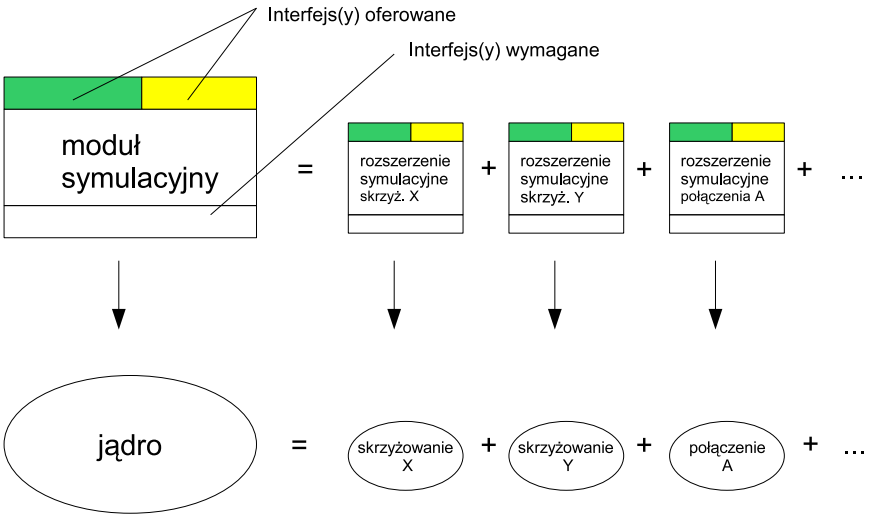
Rysunek 4.2 ilustruje powyższe rozważania. Pokazuje on, że:

- jądro jest zbudowane z elementów reprezentujących rzeczywiste elementy w sieci dróg i każdemu rzeczywistemu elementowi odpowiada pewien element (obiekt) w jądrze,
- moduł jest zbudowany z rozszerzeń; każdy element jądra ma w module jedno rozszerzenie[1],
- oferowane interfejsy modułu jako całości przekładają się na interfejsy oferowane przez poszczególne rozszerzenia.

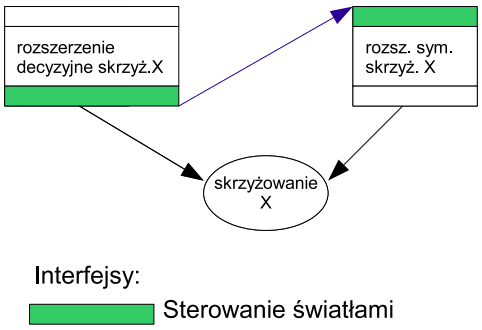
Rysunek 4.3 pokazuje na jakich zasadach odbywa się współpraca pomiędzy rozszerzeniami tego samego elementu. Zielonym kolorem zaznaczono interfejs sterowania światłami. Rozszerzenie symulacji skrzyżowania oferuje interfejs sterowania (co jest

[1] ściśle rzecz biorąc, co najwyżej jedno rozszerzenie, bo nie każdy element musi takowe posiadać

zaznaczone poprzez umieszczenie koloru tego interfejsu na górnej części prostokąta), a rozszerzenie decyzyjne wymaga do działania komponentu oferującego ten interfejs (kolor interfejsu na dolnej części prostokąta). Rozszerzenia to różne obiekty, które mogą korzystać ze swej funkcjonalności dzięki definicji interfejsów.



Rysunek 4.2. Ilustracja pojęć



Rysunek 4.3. Koncepcja współdziałania rozszerzeń pewnego elementu

4.2. Podstawowe elementy systemu

W tym punkcie przedstawiona jest architektura systemu do symulacji i optymalizacji ruchu miejskiego. Architektura ta opiera się na koncepcji uprzednio opisanej, czyli jej centralnym elementem jest jądro, a współpracę pomiędzy modułami zapewnia definicja interfejsów modułów.

Definicja interfejsów wymaga poczynienia dalszych założeń i ograniczeń. Kluczowym jest przystosowanie ich do symulacji opartej na automacie komórkowym. Powoduje to, że:

- pasy mają budowę komórkową – są podzielone na komórki o jednakowej długości,
- ruch może odbywać się tylko pomiędzy komórkami,
- symulacja ma charakter turowy.

Długość odcinka połączenia/trasy to nic innego jak liczba jego komórek, czas przejazdu to liczba tur symulacji, które zajęło pokonanie go. Jako prędkość (chwilową) można więc przyjąć ilość komórek pokonanych w turze. Prędkość maksymalna może być inna dla każdego odcinka.

Długości są jawnie podawane dla każdego połączenia, pomimo że określane są też współrzędne geometryczne węzłów. Współrzędne służą tylko do wizualizacji.

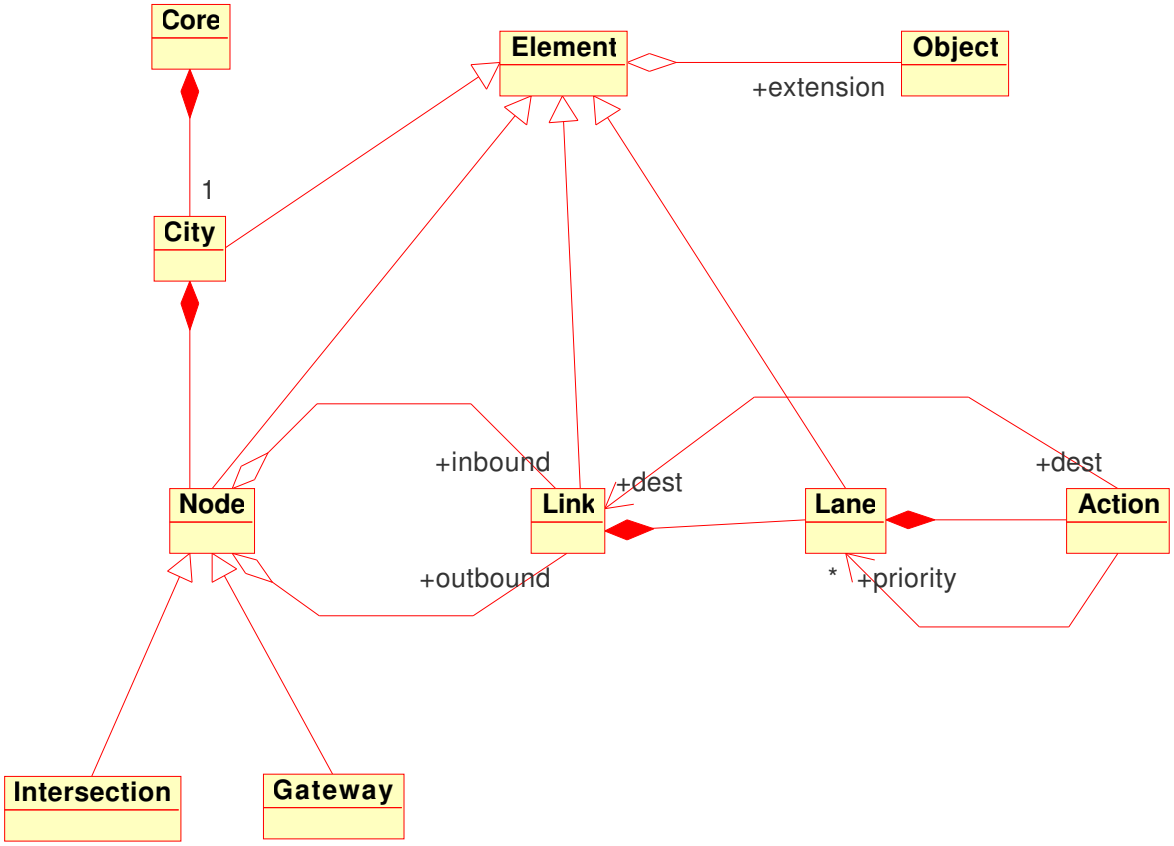
Dodatkowo, w interfejsach pojawia się pojęcie *kierowcy (driver)*. Kierowca to obiekt implementujący interfejs `Driver`. Jest to obiekt przyporządkowywany samochodowi w czasie, gdy porusza się przez miasto. Jedyną publiczną metodą tego interfejsu jest `Iterator<Link> updateRouteFrom(Link sourceLink)`, która ma umożliwić wybór trasy przez kierowcę. Twórca innego modułu symulacji może wykorzystać klasę kierowcy do innych celów (np.: zaimplementować różne rodzaje pojazdów/kierowców), ale wtedy musi dostarczyć implementację konkretnego kierowcy[2].

W tym punkcie znajduje się jedynie opis interfejsów modułów i wskazanie powiązań pomiędzy modułami. Konkretne moduły realizujące przypisane im interfejsy są opisane w punkcie następnym. Należy zauważyć, że mają one jedynie charakter przykładowy. Nowy moduł, realizujący identyczny kontrakt jak pewien przykładowy, powinien dać się bez problemu podstawić w miejsce przykładowego.

[2] niestety, w takim wypadku może używać konkretnej klasy kierowcy jedynie poprzez rzutowanie; ten fragment kodu nie jest uniwersalny, ponieważ trudno zbudować uniwersalnego kierowcę, nie było to również założeniem projektowym

4.2.1. Klasy tworzące sieć dróg

Diagram z rysunku 4.4 przedstawia podstawowe klasy jądra służące do reprezentacji sieci dróg.



Rysunek 4.4. Diagram podstawowych klas jądra systemu

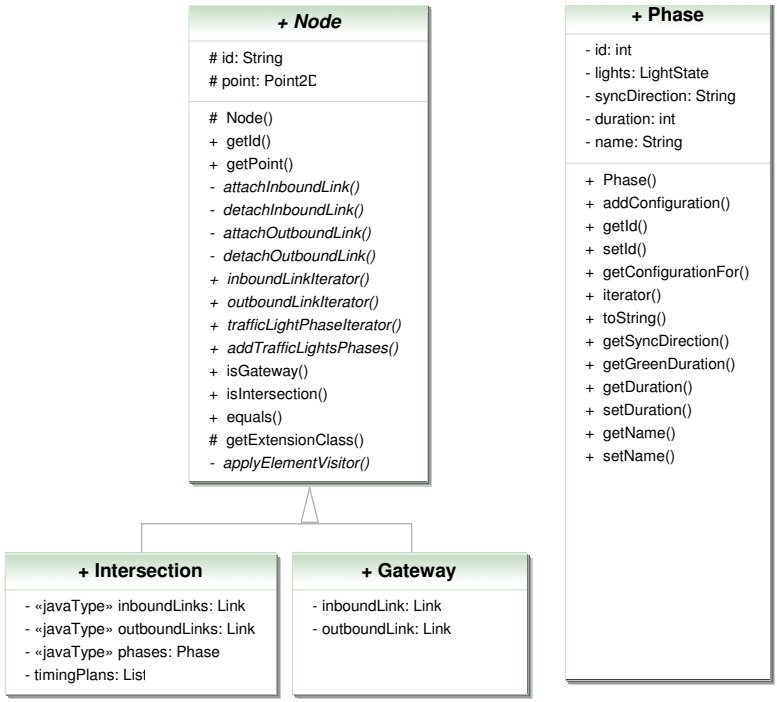
`Element` to klasa bazowa dla wszystkich elementów modelu miasta. W tej klasie zdefiniowane są mechanizmy umożliwiające używanie rozszerzeń.

Klasa `City` służy do agregacji obiektów typu `Node` oraz `Link` – reprezentuje całe miasto. Dla ułatwienia rozdzielono kolekcje *węzłów* typu `Gateway` od `Intersection`. Klasa `City` jest fabryką *połączeń*, *skrzyżowań* oraz *węzłów we/wy*. Metody zaczynające się od `create` służą właśnie do tworzenia elementów topologii miasta. W całym projekcie metody te używane są jedynie w momencie wczytywania topologii miasta z pliku konfiguracyjnego. `City` zapewnia również iteratory umożliwiające sprawne przeglądanie topologii oraz metody `find` pomagające znaleźć element używając identyfikatora.

Abstrakcyjna klasa `Node` reprezentuje *węzeł* sieci dróg. Klasy odpowiadające specyficznym typom *węzłów* to: `Gateway` – węzeł wlotowy oraz `Intersection` – skrzyżowanie. `Node` oferuje jedynie implementację mechanizmu rozszerzeń, posiada identyfi-

kator oraz współrzędne. Klasa *węzła* deklaruje abstrakcyjne metody obsługi połączeń wchodzących, połączeń wychodzących i sygnalizacji świetlnej, które konkretne klasy dziedziczące muszą implementować.

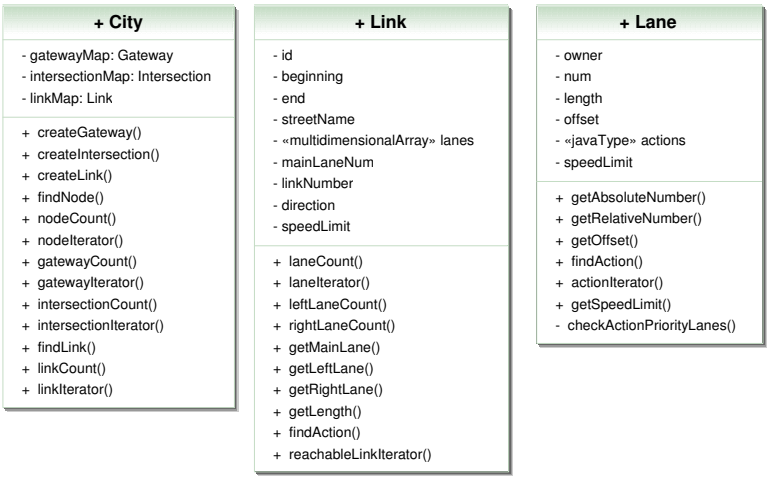
Kolejnym elementem jądra jest `Phase` — jedna *faza sygnalizacji świetlnej*. `Phase` posiada atrybuty: długość (czas trwania), nazwa, kierunek synchronizacji oraz lista stanów świateł (`LightState`). *Stan światła* określa jednoznacznie światło (na którym pasie i dla której akcji) i jego aktualny kolor. Lista *faz* (Phase) tworzy poprawny plan sygnalizacji świetlnej dla jednego skrzyżowania.



Rysunek 4.5. Klasy elementów: węzeł, skrzyżowanie, węzeł we/wy, faza cylku sygnalizacji.

Połączenie biegnące pomiędzy dwoma *węzłami* odzwierciedla klasa `Link`. Połączenie biegnące *do* węzła to połączenie *wchodzące (inbound)*, a biegnące *od* niego to połączenie *wychodzące (outbound)*. Takie połączenie to droga jedno lub dwukierunkowa. Klasa `Link` oferuje jedynie metody potrzebne do skonstruowania poprawnej drogi, iterator pasów należących do tej drogi oraz iterator dróg, do których można dojechać pasami tej drogi. Pasy do skrętu w lewo oraz prawo są udostępnione przez metody `getRightLane, getLeftLane`. `Link` posiada atrybut `streetName` odpowiadający nazwy ulicy, co umożliwia łatwiejszą identyfikację połączenia przez użytkownika.

Połączenie składa się z *pasów* reprezentowanych przez obiekty klasy `Lane`. Każdemu pasowi przyporządkowany jest zestaw *akcji* – `Action`, które można z niego wykonać. *Akcja* ma swój cel – połączenie oraz zbiór pasów nadrzędnych, na których pojazdy mają pierwszeństwo, a więc należy tym samochodom ustąpić pierwszeństwa przed wykonaniem akcji. `Lane` posiada metody, które umożliwiają odkrycie możliwych akcji (`findAction`), kolejny numer w strukturze drogi, przesunięcie (offset) względem początku drogi (gdy pas jest krótszy niż cała droga) oraz limit prędkości.



Rysunek 4.6. Klasy elementów: miasto, połączenie, pas.

Diagram 4.4 ukazuje też mechanizm rozszerzania – każdy element agreguje rozszerzenia do niego przypisane.

4.2.2. Interfejsy modułów

Każdy z przedstawionych w tym podpunkcie interfejsów modułów składa się z interfejsów dla poszczególnych klas rozszerzeń. Najczęściej jeden interfejs modułu definiuje interfejsy rozszerzeń tylko dla niektórych typów elementów (np. tylko dla miasta, jak interfejs decyzyjny). To, że interfejsy dla rozszerzeń innych typów nie istnieją nie oznacza, że moduł nie może zawierać rozszerzeń innych typów – w takim wypadku nie będą one po prostu publicznie dostępne. Z drugiej strony, moduł nie musi przyporządkowywać rozszerzenia każdemu elementowi.

Oprócz interfejsów rozszerzeń wymieniono też tzw. interfejsy dodatkowe. Interfejsy dodatkowe nie są implementowane przez rozszerzenia, pojawiają się jedynie jako typy w sygnaturze ich metod.

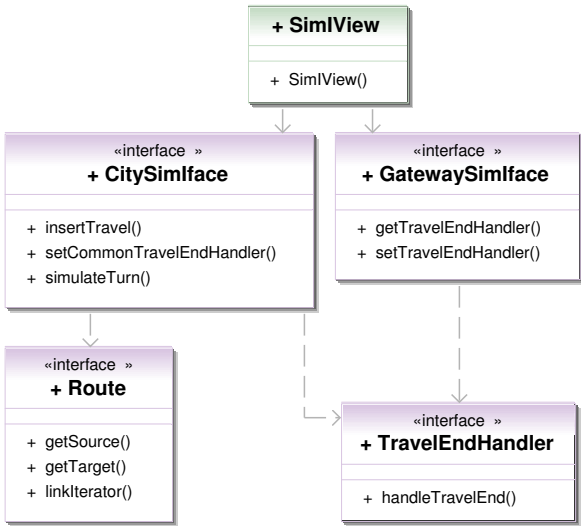
Nie definiuje się interfejsów i nie rozszerza się elementów typu `Action` (akcji). Podjęta została taka decyzja, ponieważ nie znaleziono dla tego zastosowania.

Interfejs symulacji ruchu samochodów

Interfejs symulacji ruchu — SimIView(patrz 4.7) zdefiniowany jest w pakiecie `pl.edu.agh.cs.kraksim.iface.sim`.

Interfejs ten służy do przeprowadzania symulacji ruchu samochodów tura po turze oraz umieszczaniu samochodów w symulacji. Pisząc *symulacja ruchu samochodów* mamy na myśli przesuwanie samochodów po połączeniach. Nie jest oferowana żadna kontrola nad tym (np. sterowanie światłami). Cały interfejs składa się z widoku `SimIView`, który umożliwia pobranie interfejsów miasta i przejścia(węzeł we/wy) — `CitySimIface, GatewaySimIface`. Aby możliwe było zrealizowanie funkcjonalności podróży, oferowane są interfejsy dodatkowe: TravelEndHandler — interfejs obiektu obsługującego zdarzenia zakończenia podróży oraz Route — interfejs trasy.

Samochód umieszcza się w symulacji i podaje jego *trasę* pomiędzy dwoma węzłami wlotowymi. Przejazd pomiędzy tymi węzłami zwany jest *podróżą*. Przebieg podróży oraz samej symulacji zależy całkowicie od implementacji.



Rysunek 4.7. Interfejs modułu symulacji.

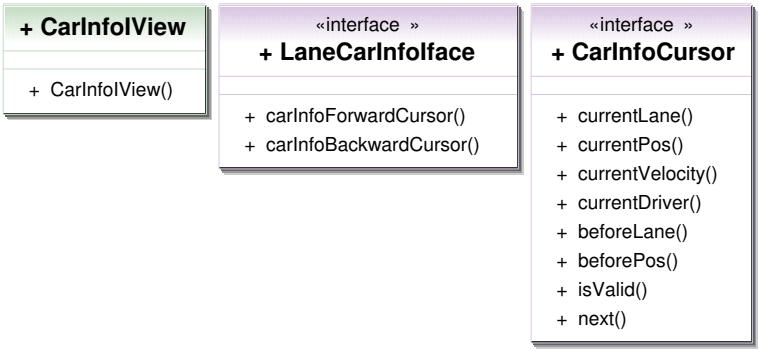
CitySimIface — Interfejs rozszerzenia elementu *miasto* (`City`), oferuje sygnatury metod: setCommonTravelEndHandler(TravelEndHandler handler) — ustawia obiekt obsługujący zdarzenie zakończenia podróży wspólny dla wszystkich węzłów wlotowych, insertTravel(TravelEndHandler handler) — umieszczenie samochodu z trasą w symulacji, simulateTurn — dokonanie tury symulacji.

GatewaySimIface — Interfejs rozszerzenia elementu *węzeł wlotowy* (Gateway): setTravelEndHandler(TravelEndHandler handler) — ustawia obiekt obsługujący zdarzenie zakończenia podróży, TravelEndHandler getTravelEndHandler() — zwraca obiekt obsługujący zdarzenie zakończenia podróży.

Interfejs modułu symulacji oferuje również 2 interfejsy dodatkowe: TravelEndHandler i Route. TravelEndHandler deklaruje sygnaturę metody, do obsługi zdarzenia końca podróży — handleTravelEnd(Object driver). Ta metoda jest wywoływana, kiedy pojazd kończy swoją trasę. Route jest interfejsem który powinna implementować klasa konkretna opisu trasy. Interfejs deklaruje sygnatury metod pobrania informacji o źródle i celu podróży: Gateway getSource(), Gateway getTarget(); oraz iterator kolejnych połączeń (dróg) trasy. Iteracja powinna odbywać się na bieżąco w trakcie symulacji, umożliwi to zastosowanie implementacji, które generują trasę dynamicznie.

Interfejs informacji o samochodach

Interfejs zdefiniowany jest w pakiecie `pl.edu.agh.cs.kraksim.iface.carinfo`



Rysunek 4.8. Interfejs modułu informacji o pojazdach.

Interfejs ten służy do uzyskiwania informacji o samochodach uczestniczących w ruchu w mieście. Ponieważ pojazdy poruszają się po pasach, jedynym interfejsem rozszerzeń jest tu LaneCarInfoIface — interfejs rozszerzenia elementu *pas*(Lane). Implementacja LaneCarInfoIface powinna umożliwić uzyskanie kursora wskazującego samochody od początku pasa jak i od końca pasa, przy użyciu metod odpowiednio: carInfoForwardCursor(), carInfoBackwardCursor(). Sam kursor zdefiniowany jest poprzez interfejs dodatkowy CarInfoCursor, mający z założenia funkcjonować podobnie do iteratora. Metody z CarInfoCursor:

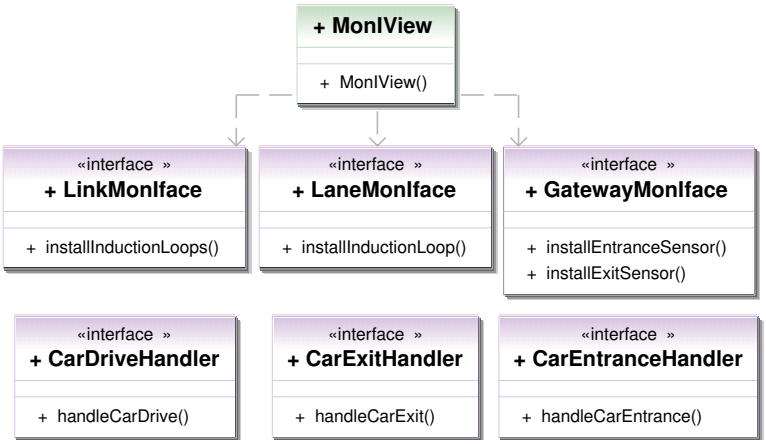
- Lane currentLane() — zwraca pas, na którym samochód zakończył ostatni ruch,
- int currentPos() — zwraca aktualną pozycję pojazdu na pasie,

- int currentVelocity() — zwraca ilość komórek, jaką samochód przebył w ostatnim ruchu,
- Object currentDriver() — zwraca kierowcę samochodu,
- Lane beforeLane() — zwraca pas, z którego samochód rozpoczynał ostatni ruch,
- int beforePos() — zwraca pozycję pasa, z której samochód rozpoczynał ostatni ruch,
- boolean isValid() — zwraca czy kursor wskazuje na jakiś samochód,
- void next() — przesuwa kursor do następnego samochodu.

Interfejs monitoringu

Interfejs zdefiniowany jest w pakiecie `pl.edu.agh.cs.kraksim.iface.mon`

Interfejs ten pozwala na zarejestrowanie akcji, które będą wykonane, gdy *samochód* przejedzie przez określony punkt *połączenia*, pojawi się lub zniknie z miasta. Interfejs określa metody dla rozszerzeń elementów *węzła we/wy* (Gateway), *połączenia* (Link) oraz *pasa ruchu* (Lane). Warto zauważyć, że kontrakt zakłada jedynie wywołanie odpowiedniej metody na obiekcie implementującym interfejs obsługi zdarzenia. Ten fragment został zaprojektowany na podstawie wzorca projektowego Command.



Rysunek 4.9. Interfejs modułu monitoringu.

GatewayMonIface — interfejs rozszerzenia elementu *węzła wejścia/wyjścia*, definiuje sygnatury metod używanych do rejestracji obiektu obsługującego zdarzenia pojawienia się pojazdu oraz zniknięcia pojazdu, są to odpowiednio: void installEntranceSensor(CarEntranceHandler handler) oraz void installExitSensor(CarExitHandler handler). Interfejsy rozszerzenia elementu połączenia (LinkMonIface) oraz pasa ruchu (LaneMonIface) definiują sygnatury metod (void installInductionLoop(int line,

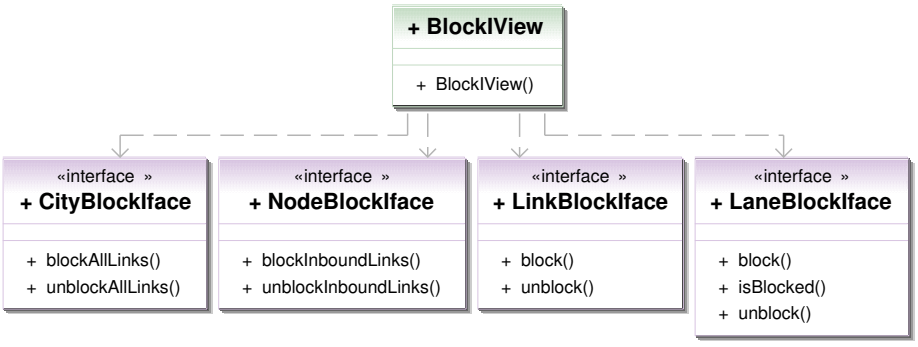
CarDriveHandler handler)), których zadaniem jest rejestracja obiektów obsługi zdarzenia przejazdu samochodu przez określony fragment *połączenia* lub *pasa*. Zdarzenia przejazdu powinno być generowane, gdy pojazd przekracza linię pomiędzy komórkami o numerach $line - 1$ i $line$[3]. Obiekt obsługi takiego zdarzenia to np. pętla indukcyjna. Pętla otrzymuje informację, że pojazd właśnie przejechał i pobiera dostępne informacje (zależnie od implementacji).

W ramach interfejsu modułu monitoringu stworzone zostały 3 interfejsy dodatkowe (CarEntranceHandler, CarExitHandler, CarDriveHandler). Te trzy interfejsy definiują sygnatury metod obsługi zdarzeń: CarDriveHandler.handleCarDrive(), CarEntranceHandler.handleCarEntrance(), CarExitHandler.handleCarExit().

Interfejs blokowania pasów

Interfejs zdefiniowany jest w pakiecie `pl.edu.agh.cs.kraksim.iface.block`

Interfejs ten pozwala na blokowanie poszczególnych pasów w mieście. Blokada pasa ruchu powoduje, że samochód nie może z niego wjechać do kończącego go węzła – jest to uogólnienie sterowania ruchem przez światła drogowe. W dalszej części dokumentacji czynność zmiany świateł drogowych przekłada się właśnie na blokowanie/odblokowywanie pasów.



Rysunek 4.10. Interfejs blokowania pasów.

Interfejs rozszerzenia elementu *miasto* — CityBlockIface, deklaruje sygnatury metod, których zadaniem jest blokowanie i odblokowywanie wszystkich pasów na wszystkich połączeniach: blockAllLinks(), unblockAllLinks().

[3] oczywiście tylko w przypadku mikro-symulacji używającej automatów komórkowych, w innym wypadku może to być odległość w metrach

Interfejs rozszerzenia elementu *węzeł* — NodeBlockIface, deklaruje sygnatury metod blokowania i odblokowywania pasów na wszystkich połączeniach wchodzących do tego węzła: blockInboundLinks(), unblockInboundLinks().

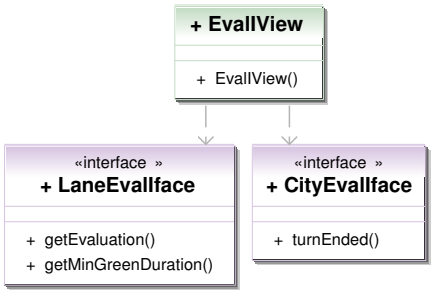
Interfejs rozszerzenia elementu *połączenie* — LinkBlockIface, deklaruje sygnatury metod blokowania i odblokowywania wszystkich pasów na danym połączeniu: block(), unblock().

Interfejs rozszerzenia elementu *pas* — LaneBlockIface, deklaruje sygnatury metod blokowania i odblokowywania pasu(block(), unblock()) oraz metodę sprawdzania stanu blokady pasa (isBlocked()).

Interfejs oceny korzyści z włączenia zielonego światła

Interfejs zdefiniowany jest w pakiecie `pl.edu.agh.cs.kraksim.iface.eval`

Interfejs ten dotyczy optymalizatora pracy świateł drogowych. Służy do oceny korzyści jaką sumarycznie odniosą samochody na danym pasie, gdy będzie się na nim paliło zielone światło. Polityka oceny powinna być spójna w ramach wszystkich pasów wchodzących do jednego skrzyżowania.



Rysunek 4.11. Interfejs oceny kosztu zmiany sygnalizacji.

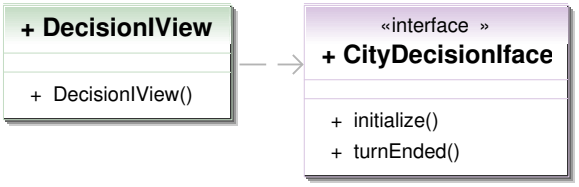
Interfejs rozszerzenia elementu *miasto* (CityEvalIface) deklaruje tylko jedną metodę — void turnEnded(). Implementacja tej metody powinna uruchamiać proces obliczania kosztów, służy ona tylko do informowania warstwy oceny, że symulacja tury już się zakończyła.

Implementacja interfejsu rozszerzenia elementu *pas* (LaneEvalIface) musi implementować metodę zwracającą wartość oceny — float getEvaluation() (jest to ocena sytuacji drogowej na danym pasie, oceny pasów wchodzących w skład skrzyżowania są porównywane i na tej podstawie można obliczyć koszt zmiany stanu sygnalizacji, najprostszą oceną może być liczba samochodów na pasie), oraz metodę zwracającą minimalny czas trwania następnej fazy zielonego światła w turach — int getMinGreenDuration().

Interfejs decyzyjny

Interfejs zdefiniowany jest w pakiecie `pl.edu.agh.cs.kraksim.iface.decision`

Interfejs ten dotyczy optymalizatora pracy świateł drogowych. Służy do informowania modułu realizującego sterowanie światłami o końcu tury (symulacji ruchu) i pozwala mu wtedy na wykonanie niezbędnych działań.



Rysunek 4.12. Interfejs decyzyjny.

Interfejs rozszerzenia elementu *miasto* — CityDecisionIface, definiuje sygnatury metod void initialize(), oraz void turnEnded(). Pierwsza ma być wywoływana podczas inicjalizacji systemu, a druga po każdej zakończonej turze symulacji. Interfejs modułu decyzyjnego nie oferuje żadnych informacji. Funkcjonalnością tego modułu ma być podjęcie i wykonanie decyzji, aktualny stan systemu można sprawdzić używając interfejsów: monitoringu, informacji o pojazdach oraz blokowania pasów.

4.2.3. Inne interfejsy

Oprócz modułów, w systemie mogą istnieć jeszcze inne komponenty, które mogą mieć charakter podmienialny (być ukryte za interfejsami).

Interfejsy są też wymagane wtedy, gdy moduły potrzebują uzyskać pewne informacje od systemu (sterownika/menadżera systemu). Przykładem takiej sytuacji jest interfejs zegara — pewien moduł może wymagać znajomości aktualnego czasu (numeru tury).

Interfejs zegara

Poniższy interfejs będzie implementował prawie zawsze sterownik symulacji:

```
public interface Clock {
    /* zwraca aktualny czas (numer tury) */
    public int getTurn();
}
```

Interfejs routera

Komponent do wyznaczania tras oferuje tylko jedną operację, co ukazuje poniższy interfejs pochodzący z pakietu `pl.edu.agh.cs.kraksim.routing`:

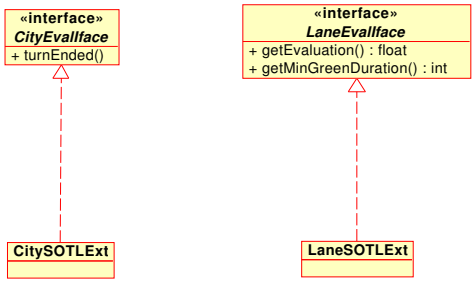
```
public interface Router {
    /* zwraca trasę pomiędzy parą węzłów */
    public Route getRoute(Gateway sourceGateway, Gateway destGateway);
}
```

Generacja ruchu

Obecnie generator ruchu jest integralną częścią symulatora i nie ma jasnego interfejsu. W przyszłej wersji systemu należałoby pokusić się o zdefiniowanie takiego i uniezależnienie symulatora od konkretnego generatora.

4.2.4. Jądro a moduły

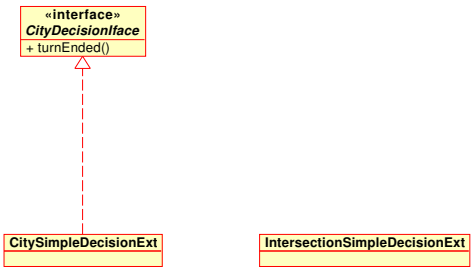
W tym rozdziale zawarto przykładowe diagramy, które powinny pomóc w zrozumieniu architektury systemu.



Rysunek 4.13. Przykład architektury - interfejs oceny, moduł SOTL

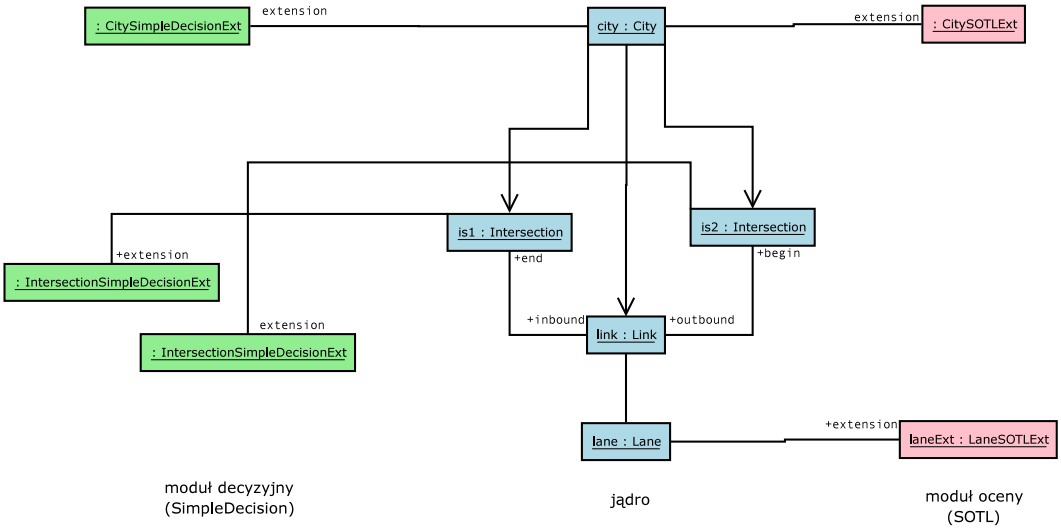
Pierwszy z nich (rys. 4.13) ukazuje interfejs oceny składający się z interfejsów dla poszczególnych typów elementów. Interfejs oceny definiuje operacje tylko dla dwóch typów elementów: miasta i pasa. Załóżmy, że istnieje moduł oceny (SOTL – opisany później) zawierający obiekty klas: CitySOTLExt oraz LaneSOTLExt (inne typy elementów niż miasto i pas nie są rozszerzane).

Drugi (rys. 4.14) ukazuje interfejs decyzyjny, który definiuje tylko jedną nową operację dla elementu typu *miasto*. Na diagramie można też zobaczyć przykład modułu decyzyjnego. Moduł ten dodaje rozszerzenia do dwóch typów elementów: *miasta* i *skrzyżowania*. W pierwszym przypadku jest to konieczne („coś" musi implementować operację turnEnded()). Rozszerzenie elementu typu *skrzyżowanie* nie ma żadnych publicznych operacji – to wewnętrzny element modułu. W naszej implementacji obiekt



Rysunek 4.14. Przykład architektury - interfejs decyzyjny, prosty moduł decyzyjny

IntersectionSimpleDecisionExt można odnieść do rzeczywistości – odpowiada on sterownikowi świateł na jednym skrzyżowaniu.

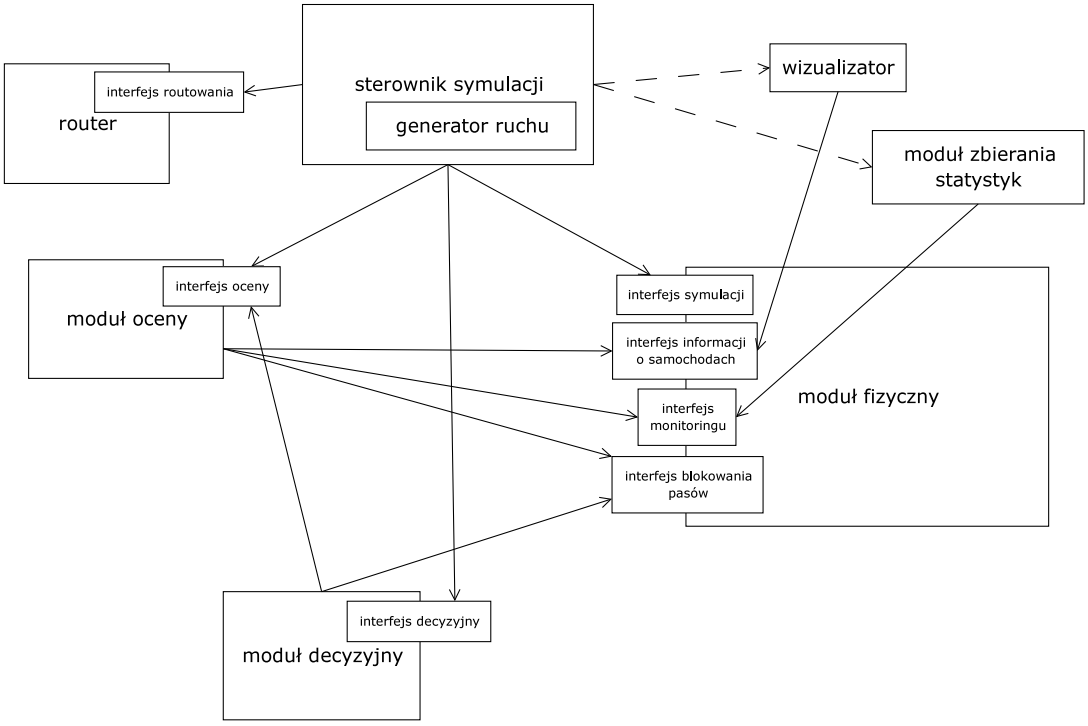


Rysunek 4.15. Przykład architektury - diagram obiektów

Trzeci (rys. 4.15) pokazuje jakie obiekty istnieją w trakcie działania systemu. Zakładamy, że miasto składa się z dwóch skrzyżowań, pomiędzy którymi poprowadzono (jednokierunkowe) połączenie składające się z jednego pasa.

4.3. System symulacji — części składowe

Poniższy opis, zilustrowany także na rysunku 4.16, nie narzuca sposobu implementacji modułu (w szczególności użytego algorytmu). Moduł musi tylko implementować wymienione przy nim interfejsy. Nie jest konieczne, aby wykorzystywał wszystkie dostarczane mu interfejsy.



Rysunek 4.16. Architektura symulatora. Dla zwiększenia przejrzystości rysunku nie umieściliśmy na nim jądra (każda składowa, której potrzebne są informacje na temat topologii miasta, z niego korzysta); strzałki narysowane linią ciągła wskazują na wykorzystanie jasno zdefiniowanych interfejsów modułów, strzałki linią przerywaną wskazują wykorzystanie modułów bez ogólnych definicji interfejsów

4.3.1. Jądro symulacji

Zawiera obiekty reprezentujące elementy sieci dróg (skrzyżowania, połączenia, itp.) oraz odzwierciedla powiązania między nimi. Udostępnia funkcjonalność pozwalającą jedynie na przeglądanie tych powiązań - odkrywanie topologii miasta.

Każdy moduł musi być przyporządkowany do pewnego jądra, przy czym moduły współpracujące ze sobą – do tego samego.

Technicznie, w symulatorze może istnieć wiele jąder z modułami – taka sytuacja odpowiada istnieniu wielu miast lub dzielnic. Może to być przydatne wtedy, kiedy model zostanie podzielony na wiele części w celu zrównoleglenia obliczeń. Dzielnice mogą zostać umieszczone na różnych węzłach obliczeniowych, a bramy wjazdowe/wyjazdy będą łączyły rozproszone elementy systemu. W takiej konfiguracji pojawiają się problemy synchronizacji, jednakże nie jest to tematem pracy.

4.3.2. Moduły rozszerzające

Na rysunku 4.16 (str. 70) umieszczone zostały wszystkie moduły rozszerzające wymagane do zbudowania symulatora ruchu. Moduły oznaczone są dużymi prostokątami. Interfejsy implementowane(dostarczane) przez moduł narysowano jako mniejsze prostokąty położone na krawędzi symbolu modułu. Natomiast interfejsy wykorzystywane przez moduł zaznaczone są strzałką narysowaną linią ciągłą, strzałka prowadzi od modułu do wykorzystywanego interfejsu (np.: *moduł decyzyjny* wykorzystuje *interfejs oceny* oraz *interfejs blokowania pasów*). Interfejsy zostały szczegółowo opisane w punkcie 4.2.2 na stronie 61. Konkretne implementacje modułów przedstawia rozdział 4.5 str. 77.

Krótki opis zadań podstawowych modułów:

Moduł fizyczny zajmuje się symulacją ruchu, udostępnianiem informacji o tej symulacji oraz umożliwia wpływ na nią poprzez sterowanie blokadą pasów.

Moduł oceny zajmuje się oceną korzyści z włączenia zielonego światła.

Moduł decyzyjny zajmuje się podejmowaniem i realizacją decyzji o zmianie świateł (blokowaniu/odblokowywaniu pasów).

Moduł wyznaczania tras zajmuje się wyznaczaniem tras przejazdu.

Moduł zbierania statystyk służy do zebrania statystyk z symulacji, aby mogły być później udostępnione użytkownikowi.

Wizualizator służy do graficznej wizualizacji przebiegu symulacji.

Sterownik systemu steruje przebiegiem całej symulacji.

Generator ruchu — tak naprawdę część sterownika systemu. Zajmuje się wygenerowaniem kierowców (wg konkretnej dla implementacji zasady/schematu) oraz nadzorowaniem rozpoczynania ich kolejnych podróży. Rozpoczęcie podróży polega na umieszczeniu samochodu z tym kierowcą w symulacji.

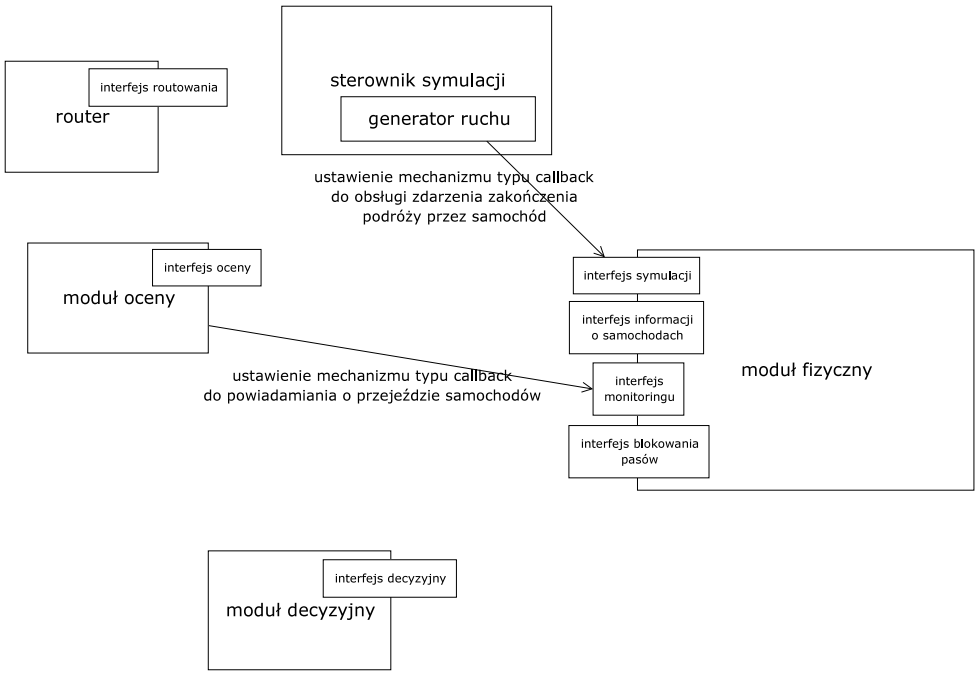
4.4. Dynamika systemu symulacji

Kolejne podpunkty przedstawiają etapy działania symulacji. Aby skupić uwagę na samej symulacji, z rozważań pominięto wizualizator i moduł zbierania statystyk.

Ze względu na czytelność, na rysunkach nie znalazło się jądro, które służy poszczególnym składnikom jedynie do zdobywania informacji na temat powiązań w sieci dróg. Jest to faktycznie istniejąca część systemu – ma reprezentację w pamięci komputera, ale statyczna – symulacja nie ma na jej stan żadnego wpływu (cała symulacja odbywa się „poza jądrem").

4.4.1. Inicjalizacja części składowych

Każda część dokonuje swojej inicjalizacji podczas powstawania. W tym podpunkcie wymieniono tylko te nietrywialne jej aspekty, które wymagające współdziałania części(rys. 4.17).



Rysunek 4.17. Współpraca części: Inicjalizacja części składowych

Inicjalizacja modułu oceny. Konkretny moduł oceny może (ale nie musi, jeśli to nie jest mu do niczego potrzebne) zarejestrować podczas inicjalizacji metody obsługi zdarzeń przejazdu samochodów w miejscach przez niego określonych. Dla przykładu: zwiększanie licznika, gdy samochód przejeżdża przez początek połączenia (innymi słowy wjeżdża na połączenie) oraz zmniejszanie licznika, gdy przejeżdża przez jego koniec (zjeżdża z połączenia) może posłużyć do wyliczania liczby samochodów aktualnie znajdujących się na połączeniu.

Metody do wykorzystania:

- `LaneMonIface.installInductionLoop()`,
- `LinkMonIface.installInductionLoops()` — metoda uogólniona dla połączenia (Link), deleguje wywołanie do metod na odpowiednich instancjach obiektu Lane.

Inicjalizacja generatora ruchu. Generator ruchu musi reagować na koniec podróży kierowcy, ponieważ dopiero wtedy może wyznaczyć czas wyruszenia w kolejną

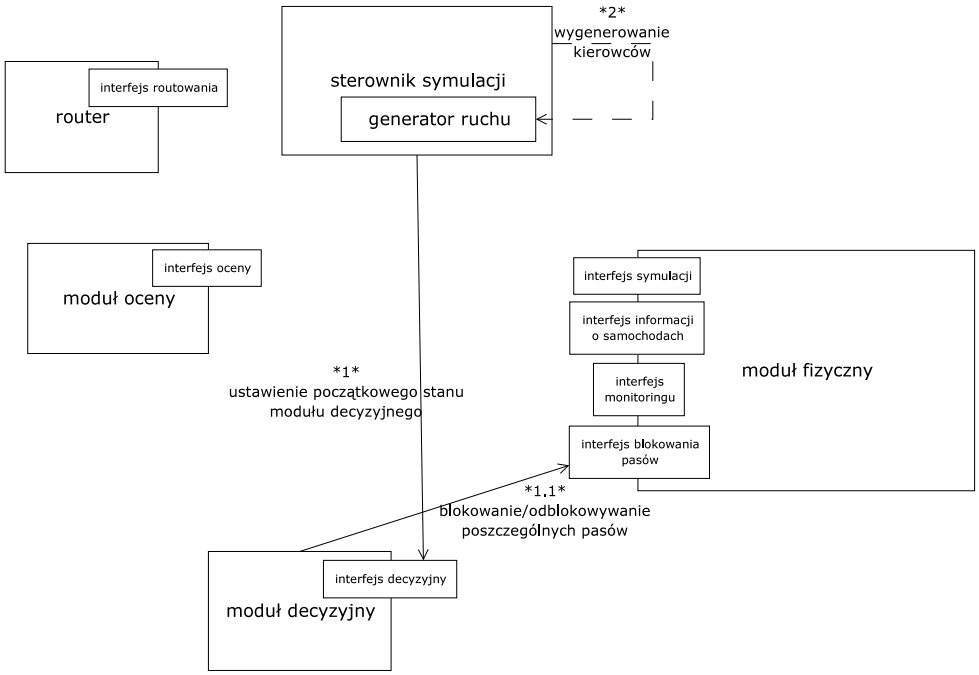
(o ile jest to w jego schemacie podróżowania). W związku z tym generator ruchu rejestruje swoją obsługę zdarzenia końca podróży.

Metody do wykorzystania:

- `CitySimIface.setCommonTravelEndHandler()` — ustawia, wspólną dla wszystkich węzłów wyjazdowych, klasę dla obsługi zdarzenia zakończenia podróży,
- `GatewaySimIface.setTravelEndHandler()` — ustawia sposób obsługi zdarzenia zakończenia podróży dla konkretnego węzła.

4.4.2. Start symulacji

Podczas startu symulacji dokonuje się kilku czynności przygotowywujących (rys. 4.18).



Rysunek 4.18. Współpraca części: Start symulacji

Ustawienie początkowego stanu modułu decyzyjnego. Sterownik symulacji nakazuje ustawienie początkowego stanu modułu decyzyjnego wywołując metodę `CityDecisionIface.initialize()`. Ta czynność mogłaby być dokonywana w trakcie inicjalizacji systemu, została jednak wydzielona, aby umożliwić późniejsze zresetowanie stanu modułu decyzyjnego.

W trakcie ustawienia stanu początkowego moduł decyzyjny blokuje/odblokowuje poszczególne pasy.

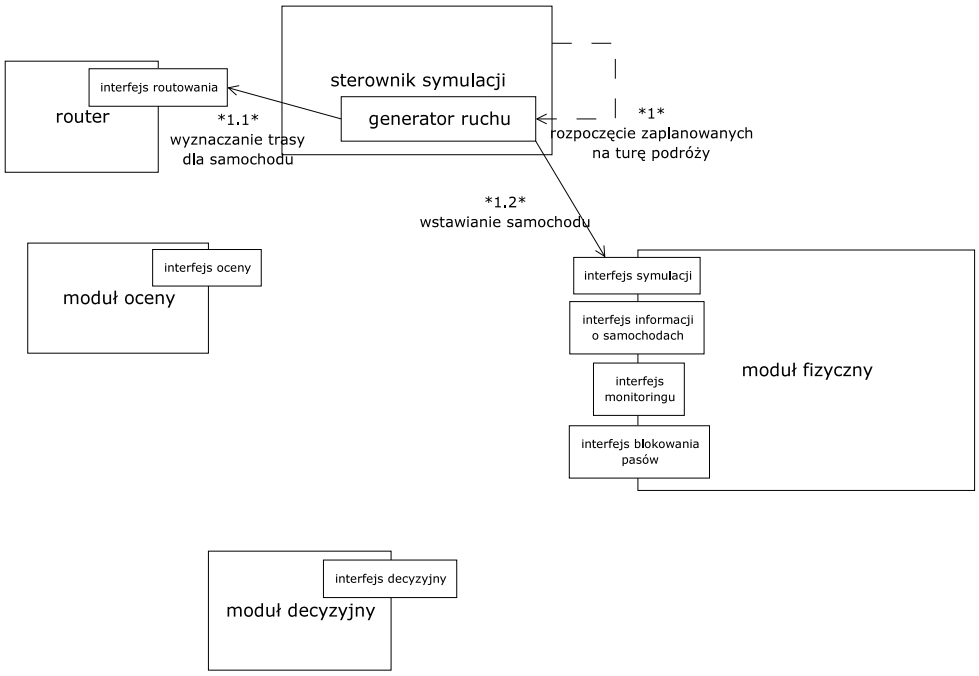
Metody do wykorzystania: wszystkie z interfejsu blokowania pasów.

Wygenerowanie kierowców. Sterownik symulacji nakazuje wygenerowanie kierowców. Generator ruchu czyni to zgodnie ze swoją polityką.

4.4.3. Pętla symulacyjna

Ilustracja pętli symulacji, ze względu na czytelność, została podzielona na kilka rysunków. Każdy z nich przedstawia inną czynność. Numeracja sekwencji czynności jest jednak wspólna dla wszystkich rysunków.

Rozpoczęcie zaplanowanych na turę podróży to pierwszy krok wykonywany w pętli symulacji (rys. 4.19).

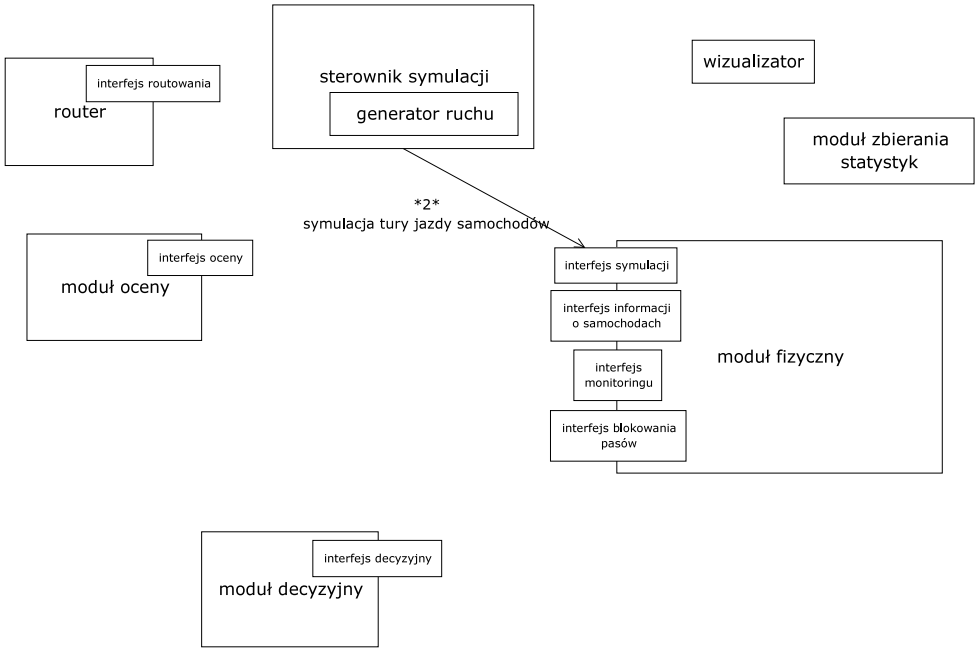


Rysunek 4.19. Współpraca części: Pętla symulacyjna - rozpoczęcie zaplanowanych podróży

Sterownik symulacji nakazuje generatorowi ruchu rozpoczęcie przeznaczonych na tą turę podróży. Generator ruchu przed umieszczeniem każdego samochodu w symulacji wyznacza jego trasę. Wykorzystuje metodę: `Router.getRoute()`. Następnie samochód jest umieszczany w symulacji z wykorzystaniem metody `CitySimIface.insertTravel()`.

Symulacja tury jazdy samochodów to etap, w którym pojazdy przemieszczają się w modelu miasta (rys. 4.20).

Sterownik symulacji nakazuje modułowi fizycznemu dokonać symulacji jednej tury poprzez wywołanie metody `CitySimIface.simulateTurn()`. Metoda `simulateTurn` przekazuje sterowanie do konkretnych elementów modelu miasta: połączeń oraz pasów.



Rysunek 4.20. Współpraca części: Pętla symulacyjna - symulacja tury jazdy samochodów

Uaktualnienie oceny pasów. Sterownik symulacji informuje moduł oceny o zakończeniu tury i nakazuje mu uaktualnić oceny pasów, w tym celu wywołuje metodę: `CityEvalIface.turnEnded()` (rys. 4.21). W trakcie dokonywania zażądanych obliczeń, moduł oceny może skorzystać z informacji o samochodach (gdzie aktualnie są i co zrobiły w ostatniej turze) oraz ze stanu blokady pasów w tej turze.

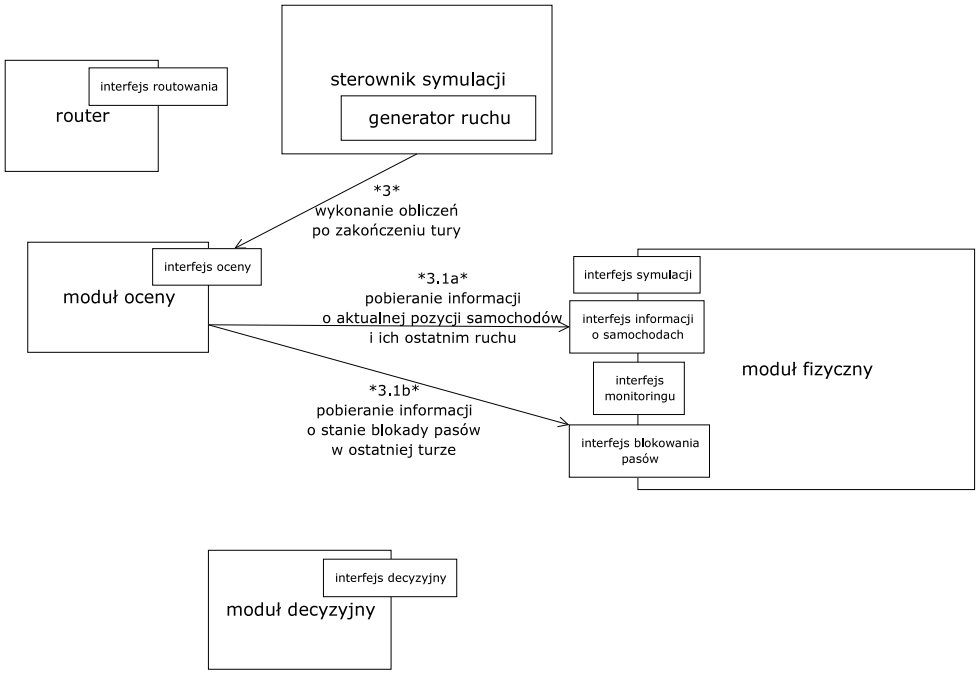
Metody do wykorzystania:

- wszystkie z interfejsu informajci o samochodach,
- `LaneBlockIface.isBlocked()`.

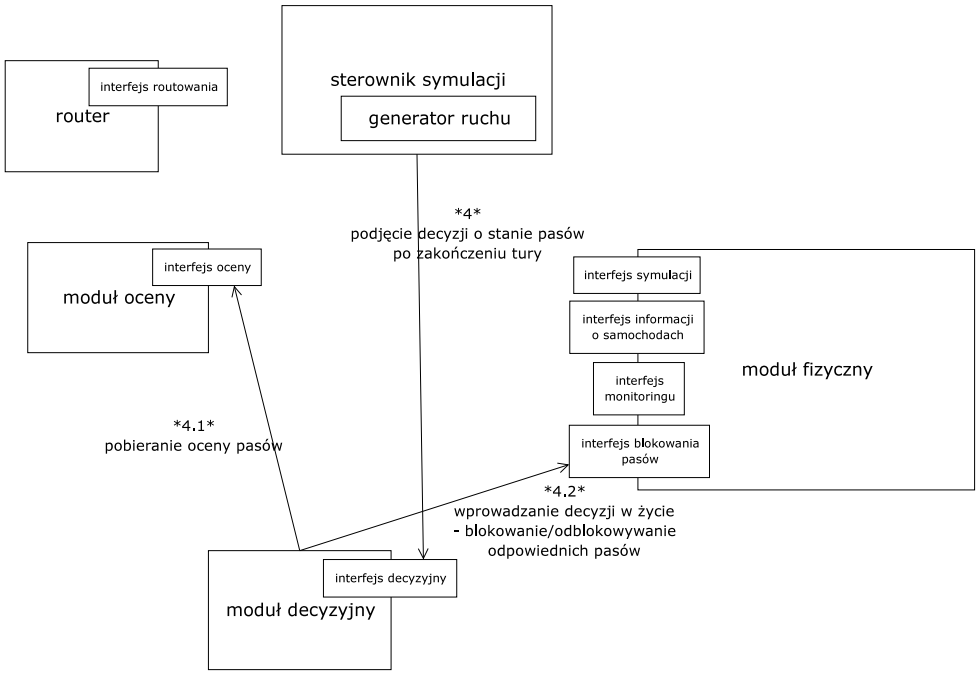
Podjęcie decyzji o stanie pasów kończy pętlę symulacyjną (rys. 4.22). Sterownik symulacji informuje moduł decyzyjny o zakończeniu tury i nakazuje mu podjąć oraz wprowadzić w życie decyzje – wywołuje metodę: `DecisionEvalIface.turnEnded()`. Moduł decyzyjny pobiera ocenę pasów korzystając z metod:

- `LaneEvalIface.getEvaluation()` — wartość funkcji oceny dla tego pasa ruchu,
- `LaneEvalIface.getMinGreenDuration()` — minimalny czas trwania sygnału zielonego, wartość ma zastosowanie w sygnalizacji akomodacyjnej oraz acyklicznej.

Następnie dokonuje obliczeń zgodnie ze swoją polityką i wywołuje odpowiednie metody interfejsu blokowania pasów.



Rysunek 4.21. Współpraca części: Pętla symulacyjna - uaktualnienie oceny pasów



Rysunek 4.22. Współpraca części: Pętla symulacyjna - podjęcie decyzji o stanie pasów

4.5. Konkretne moduły

W tym punkcie przedstawiono działanie konkretnych modułów użytych w symulatorze/optymalizatorze. W szczególności opisujemy użyte w modułach algorytmy.

4.5.1. Moduł fizyczny

Implementacja modułu fizycznego znajduje się w pakiecie `pl.agh.edu.cs.kraksim`

Działanie modułu symulacyjnego jest oparte na algorytmie NaSch[17]. Algorytmy symulacji ruchu na wprost oraz zachowania na skrzyżowaniu zostały zrealizowane zgodnie z koncepcją przedstawioną w rozdziale 3.4 na stronie 44.

Warto przypomnieć, że jednostka długości to 7,5 metra, jedna tura odpowiada jednej sekundzie. Zatem jednostka odległości na turę to $27\frac{km}{h}$. Domyślnie maksymalna prędkość to 2 jednostki, jednak dla każdego połączenia można ustalić inne wartości w opisie topologii sieci.

Implementacja algorytmu Nagla-Schreckenberga jest prosta i bezpośrednia. Kolejne bloki kodu odpowiadają krokom algorytmu. W systemie Kraksim ten algorytm zaimplementowany jest na poziomie pasa ruchu (LaneRealExt, fragment kodu pokazuje listing 4.1), w module nazwanym „fizycznym". Fizyczny dlatego, że odpowiada za symulację zachowań ze świata rzeczywistego.

W powyższym fragmencie kodu, kolejne kroki algorytmu są opisane odpowiednimi komentarzami. Zmienne użyte w tym kodzie oznaczają odpowiednio: *cit* to iterator po kolekcji pojazdów na pasie ruchu, *car* to aktualnie przetwarzany pojazd, *nextCar* to pojazd następny (może być *null*, gdy *car* jest ostatnim pojazdem). W czwartym kroku algorytmu (od linii 365) samochód wykonuje odpowiednią akcję. W zależności od warunków może to być postój, ruch do przodu, przejazd przez skrzyżowanie albo wyjazd przez węzeł wylotowy. Warunki dla ruchu pojazdu sprawdzane są w metodzie *driveCar* (podkreślony identyfikator w listingu 4.1).

W tym fragmencie kodu źródłowego (listing 4.1) można zauważyć stałe: *speedLimit* i *decelProb*, które oznaczają odpowiednio: lokalny dla pasa ruchu limit prędkości oraz prawdopodobieństwo zmniejszenia prędkości (wartość globalna dla algorytmu Na-Sch, a wiec również dla całej symulacji).

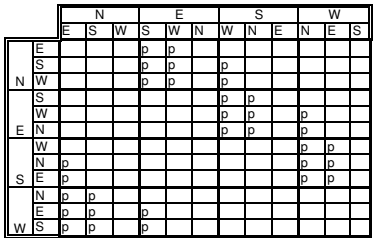
Skrzyżowania

Implementacja reguł przejazdu przez skrzyżowanie jest bardzo ważnym elementem symulacji. Na skrzyżowaniu najważniejsze są reguły pierwszeństwa. Te zaprojekto-

Kod źródłowy 4.1 Fragment metody LaneRealExt.simulateTurn() — algorytm Na-Sch.

```
do {
    nextCar = cit.hasNext() ? cit.next() : null;
    //remember starting point
345 car.setBeforeLane( lane );
    car.setBeforePos( car.pos );
    // Nagel-Schreckenberg
    // 1. Init velocity variable
    int v = 0;
350
    // 2. Acceleration
    if ( car.velocity < speedLimit ) {
      v = car.velocity + 1;
    } else {
355   v = car.velocity;
    }

    // 3. Deceleration
    if ( params.rg.nextFloat() < params.decelProb ) {
360   v--;
    }

    // 4. Drive (Move the car)
    int freePos = Integer.MAX_VALUE;
365 if ( nextCar != null ) {
      freePos = nextCar.pos - 1;
    }

    boolean stay = driveCar( car, car.pos, freePos, v, 0, ilp, false );
370 if ( !stay ) {
      if ( nextCar != null ) cit.previous();
      cit.previous();
      cit.remove();
      if ( nextCar != null ) cit.next();
375 }

    // remember this car as next (we are going backwards)
    car = nextCar;
} while ( car != null );
```

wano jako macierz pierwszeństwa przejazdu. Przykładowe skrzyżowanie przedstawia rys. 4.23.



Rysunek 4.23. Reguły pierwszeństwa na skrzyżowaniu. Kolumny oznaczają rozważane akcje, a w wierszach są akcje, które mogą być nadrzędne. N, E, S, W — kierunki północ, wschód, południe, zachód. *p* — oznacza że rozważana akcja jest podrzędna wobec innej (np.: akcja E-S jest podrzędna wobec N-E, N-S, N-W, W-E i W-S, oznacza to że kierowca chcący skręcić z kierunku E w drogę S, musi ustąpić pierwszeństwa tym wymienionym wcześniej).

Ponieważ jest to macierz rzadka, najlepiej jest ją zaimplementować jako listę list. Dla skrzyżowania zdefiniowana jest lista możliwych do wykonania akcji, a akcja jest określana przez pas źródłowy (z którego pojazd rusza) i pas docelowy (na który samochód wjeżdża) oraz listę pasów, z których akcje mają wyższy priorytet niż ta akcja. Sprawdzanie reguły dla konkretnego pojazdu polega na przeszukaniu listy pasów nadrzędnych dla akcji, którą kierowca ma zamiar wykonać oraz sprawdzenie czy z tych kierunków nie nadjeżdża inny pojazd (zestaw reguł systemu TRANSIMS opisano przy okazji omawiania koncepcji modelu symulacji 3.4 na stronie 44). Implementacja pokazana jest na fragmencie kodu źródłowego 4.2 na stronie 80 . Ten fragment kodu wywoływany jest na końcu pętli symulacyjnej jako fragment 4-tego punktu algorytmu Na-Sch.

Obsługa świateł na skrzyżowaniu jest bardzo prosta. Zielone światło oznacza możliwość przejazdu, czyli z perspektywy pasa ruchu sytuacja jest identyczna jak przy braku sygnalizacji. Sygnał czerwony oznacza blokadę, wtedy pojazdy mogą poruszać się po pasie, ale gdy dojadą do końca nie mogą go opuścić (warunek sprawdzany jest w metodzie *driveCar* w klasie *LaneRealExt*). Światło żółte w przypadku tego systemu oznacza to samo co światło czerwone, z tym że umożliwia przejechanie pojazdom, które nie mają możliwości wyhamowania przed skrzyżowaniem[4].

[4] droga hamowania liczona jest jako ilość komórek odpowiadająca aktualnej wartości prędkości pojazdu, jest to więc pewna wartość idealna podobno do wartości rzeczywistej

Kod źródłowy 4.2 Fragment metody LaneRealExt.handleCarAction() — obsługa skrzyżowania.

```
// pobierz liste pasow nadrzednych dla danej akcji
Lane[] pl = action.getPriorLanes();

for (int i~= 0; i~< pl.length; i++) {
  // sprawdz czy z~tego kierunku nadjezdza inny pojazd
  if ( realView.ext( pl[i] ).carApproaching ) {

    // sprawdz czy wystepuje zakleszczenie
    if ( checkDeadlock( action.getSource(), pl[i] ) ) {
      deadLockRecovery();
    }

    // pojazd nie moze przejechac przez skrzyzowanie, zwroc false
    return false;
  }
}

LinkRealExt targetLink = realView.ext( action.getTarget() );
// pojazd moze przejezdzac, ale sprawdz czy na pasie docelowym jest
// miejsce i~jezeli tak, to przesun tam pojazd
return targetLink.enterCar( car, stepsMax, stepsDone );
```

Rozwiązywanie problemów na skrzyżowaniu

Na skrzyżowaniu, na którym panuje reguła prawej dłoni, może dojść do zakleszczenia. W celu rozwiązania problemu zakleszczenia jeden z kierowców musi ustąpić (tzn. oddać pierwszeństwo), wtedy kolejny kierowca, który był przez niego blokowany może ruszyć i tym samym rozwiązać problem.

W celu rozwiązania problemu należy go wykryć. Aby wykryć zakleszczenie zaimplementowano prosty algorytm wykrywania cyklu w grafie. Jest on uruchamiany, gdy pojazd chce przejechać przez skrzyżowanie (co widać we fragmencie kodu 4.2). Po wykryciu zakleszczenia losowany jest jeden pojazd — ofiara (nazwijmy go A) — i oznaczany jako czekający. Gdy pojazd „czeka”, nie jest brany pod uwagę przez innych kierowców. W następnej turze samochód, który zatrzymał się na skrzyżowaniu ponieważ musiał ustąpić pierwszeństwa pojazdowi A, nie jest już przez A blokowany i może jechać.

4.5.2. Moduł decyzyjny

Działanie modułu decyzyjnego jest bardzo proste. Dla każdego skrzyżowania światło zielone jest zapalane na tym pasie, dla którego moduł oceny zwróci największą wartość. Światło pali się co najmniej przez tyle tur, ile wskaże moduł oceny, ale może dłużej, jeśli po tym czasie dla żadnego innego pasa wartość funkcji oceny nie przekroczy 0.

Konfigurowalne parametry modułu

Na etapie uruchamiania można skonfigurować ile tur ma trwać stan przejściowy pomiędzy światłami (*transitionDuration*), co odpowiada światłu żółtemu w rzeczywistości.

4.5.3. Moduł oceny SOTL - Self Organizing Traffic Lights

Zasadę działania tej metody sterowania światłami opisano już w rozdziale teoretycznym oraz dokładniej w koncepcji wykonania projektu. Tutaj zostanie opisany sposób realizacji modułu, metody dostarczania danych o stanie ruchu drogowego oraz konfiguracja.

SOTL jest autonomiczną metodą sterowania światłami, oznacza to, że poszczególne sygnalizatory nie komunikują się ze sobą, nie ma również centralnego sterowania. Ocena jest dokonywana na podstawie lokalnej sytuacji.

SOTL wymaga do poprawnego działania zegara, dlatego korzysta z interfejsu *Clock*, który jest dostarczany przez moduł symulacji. Ta metoda działa na podstawie danych o liczbie pojazdów na pasach dróg wchodzących do danego skrzyżowania. Ponieważ każdy pas może być oceniany niezależnie, wystarczyło zaimplementować rozszerzenie pasa ruchu (LaneSOTLExt), widok modułu i klasy pomocnicze. Odpowiednie metody wykonania oceny wywoływane są z rozszerzenia miasta dla tego modułu — CitySOTLExt.

Aby ocenić liczbę pojazdów w strefie przed skrzyżowaniem, stosuje się dwie pętle zliczające[5]. Jedna pętla na początku strefy zlicza pojazdy wjeżdżające i na tej podstawie zwiększany jest licznik pojazdów. Druga pętla przy samym skrzyżowaniu zlicza pojazdy wyjeżdżające, a licznik jest zmniejszany. Dzięki temu sterownik sygnalizacji ma dostęp do bieżących danych o ilości pojazdów w strefie.

[5] w zasadzie mogą być to dowolne detektory zdolne zliczać przejeżdżające pojazdy

Konfiguracja

Parametry konfiguracyjne opisano przy okazji prezentacji koncepcji. Można je ustawić w pliku konfiguracyjnym `config.properties` jako wartość klucza algorithm. Parametry dla tego algorytmu podawane są po dwukropku jako pary klucz–wartość (w tym przypadku `sotl:klucz=wartość, klucz2=wartość2`). Domyślnie jest to: `algorithm = sotl:zone=18`.

4.5.4. Moduł oceny RL - Reinforcement Learning

RL podobnie jak SOTL jest metodą autonomicznego sterowania sygnalizacją świetlną. Jednak tutaj nie wystarczy informacja o ilości pojazdów na pasie. RL do poprawnego działania potrzebuje oszacowania pozycji pojazdu na pasie, aby móc przyporządkować odpowiednie wartości fragmentom pasa ruchu. Dopiero na podstawie wartości Qb oraz Qu[6] dla wszystkich komórek na pasie, wyliczana jest ocena dla tego pasa.

Aby oszacować pozycję pojazdu na pasie, można w systemie rzeczywistym zamontować pętle zliczające na początku i na końcu pasa ruchu, a dla tego pasa ruchu prowadzić symulację na podstawie pewnych danych statycznych o ruchu pojazdów (np.: symulacja met. Na-Sch). Jeżeli zamiast pętli indukcyjnych zastosuje się wideodetektory można uzyskać dokładniejsze informacje o prędkości pojazdów.

W tym celu należałoby stworzyć drugą warstwę symulacji, traktując model podstawowy jako rzeczywistość. Jednak symulacja rzeczywistości Kraksim już jest „wirtualna", a dla celów badawczych ważniejsze było przetestowanie implementacji tej metody RL, niż systemu szacowania pozycji pojazdu na podstawie danych z pętli zliczających. Dlatego postanowiono wykorzystać bezpośrednio dane dostępne z modułu symulacji [7].

Moduł oceny RL, podobnie jak SOTl, implementuje tylko rozszerzenie pasa ruchu, rozszerzenie miasta (LaneRLExt, CityRLExt) oraz klasy pomocnicze.

Ponieważ wydajności symulacji z użyciem modułu oceny RL, który dokonuje oceny kosztów na podstawie pozycji pojazdów na całej długości pasa, była bardzo niska, wprowadzono tutaj pojecie strefy podobnie jak dla metody SOTL. To pojęcie strefy ogranicza działanie metody (prawdopodobnie obniża również jakość oceny) do pew-

[6] patrz rozdziały 2.4.4 oraz 3.5.3 dla szczegółowego opisu zagadnień teoretycznych oraz koncepcji wykonania

[7] potraktowano tym samym symulację rzeczywistości jako system oszacowania pozycji pojazdu, warto zauważyć, że gyby system był podłączony do detektorów miejskich to danych o pozycji pojazdu nie dałoby się uzyskać

nego wycinka drogi przed skrzyżowaniem. Jednak warto zauważyć, że w rzeczywistym systemie trudno jest prowadzić ocenę pozycji i prędkości pojazdów na drodze o długości kilku kilometrów (gdzie po drodze jest wiele elementów zaburzających przejazd jak wjazdy i wyjazdy z posesji prywatnych lub małe parkingi), a największy wpływ na symulacje ma i tak odcinek pasa znajdujący się bezpośrednio przed skrzyżowaniem.

Konfiguracja

Możliwa konfiguracja została omówiona w części teoretycznej i wszystkie te elementy konfiguracji zostały zaimplementowane. Konfiguracja jest możliwa z poziomu pliku konfiguracji użytkownika `config.properties` przy użyciu klucza `algorithm`. Wartości parametrów algorytmu podawane są jako pary klucz–wartość po dwukropku po nazwie metody (w tym przypadku `rl:klucz=wartość, klucz2=wartość2`).

Domyślnie jest to: `algorithm=rl`.

4.5.5. OptAPO

Algorytm OptAPO[8] należy do rodziny metod koordynowanej mediacji[9], które służą do rozproszonego rozwiązywania problemów spełniania ograniczeń (DCSP — Distributed Constraint Satisfaction Problem). Do jego prawidłowego działania potrzebne są podobne informacje jak w przypadku algorytmu SOTL, to znaczy ilość pojazdów dojeżdżających do skrzyżowania. Zegar nie jest potrzebny.

OptAPO jest metodą opartą na mediacji pomiędzy agentami, dlatego należy użyć platformy agentowej. Na potrzeby tego rozwiązania zaprojektowano i wykonano uproszczony system agentowy (turowy), zorientowany na rozwiązywanie wyboru kierunku synchronizacji skrzyżowań za pomocą metody OptAPO.

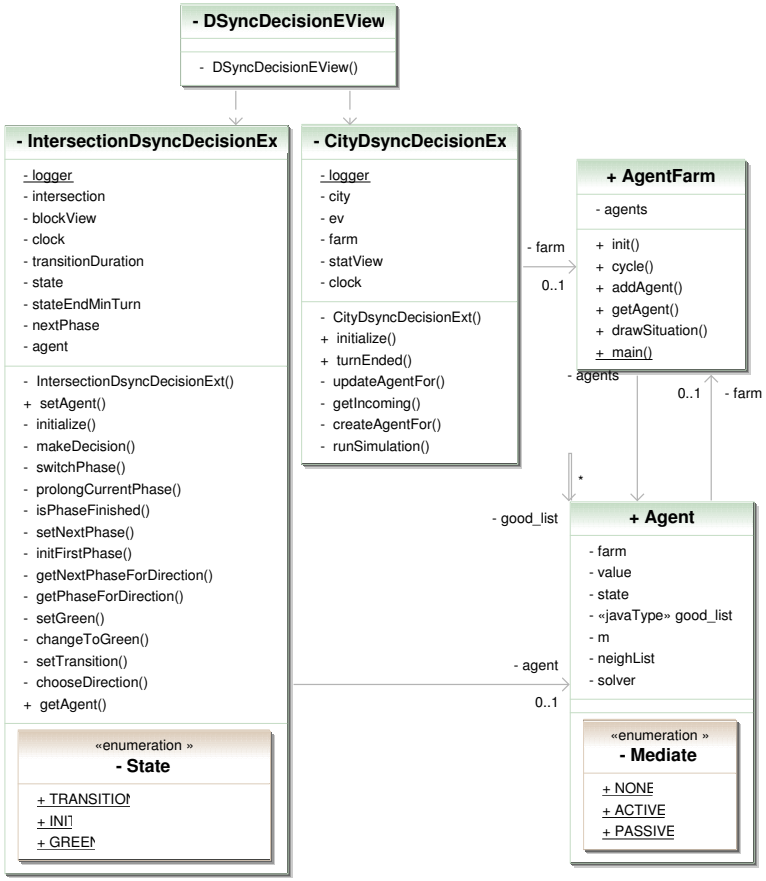
To rozwiązanie zostało zaimplementowane na poziomie modułu decyzyjnego. Ponieważ nie ma innych rozwiązań używających modułu decyzyjnego wybierającego kierunki synchronizacji, nie rozdzielono modułu oceny od modułu decyzyjnego. Moduł decyzyjny DSyncDecision[10] agreguje logikę oceny, która jest oparta na metodzie OptAPO.

Sam moduł rozszerza tylko elementy miasta i skrzyżowania (odpowiednio CityDSyncDecisionExt oraz IntersectionDSyncDecisionExt). Jak ilustruje diagram na rysunku 4.24, każdemu rozszerzeniu skrzyżowania odpowiada jeden agent optymalizu-

[8] Optimal Asynchronous Partial Overlay — szczegółowo opisany już w rozdziale 2.4.5

[9] ang.: CM — Cooperative Mediation

[10] DSyncDecision — nazwa ma oznaczać Direction Synchronization Decision, czyli moduł decyzyjny synchronizacji kierunku



Rysunek 4.24. Moduł koordynacji pracy skrzyżowań (szczegóły implementacji agenta zostały ukryte dla zwiększenia przejrzystości).

jący kierunek synchronizacji. Agenty działają na prostej platformie agentowej (własna implementacja), która jest przypisana do rozszerzenia miasta. Rozszerzenie miasta jest odpowiedzialne za inicjalizacje agentów i ich skojarzenie z odpowiednimi skrzyżowaniami.

Moduł zaimplementowany jest w ten sposób, że co pewien ustalony okres uruchamiany jest proces optymalizacji. Proces ten jest zbieżny do optymalnego rozwiązania. Po kilku kolejnych przebiegach mediacji system osiąga aktualnie optymalny stan. Po kilku początkowych mediacjach poprawiane są problemy lokalne, a następnie, gdy każdy agent zwiększa swoją listę znanych agentów, poprawiane są rozwiązania o coraz większym zasięgu. Ponieważ w przypadku optymalizacji ciągłej bardzo ważna jest szybkość reakcji, ograniczono czas trwania każdej sesji optymalizacji do liczby tur,

która jest iloczynem ilości agentów–skrzyżowań przez 10. Jest to liczba tur uzyskana empirycznie.

Po skończonej sesji optymalizacji zmienne oznaczające kierunek synchronizacji aktualizowane są w każdym obiekcie skrzyżowania.

4.5.6. Generator ruchu

Generator nie jest oddzielnym komponentem, ale jest to na tyle ważny obszar symulatora/optymalizatora, że zostanie tu pokrótce opisany.

Definiuje się tak zwane *schematy podróżowania*. Jeden schemat dotyczy pewnej liczby samochodów i wyznacza kolejne węzły wlotowe, do których kierował się będzie samochód (każda trasa wiedzie od jednego węzła wlotowego do drugiego). Z każdym węzłem schematu, z wyjątkiem ostatniego, jest związany czas wyruszenia z węzła. Czas wyruszenia jest zadany rozkładem losowym.

Działanie generatora ruchu sprowadza się do stworzenia samochodów dla wszystkich schematów podróżowania (w odpowiedniej krotności), a następnie umieszczania ich w modelu w turach wylosowanych zgodnie z podanymi przez użytkownika rozkładami.

Konfigurowalne parametry generatora ruchu

Możliwe jest określenie zarodka używanego generatora liczb losowych *genSeed*.

4.5.7. Router

Do wyznaczania tras przejazdu używany jest algorytm Dijkstry - samochód jedzie od jednego węzła wlotowego do innego po najkrótszej (względem długości krawędzi) trasie. Dla danej pary węzłów źródło–cel zawsze zostanie wybrana ta sama trasa.

Generator ma również możliwość wyznaczania tras na podstawie tablicy średnich czasów przejazdu, która jest mu dostarczona w postaci kolekcji par: droga – czas. Ta funkcjonalność jest wykorzystywana przez pojazdy szukające objazdów, gdy działa system informacji o zatłoczeniach. Tablica czasów przejazdu budowana jest przez system informacji o zatłoczeniach na podstawie średnich danych z ustalonego okresu czasu np.: z ostatnich 10-ciu minut. Czasy przejazdu szacowane są na podstawie danych o średniej prędkości i długości drogi (iloczyn tych dwóch wartości).

Jeżeli czas gromadzenia danych ustalony jest na 10 minut, to na starcie budowana jest tablica, w której czasy przejazdu obliczone są jako iloczyn długości drogi i maksymalnej dozwolonej prędkości. Przez 10 minut gromadzone są dane, które posłużą do obliczenia wartości średnich. Po upłynięciu 10-ciu minut, na podstawie zgromadzo-

nych danych budowana jest tablica średnich czasów przejazdu, a dane statystyczne są zerowane (rozpoczyna się kolejny dziesięciominutowy okres pomiarowy). Ta tablica będzie używana przez generator tras, aż do momentu uaktualnienia wartości średnich (za 10 minut).

Jeżeli parametr określający prawdopodobieństwo zmiany trasy przez kierowcę jest ustawiony, to kierowca przed każdym skrzyżowaniem sprawdza czy powinien szukać nowej trasy. Jeżeli wynik testu jest pozytywny, kierowca próbuje wyznaczyć nową trasę. Nowa trasa wyznaczana jest również algorytmem Dijkstry, jednak w tym przypadku po uwagę brane są średnie czasy przejazdu, a nie długość drogi. Jest to jedyne kryterium i na dodatek bardzo proste, ale umożliwia wybranie szybszej ścieżki. W celu poprawienia realizmu symulacji, należałoby zaimplementować wielokryterialny algorytm poszukiwania optymalnej ścieżki z uwzględnieniem wag dla różnych kryteriów.

Rozdział 5

Eksperymenty

System symulacji ruchu drogowego prezentowany w tej pracy został poddany serii testów. Starano się tak dobrać eksperymenty, aby pokazać i zweryfikować poziom realizacji celów pracy. Eksperymenty umożliwiają ocenę poprawności zachowania modelu ruchu drogowego (zgodnie z zachowaniem obserwowanym w sytuacjach rzeczywistych), porównanie właściwości różnych topologii oraz porównanie wybranych metod optymalizacji.

Pierwszy eksperyment sprawdza charakterystyki ruchu dla pięciu wariantów modelu skrzyżowania typu T, oraz dla czterech wariantów skrzyżowania dwóch dróg (typu X).

Drugi eksperyment sprawdza zachowanie systemu na uproszczonym modelu miasta. Badany jest wpływ natężenia ruchu na inne charakterystyki symulacji. W drugiej części tego eksperymentu przeprowadzony jest test na modelu miasta z dobudowaną nową drogą. Sprawdza się jaki wpływ na charakterystyki ruchu ma dodanie nowego połączenia do istniejącej topologii.

Metody sterowania sygnalizacją świetlną porównane są w trzecim eksperymencie. Jako model sieci drogowej wybrano układ 9-ciu skrzyżowań. W tym przypadku ocenia się możliwość optymalizacji ruchu za pomocą różnych metod sterowania sygnalizacją świetlną.

Ostatni — czwarty — eksperyment ma na celu zbadanie wpływu systemu informacji o korkach na sytuację drogową w mieście przy założeniu, że część kierowców w wybierze objazdy w odpowiedzi na otrzymane informacje o korku. Testy przeprowadzono na modelu kratowym oraz na uproszczonym modelu miasta. Ten eksperyment ma odpowiedzieć na pytanie, czy odpowiednia informacja o ruchu drogowym może usprawnić ruch w mieście.

Wszystkie eksperymenty opisane są zgodnie ze schematem: cel eksperymentu, model sieci drogowej, schemat ruchu, uzyskane wyniki, analiza wyników oraz wnioski. Ten sposób pozwala na uporządkowanie informacji oraz przedstawienie ich w przejrzystej formie.

5.1. Topologie skrzyżowań

Eksperyment polega na sprawdzeniu wpływu zmian w budowie skrzyżowania na jego przepustowość. Test pomaga również zweryfikować zachowanie projektowanego systemu symulacji. Testom poddane zostały dwa skrzyżowania. Jedno skrzyżowanie typu T i jedno skrzyżowanie typu X (z drogą główną). Wybrano te dwa typy skrzyżowań, ponieważ występują one często w rzeczywistej sieci drogowej.

5.1.1. Model skrzyżowania typu T

Test jest przeprowadzany na skrzyżowaniu typu T. Można je porównać do zjazdu z obwodnicy miasta.

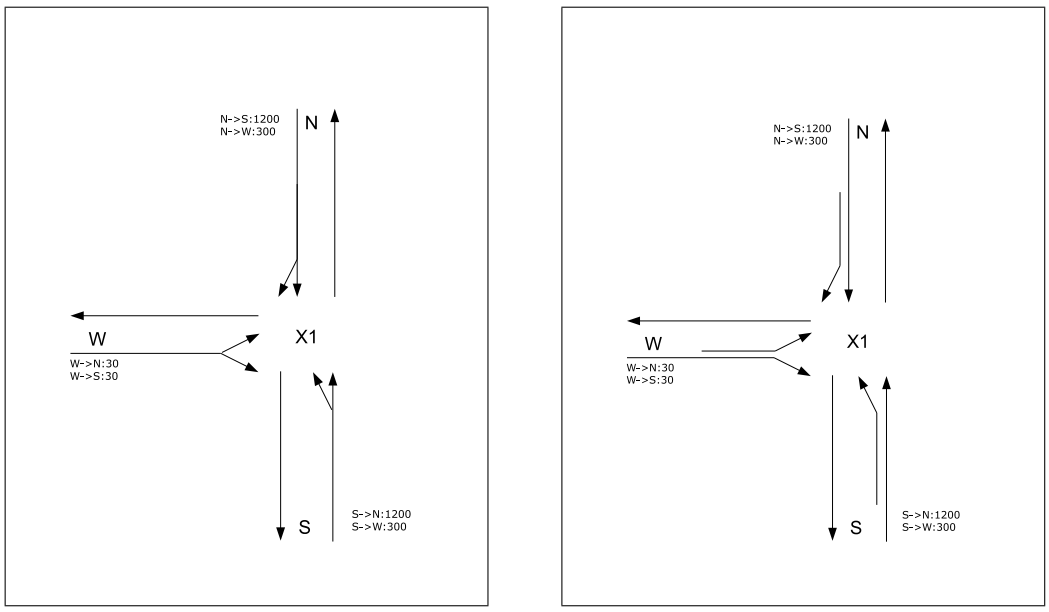
Model podstawowy: Jak pokazuje rysunek 5.1(a) jest to skrzyżowanie łączące drogę główną Północ-Południe (N-S) z drogą dojazdową z zachodu (W).

Schemat ruchu: Symulacja zaplanowana jest na jedną godzinę (3600 tur). Podróże generowane są jednostajnie przez cały czas trwania testu. Ilość pojazdów wyjeżdżających na różne trasy:

	N	S	W
N	x	1200	30
S	1200	x	30
W	300	300	x

Ten schemat ruchu, charakteryzuje się dużo większą liczbą podróży na godzinę na drodze głównej (1200 pojazdów na godzinę) w porównaniu do ilości podróży na drodze

bocznej (zachodniej).

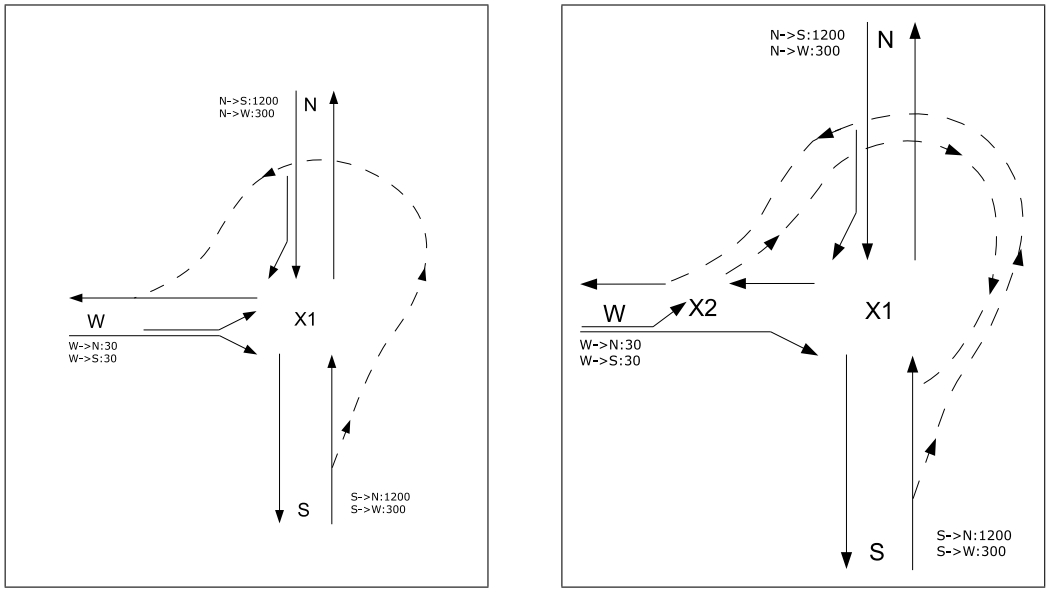


(a) model bazowy (b) model z pasami do skrętu w lewo

Rysunek 5.1. Skrzyżowanie T: model podstawowy oraz model z pasami do skrętu w lewo.

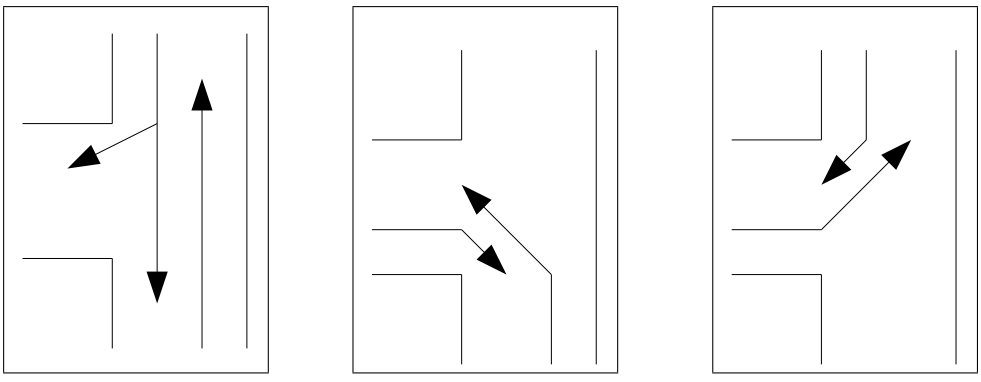
Modyfikacje:

1. Pasy do skrętu w lewo i w prawo.
   Do drogi która łączy się ze skrzyżowaniem od południa dodano pas do skrętu w lewo o długości 35 komórek, do drogi północnej pas do skrętu w prawo – 10 komórek, a do drogi zachodniej pas do skrętu w lewo – 20 komórek(patrz rys. 5.1(b)).
2. Sygnalizacja świetlna.
   Tak jak w poprzednim modelu (rys. 5.1(b)) drogi posiadają pasy do skrętu w lewo. Pas do skrętu w prawo na drodze N→W wydłużono do 20 komórek (światła mogą wydłużyć kolejkę na tym pasie). Dodatkowo na skrzyżowaniu zainstalowana jest sygnalizacja świetlna akomodacyjna z trzema fazami. Połączenie strumieni ruchu w grupy pokazane jest na rys. 5.3. W tym teście za sterowanie sygnalizacją odpowiedzialny jest algorytm *SOTL*, dane do sterownika dostarczają pętle indukcyjne. Na każdym pasie 20 komórek przed skrzyżowaniem umieszczona jest pierwsza pętla (wjazdowa), która zlicza pojazdy dojeżdżające do skrzyżowania, a przy samym skrzyżowaniu umieszczona jest pętla zliczająca pojazdy już opuszczające dany pas (wyjazdowa). Dzięki temu sterownik sygnalizacji posiada informacje o długości kolejki pojazdów.



(a) rampa zjazdowa (b) rampa dwukierunkowa

Rysunek 5.2. Skrzyżowanie T: modyfikacje modelu podstawowego polegające na użyciu rampy zjazdowej.



Rysunek 5.3. Trzy fazy programu sygnalizacji na skrzyżowaniu T

3. Rampa zjazdowa jednokierunkowa.
   Zlikwidowano możliwość skrętu w lewo na skrzyżowaniu i dodano konstrukcję rampy, która umożliwia bezkolizyjny zjazd z drogi głównej na drogę zachodnią (W). Drogi N → X oraz W → X posiadają odpowiednio pasy do skrętu w prawo i w lewo (rys. 5.2(a)).
4. Rampa dwukierunkowa (zjazd i wjazd).
   Model podstawowy z rampą dwukierunkową, która umożliwia zarówno zjazd z drogi głównej z południa (S → N) na zachód oraz wjazd z drogi zachodniej na drogę główną w kierunku północnym (patrz. 5.2(b)). Aby rampa była dwukierunkowa, trzeba zbudować kolejne skrzyżowanie. To skrzyżowanie łączy drogi o małym natężeniu ruchu. Dlatego zakładam, że zysk wynikający z wyeliminowania wszystkich skrętów w lewo będzie dużo większy niż opóźnienia wprowadzone na kolejnym skrzyżowaniu.

Analiza wyników: Z kierunków N oraz S wyjeżdża 1500 (łącznie 3000, na jednopasmowej drodze jest to bardzo intensywny ruch) pojazdów na godzinę. W idealnych warunkach wszystkie powinny mieć możliwość przejechania przez skrzyżowanie. Największą wadą w podstawowym modelu jest występowanie tras kolizyjnych. Jeżeli dwie różne trasy kolidują ze sobą, to w danej chwili ruch może odbywać się tylko na jednej z nich. Wyniki symulacji ruchu na modelu standardowym zaprezentowano na rys. 5.4(a). Różnica w natężeniu ruchu na różnych kierunkach (30 $\frac{\text{pojazdów}}{godz.}$ Wroad-X1 oraz 1500 $\frac{\text{pojazdów}}{godz.}$ NroadX1 i SroadX1) jest bardzo duża, dlatego wykresy są wykonane w skali logarytmicznej.

Jak widać na tym wykresie, w czasie trwania symulacji wszystkie pojazdy jadące z kierunku północnego opuszczają skrzyżowanie. Jest to kierunek na drodze głównej całkowicie niezależny od innych. Duże natężenie w kierunku z *Nroad* do *Sroad* powoduje, że skręcający w lewo z kierunku południowego *Sroad* muszą długo czekać i blokują cały pas drogi południowej *SroadX1*. W efekcie w kierunku *z Sroad do Nroad* przejeżdża o połowę mniej pojazdów, niż z kierunku przeciwnego *z Nroad do Sroad* (chociaż w obydwu kierunkach generowane jest tyle samo podróży). Jest to poniżej 800 pojazdów na godzinę z południa i około 1500 pojazdów na godzinę z północy. Z kierunku *WroadX1* tylko jeden pojazd zdołał przejechać przez skrzyżowanie, podczas gdy ilość tras wygenerowanych w tym kierunku wynosi 60. Jest to trasa podrzędna dla drogi północ-południe i przy tak dużym natężeniu ruchu na drodze głównej przejazd przez skrzyżowanie jest niemożliwy.

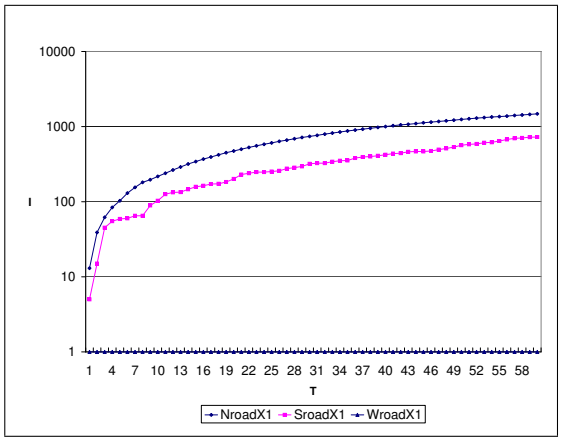
Wprowadzenie do modelu pasów do skrętu w lewo dla jadących z południa na zachód oraz dla jadących z zachodu na północ (rys. 5.4(b)) poprawia płynność ruchu. Kierowcy, którzy chcą skręcić w lewo z drogi *SroadX1* nie blokują już ruchu na trasie południe-północ (oczywiście do momentu aż zajmą cały pas do skrętu w lewo). Liczba pojazdów przejeżdżających przez skrzyżowanie z południa wzrosła z 725 do 925. Z kierunku zachodniego przejechało 21 pojazdów. Ponieważ pas do skrętu w lewo ma pojemność 20 pojazdów, oznacza to iż przejechały tylko pojazdy skręcające w prawo i to nie wszystkie. Kierowcy mający zamiar skręcić w lewo zablokowali pas do skrętu w lewo i całą drogę *WroadX1*. Na trasie z *Wroad* do Nroad nie ma żadnej poprawy. Przy dużym ruchu na drodze głównej kierowcy nie mogą skręcić w lewo z drogi podrzędnej.

Sygnalizacja świetlna (rys. 5.4(c)) rozwiązuje problemy blokowania pasów do skrętu w lewo. Dzięki temu wszystkie pojazdy mogą przejechać przez skrzyżowanie. Jedyną wadą wynikającą z zastosowania sygnalizacji świetlnej jest fakt cyklicznego blokowania kierunków północ południe przez światło czerwone.

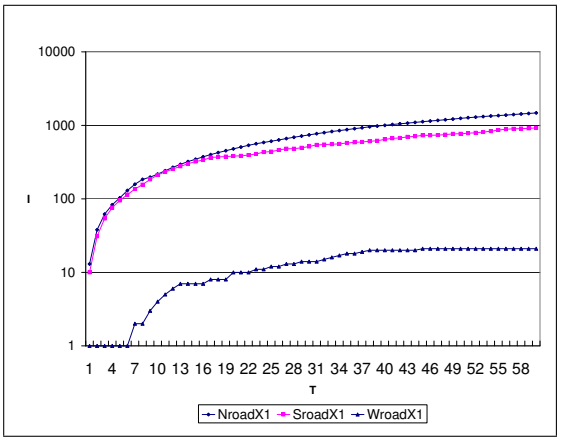
Jak pokazują wyniki testów dla modeli z rampą (tzw. ślimak), problemy związane z blokowaniem kierunków kolizyjnych można całkowicie wyeliminować tylko rozdzielając te kierunki w przestrzeni. W przypadku jednokierunkowej rampy zjazdowej, kierowcy którzy chcą skręcić w lewo nie muszą już ustępować pierwszeństwa pojazdom jadącym z północy. Wjeżdżają na rampę i przy zjeździe na drogę X1Wroad ustępują pierwszeństwa tylko pojazdom jadącym od skrzyżowania X1 do Wroad (gdzie panuje o wiele mniejszy ruch). Pojazdy na trasie południe-zachód nie są blokowane przez inne pojazdy, również same nie blokują ruchu. Pojazdy jadące z zachodu — chociaż nadal ustępują pierwszeństwa pojazdom z północy i południa — mają możliwość przejazdu przez skrzyżowanie.

Rampa dwukierunkowa (rys. 5.2(b) na stronie 90) wymaga wprowadzenia dodatkowego skrzyżowania — X2. Ponieważ jest to skrzyżowanie o małym natężeniu ruchu, kolejki pojazdów są małe. Główne skrzyżowanie — X1, gdzie natężenie ruchu jest bardzo duże, jest teraz całkowicie bezkolizyjne. Ruch odbywa się płynnie, bez zatrzymań. Jedynie pojazdy jadące z zachodu na południe muszą poczekać, aby skręcić w prawo. W badanym przypadku nie ma zysku z wprowadzenia dodatkowego skrzyżowania do modelu oraz dobudowania drugiego kierunku do rampy. Na rysunkach 5.5 oraz 5.6 przedstawione są podsumowane wyniki tego eksperymentu.

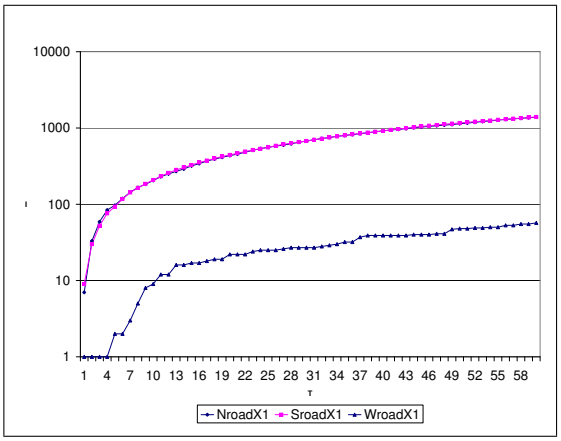
W systemie na każdą minutę eksperymentu generowanych jest ok. 50 podróży. Aby ruch był płynny, skrzyżowanie powinno mieć przepustowość przynajmniej 50 pojazdów na minutę. W innym przypadku więcej pojazdów będzie przybywało do



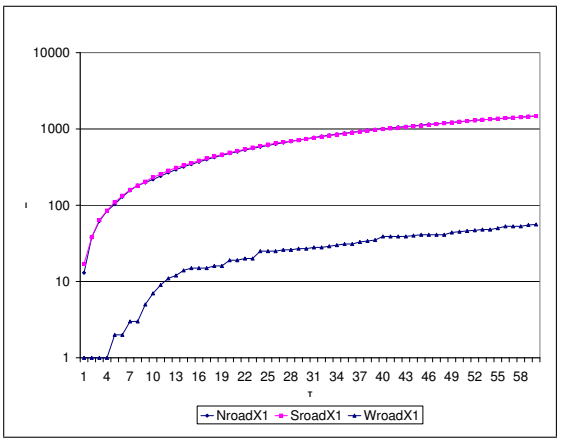
(a) model bazowy



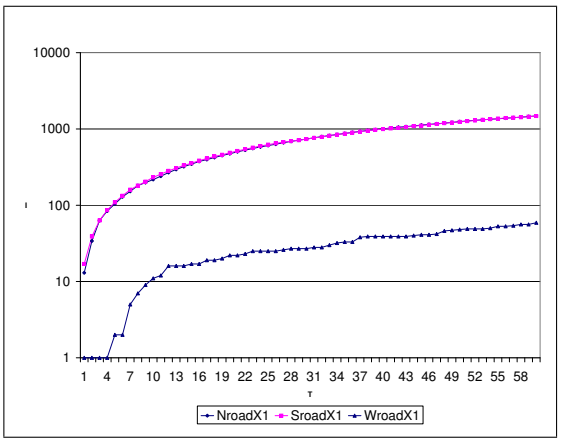
(b) lewoskręty



(c) sygnalizacja świetlna

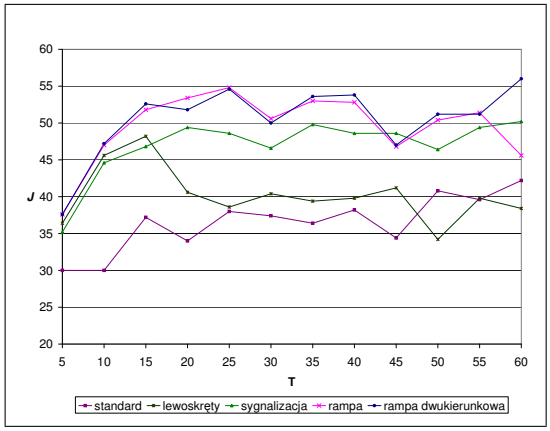


(d) rampa zjazdowa

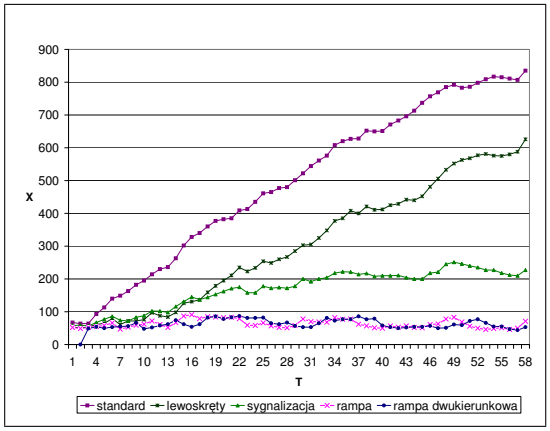


(e) rampa dwukierunkowa

Rysunek 5.4. Skrzyżowanie T: wykresy przedstawiają ilość przejazdów przez skrzyżowanie z kierunków: południowy – SroadX1, północny – NroadX1, zachodni – WroadX1. Na wykresach c), d) oraz e) punkty SroadX1 i NroadX1 pokrywają się.

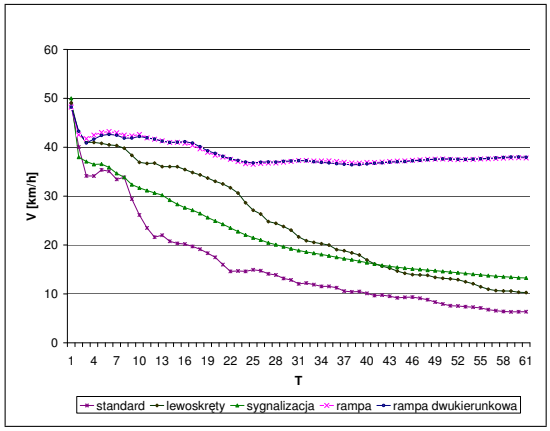


(a) przepływ

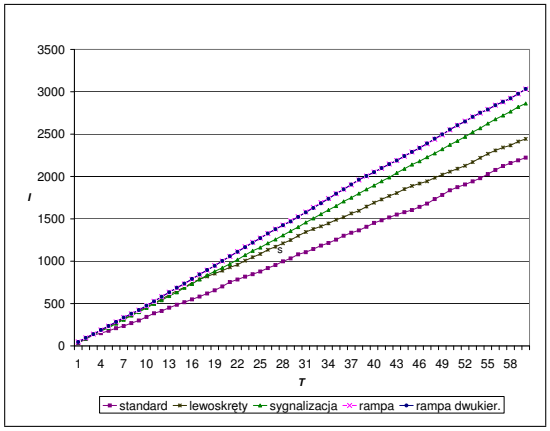


(b) ilość pojazdów

Rysunek 5.5. Przepustowość i pojemność skrzyżowania T: (a) Przepływ ruchu (ilość pojazdów przejeżdżająca przez skrzyżowanie na minutę); (b) ilość pojazdów w sieci, w tym przypadku w obrębie jednego skrzyżowania, oraz na pasach dojazdowych do niego.



(a) średnia prędkość



(b) ilość przejazdów

Rysunek 5.6. Skrzyżowanie T — Średnia prędkość i liczba przejazdów: (a) średnia prędkość na skrzyżowaniu w kolejnych minutach eksperymentu; (b) całkowita ilość przejazdów przez skrzyżowanie.

systemu niż z niego wyjeżdżało. Rysunek 5.5(a) pokazuje średnią przepustowość na wszystkich wersjach badanego skrzyżowania. Model bazowy ma przepustowość na poziomie 35–40 pojazdów na minutę, model z pasami do skrętu ok. 40. Te wartości są dużo niższe niż wymagane minimum, więc gęstość ruchu będzie rosła z każdą minutą. Model z sygnalizacją świetlną ma przepustowość na poziomie 50 pojazdów na minutę. Jest to rozwiązanie niestabilne, ponieważ dowolne zaburzenie w ruchu (dłuższe zatrzymanie lub większa liczba pojazdów wjeżdżających w okolicę) może spowodować nagły wzrost zatłoczenia i korek. To przypuszczenie potwierdza trend wzrostowy na wykresie (rys. 5.5(b)) ilości pojazdów w modelu, warto zauważyć, że 200 pojazdów w modelu o pojemności 300 pojazdów to już sytuacja dużego zatłoczenia. Na samym wykresie przepustowości(rys. 5.5(a)) można zauważyć, że obydwa rozwiązania z rampą spełniają wymagania przepustowości większej niż 50 pojazdów na minutę.

Wykres 5.5(b) przedstawia ilość pojazdów w sieci. Pojemność symulowanego modelu to 300 pojazdów, na wykresie uwzględnione są kolejki pojazdów oczekujących na wjazd do modelu (dlatego można zauważyć wartości wyższe niż 300). Wzrostowy trend tego wykresu świadczy o ciągłym wzroście zagęszczenia, czyli pogarszanie się stanu ruchu drogowego. W modelu bazowym zagęszczenie osiąga 100% (czyli każda komórka jest zajęta przez pojazd) już w 16-tej minucie eksperymentu. Następne pojazdy nie mogą nawet wjechać do symulowanego modelu.

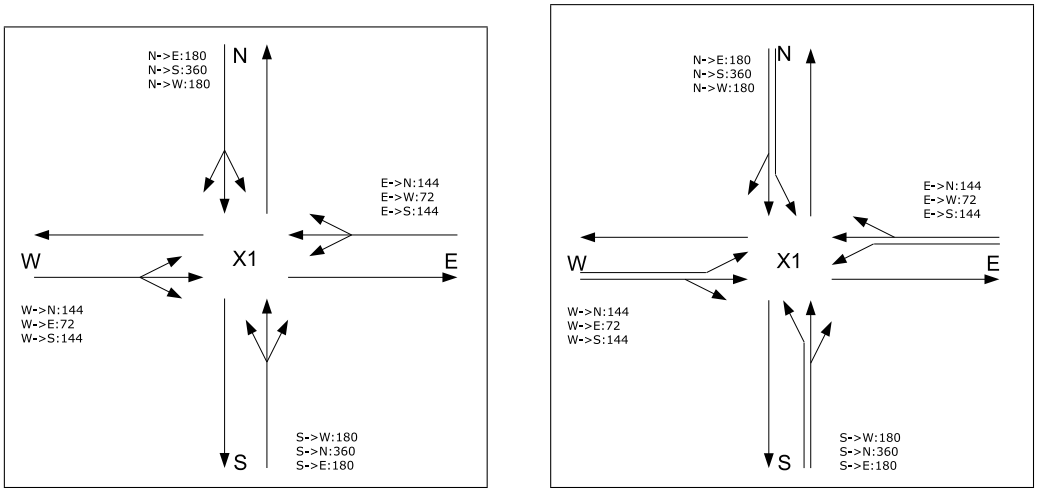
Gdy generacja ruchu wejściowego kończy się w 60-tej minucie, potrzeba jeszcze 30 minut, aby wszystkie pojazdy oczekujące mogły przejechać skrzyżowanie. W modelu z lewoskrętami model przepełnia się w 30 minucie (to tylko różnica skali), a po zakończeniu generacji nowych podróży potrzeba 20 minut na zakończenie testu. Wykres ilości pojazdów dla modelu z sygnalizacją świetlną również ma tendencję wzrostową, a samo zagęszczenie jest duże. Jednak tutaj nie dochodzi do przepełnienia pojemności modelu. Modele z rampą są wystarczające dla tego schematu ruchu, ilość pojazdów w modelu jest na stałym poziomie ok. 100 pojazdów. Jest to niewielkie zagęszczenie, co świadczy o luźnym ruchu. Dla zobrazowania jak wzrost zagęszczenia wpływa na średnią prędkość przejazdu wykonany został rysunek 5.6(a).

5.1.2. Model skrzyżowania drogi głównej z drogą podrzędną

Test jest przeprowadzany na skrzyżowaniu dwóch dróg: drogi głównej — NS z drogą poboczną — WE.

Model podstawowy: Jak pokazuje rysunek 5.7(a) jest to skrzyżowanie łączące

drogę główną Północ-Południe (N-S) z drogami z zachodu (W) i ze wschodu (E). Dozwolone są wszystkie kierunki ruchu (dla uproszczenia pominięto tylko zawracanie).



(a) model podstawowy

(b) model pasami do skrętu w lewo na wszystkich kierunkach

Rysunek 5.7. Skrzyżowanie X: model podstawowy i modyfikacje.

Schemat ruchu: Czas symulacji to jedna godzina (3600 tur). Podróże generowane są jednostajnie przez cały czas trwania testu. Rozkład podróży:

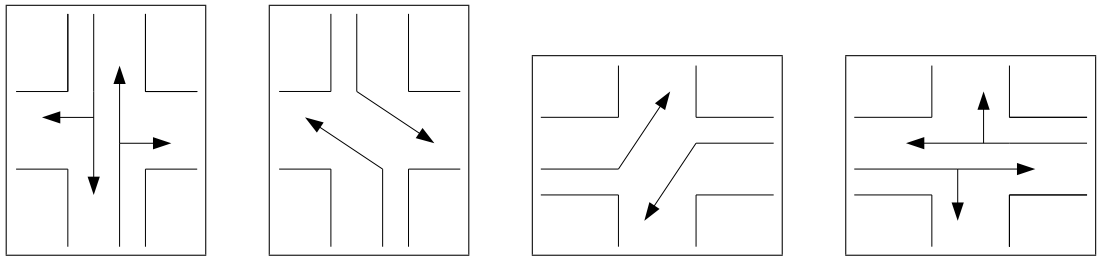
	N	E	S	W
N	x	180	360	180
E	144	x	144	72
S	360	180	x	180
W	144	72	144	x

Stosunek liczby podróży z północy i południa do liczby podróży z kierunków wschód i zachód wynosi 2:1. Jest to skrzyżowanie dwóch dróg o podobnej klasie, jedna z nich jest wyróżniona jako główna.

Modyfikacje:

1. Pasy do skrętu w lewo na drodze głównej.
   Droga główna posiada pasy do skrętu w lewo. Teraz pojazdy skręcające w lewo z drogi N-X oraz S-X nie blokują ruchu tym, które jadą prosto.
2. Pasy do skrętu w lewo na wszystkich drogach.
   Wszystkie drogi połączone tym skrzyżowaniem posiadają pasy do skrętu w lewo (rys 5.7(b)).

3. Sygnalizacja świetlna.



(a) (b) (c) (d)

Rysunek 5.8. Plan sygnalizacji świetlnej skrzyżowania X: (a) na wprost i w prawo: NS, SN; (b) w lewo: NE, SW; (c) na wprost i w prawo: WE, EW, (d) w lewo: WN, ES.

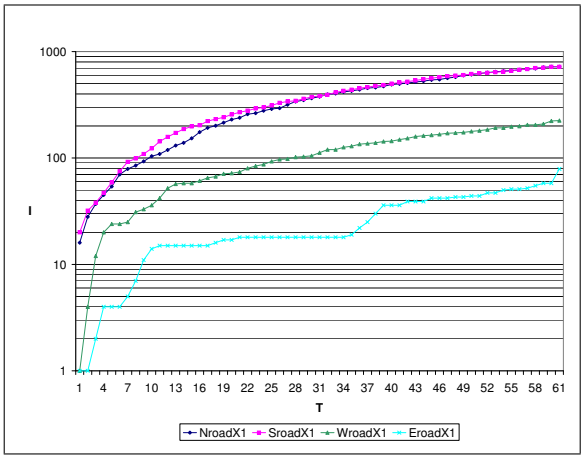
W tym modelu, tak jak w podpunkcie 2, pasy do skrętu w lewo występują na wszystkich drogach. Na skrzyżowaniu zainstalowana jest akomodacyjna sygnalizacja świetlna z czterema fazami ruchu (patrz rys. 5.8).

Analiza wyników: Szczegółowe wyniki symulacji ruchu dla wszystkich modeli zaprezentowano na rysunku 5.9. Jest to ilustracja ilości przejeżdżających przez skrzyżowanie pojazdów w kolejnych minutach symulacji. Przez skrzyżowanie powinno przejechać 720 pojazdów z kierunku północnego, tyle samo z kierunku południowego oraz po 360 pojazdów z kierunków: wschodniego i zachodniego.

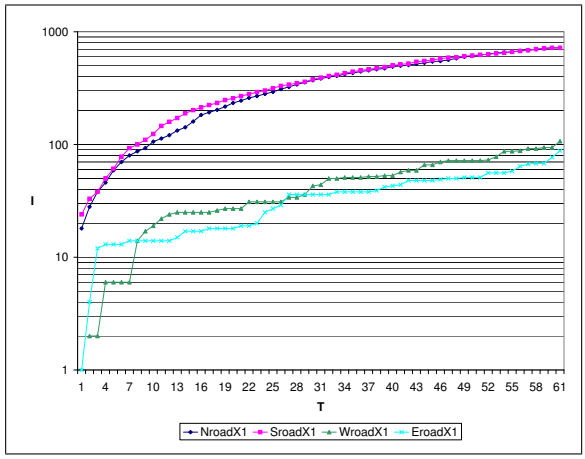
Analizując rys. 5.9(a) dla modelu podstawowego, można zauważyć, że wszystkie pojazdy z kierunków północ i południe zdołają przejechać w trakcie symulacji. Z kierunku zachodniego w tym czasie przejeżdża tylko 225 pojazdów, a z kierunku wschodniego 80 pojazdów[1]. Przy tak intensywnym ruchu na drodze głównej, kierowcy na drogach podporządkowanych nie mają warunków umożliwiających przejazd przez skrzyżowanie.

Dodatkowe pasy do skrętu w lewo z drogi głównej (rys. 5.9(b)) nie poprawiają sytuacji. Zwiększyła się płynność ruchu na drodze głównej (oraz średnia prędkość na tej drodze), natomiast sytuacja na drodze podporządkowanej jest jeszcze gorsza. Z kierunków wschód i zachód przez skrzyżowanie przejechało odpowiednio 107 i 88 pojazdów.

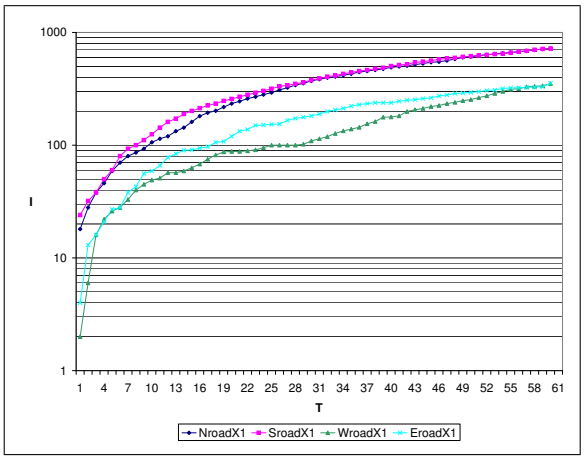
[1] Taka różnica pomiędzy kierunkami wschód i zachód może wynikać z zastosowania iteratora, który po kolei przegląda drogi wejściowe w celu znalezienia pojazdów wjeżdżających na skrzyżowanie, kierunek zachodni jest wybierany przed wschodem. Jest to element wymagający udoskonalenia. Wybór powinien być losowy



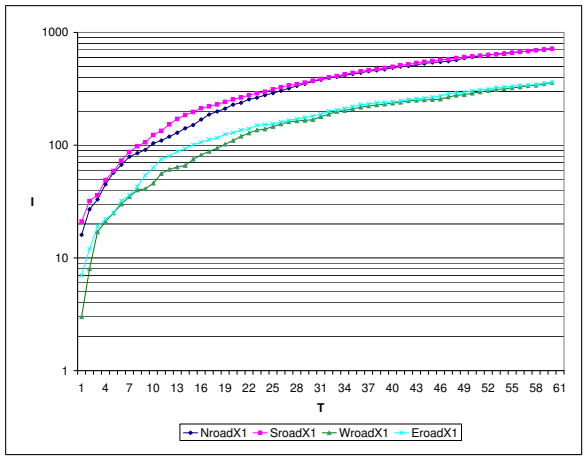
(a) model podstawowy



(b) pasy do skrętu w lewo na drodze głównej



(c) pasy do skrętu w lewo na wszystkich drogach



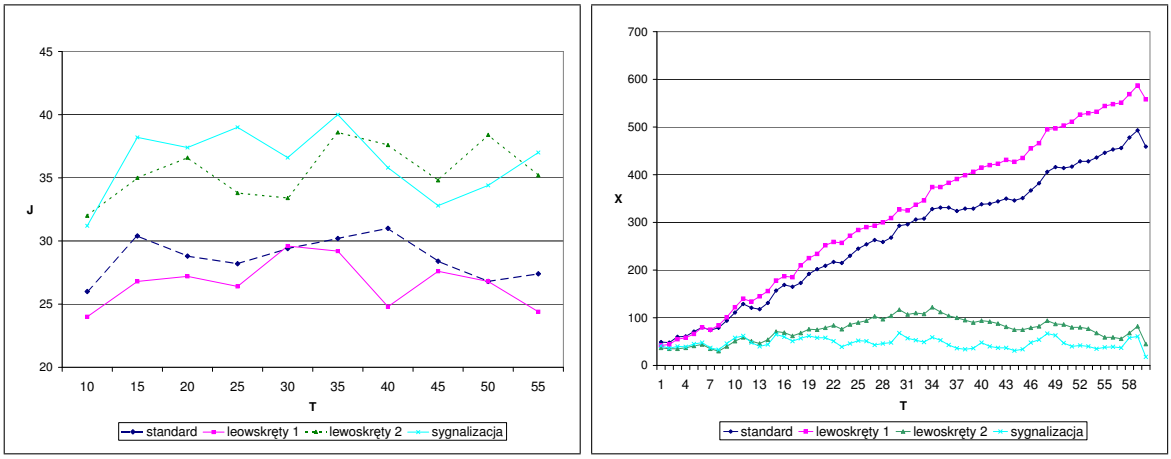
(d) sygnalizacja świetlna

Rysunek 5.9. Ilość przejazdów przez kolejne warianty skrzyżowania X z kierunków: południe, północ, zachód i wschód (odpowiednio SroadX1, NroadX1, WroadX1, EroadX1).

Pasy do skrętu w lewo na wszystkich drogach rozwiązują problem tego skrzyżowania(patrz rys. 5.9(c)). Kierowcy, którzy chcą skręcić w lewo z drogi podporządkowanej, nie blokują już pozostałego ruchu z tej drogi. Prawie wszystkie pojazdy wygenerowane na potrzeby tego eksperymentu przejeżdżają przez skrzyżowanie w ciągu jednej godziny. Pozostaje mniej niż 10 pojazdów, które mogą opuścić skrzyżowanie w ciągu jednej minuty.

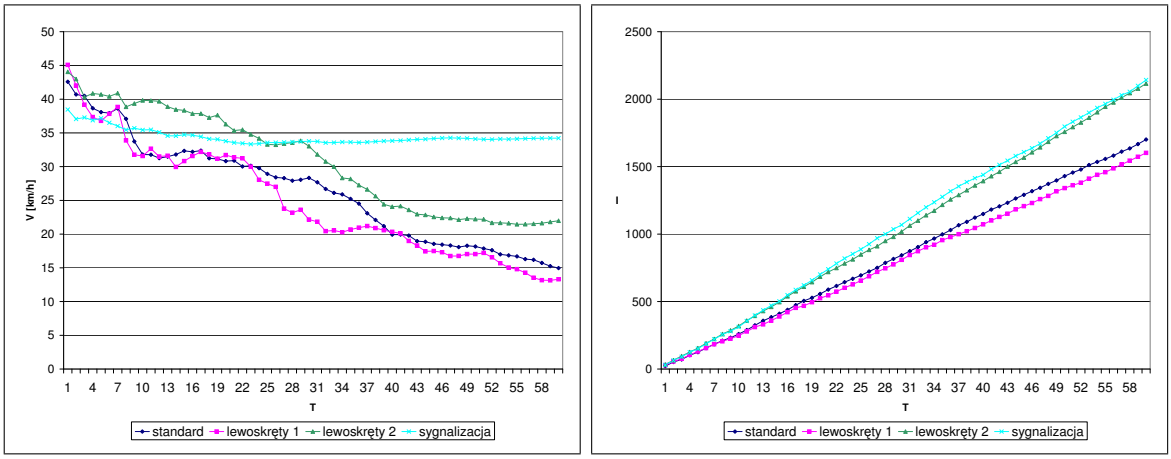
Po zainstalowaniu sygnalizacji świetlnej na tym skrzyżowaniu można zaobserwować dalsze poprawienie warunków ruchu (patrz rys. 5.9(d)). Wszystkie pojazdy opuszczają skrzyżowanie w ciągu jednej godziny.

W tym eksperymencie generowane jest ok. 36 podróży na minutę. Rysunek 5.10(a) pokazuje przepływ strumienia ruchu dla modelu podstawowego i wszystkich modyfi-



(a) przepływ (b) ilość pojazdów

Rysunek 5.10. Przepustowość i pojemność skrzyżowania X: (a) Przepływ ruchu (ilość pojazdów przejeżdżająca przez skrzyżowanie na minutę); (b) ilość pojazdów w sieci, w tym przypadku w obrębie jednego skrzyżowania, oraz na pasach dojazdowych do niego.



(a) średnia prędkość (b) ilość przejazdów

Rysunek 5.11. Skrzyżowanie X — Średnia prędkość i liczba przejazdów: (a) średnia prędkość na skrzyżowaniu w kolejnych minutach eksperymentu; (b) całkowita ilość przejazdów przez skrzyżowanie.

kacji. Przez skrzyżowanie w modelu podstawowym przejeżdża od 25–30 pojazdów na minutę i jest to mniej niż liczba generowanych podróży. Efektem tego jest rosnące zatłoczenie na skrzyżowaniu i w jego obrębie (co potwierdza rys. 5.10(b)). Podobnie wygląda sytuacja w modelu z pasami do skrętu w lewo na drodze głównej.

Pasy do skrętu w lewo na wszystkich drogach prowadzących do skrzyżowania znacząco zwiększają jego przepustowość. Podczas symulacji wszystkie pojazdy mogły opuścić skrzyżowanie (przejeżdżało przez nie od 30 do 40 pojazdów na minutę).

Patrząc na dane zaprezentowane na rys. 5.10(a) można twierdzić, że instalacja sygnalizacji świetlnej nie przynosi dalszej poprawy. Jak pokazuje rys. 5.10(b) ilość pojazdów w systemie (a w związku z tym również gęstość ruchu) jest w symulacji modelu z pasami do skrętu w lewo dwukrotnie większa niż w symulacji tego modelu rozszerzonego o sygnalizacją świetlną. W tym przypadku gęstość ruchu wzrasta z 25% do ponad 50%. Wystarczy taka różnica, aby ze stanu ruchu luźnego system przeszedł do stanu zatłoczenia lub nawet korka. Dane zaprezentowane na rys. 5.11(a) pokazują, że średnia prędkość w modelu z pasami do skrętu w lewo spada z każdą minutą symulacji (do poziomu ok. 20 km/h). Potwierdza to, że w systemie występuje stan zatłoczenie. Szczegółowa analiza wszystkich danych (szczególnie rys. 5.9(c) oraz rys. 5.9(d)), doprowadza do wniosku, że korek panuje na pasach do skrętu w lewo na drodze podrzędnej. Przy tym samym schemacie ruchu wejściowego, prędkość w modelu z sygnalizacją świetlną jest stała i wynosi ok. 35 km/h (uwzględnione są czasy oczekiwania na czerwonym świetle).

5.2. Prosty model miasta

Zmierzono czas trwania symulacji (do zniknięcia ostatniego samochodu), średnią długość przejazdu pomiędzy dowolnymi dwoma węzłami wlotowymi, odchylenie standardowe tej wartości, średnią prędkość pomiędzy dowolnymi dwoma konkretnymi węzłami wlotowymi, a także średnią prędkość na każdym połączeniu.

Standardowy model miasta. Na potrzeby eksperymentu zbudowano uproszczony model miasta Kraków (rys. 5.12). Ponieważ model będzie poddawany modyfikacjom, ta wersja jest nazywana *modelem standardowym*. Długości dróg tego modelu zebrane są w tablicy 5.1, długości podano w komórkach modelu oraz w metrach.

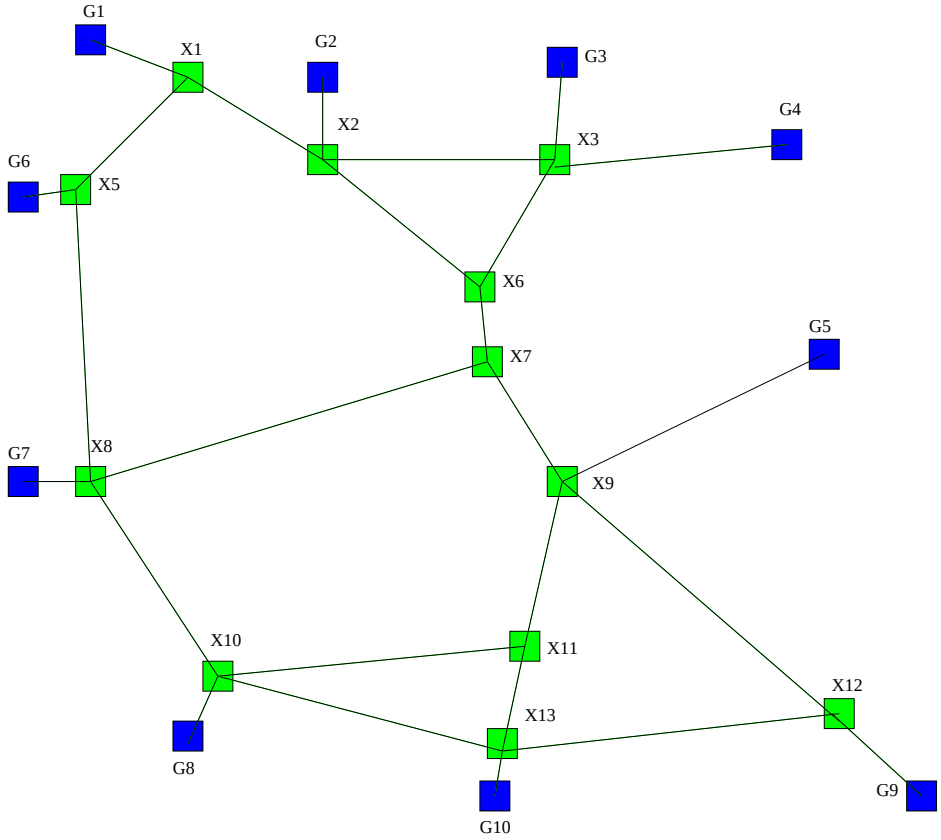
Schemat ruchu. We wszystkich testach użyto podobnego schematu ruchu. Pomiędzy każdą parą węzłów wlotowych w przeciągu 12h (43200 tur) było generowanych

droga	długość [komórka(metr)]
G1-X1	100 (750)
X5-X1	250 (1875)
X1-X2	250 (1875)
G2-X2	100 (750)
X2-X3	150 (1125)
X2-X6	350 (2625)
G3-X3	100 (750)
X3-G4	100 (750)

droga	długość [komórka(metr)]
X3-X6	250 (1875)
G6-X5	100 (750)
X5-X8	450 (3375)
X6-X7	150 (1125)
X8-X7	650 (4875)
X7-X9	200 (1500)
G7-X8	100 (750)
X8-X10	350 (2625)
X9-G5	350 (2625)

droga	długość [komórka(metr)]
X9-X12	550 (4125)
X9-X11	250 (1875)
X10-X11	450 (3375)
X10-X13	450 (3375)
X10-G8	100 (750)
X11-X13	150 (1125)
X13-X12	550 (4125)
X12-G9	100 (750)
X13-G10	100 (750)

Tablica 5.1. Model miasta. Długości dróg.



Rysunek 5.12. Standardowy model miasta

n samochodów według jednostajnego rozkładu prawdopodobieństwa. Oznacza to, że z każdego węzła wlotowego wjeżdża do miasta $9 \cdot n$ pojazdów w ciągu 12h. Co istotne, za każdym razem podawano ten sam zarodek generatora liczb losowych używanego przez generator ruchu, dzięki czemu uzyskano powtarzalność generacji ruchu.

Warto zauważyć, że jest to bardzo prosty schemat. Przy jego tworzeniu nie brano pod uwagę umiejscowienia takich obiektów jak teatry, kina, supermarkety, centrum

miasta i osiedla mieszkalne. Nie jest to realistyczna symulacja ruchu miejskiego, a jedynie eksperyment mający pokazać zachowanie modelu (chociaż sieć zbudowana jest w oparci o mapę Krakowa). W ramach kolejnego projektu można podjąć próbę przeprowadzenia symulacji z użyciem modelu odzwierciedlającego rzeczywistość oraz używając danych pomiarowych dotyczących ruchu w Krakowie (dane dostępne są w urzędzie miasta w dokumentacji KBR 2003[2]).

Parametry modułu fizycznego. Podczas testowania użyto jednakowych parametrów (opisano w rozdziale 4.5.3):

- *cellLenght* = 7.5 m,
- *turnDuration* = 1 s,
- *maxVelocity* = 2, co odpowiada ok. 54 km/h,
- *decelProb* = 0.2,
- *priorLaneTimeHeadway* = 4.

Parametry modułu oceny SOTL. Ustawiono domyślne wartości parametru określającego długość strefy kolejki przed skrzyżowaniem — *zoneLength* (20).

Parametry modułu oceny RL. Ustawiono domyślne wartości parametrów algorytmu — *discount* (0,9), *halvePeriod* (60) oraz przyjęto ograniczenie strefy kolejki przed skrzyżowaniem do 20 komórek (podobnie jak w przypadku metody SOTL).

5.2.1. Wpływ natężenia ruchu na średnią prędkość

Celem tego eksperymentu jest zbadanie zachowania się algorytmów sterowania światłami (SOTL oraz RL) w różnych warunkach drogowych oraz ocena wpływu natężenia ruchu na średnią prędkość w modelu miasta. Długość stanu przejściowego była stała i wynosiła 8, a zmienną była ilość samochodów generowanych na każdej trasie. Użyto wartości, które mają symulować małe, średnie oraz duże natężenie ruchu — odpowiednio 300, 500 oraz 600[3].

[2] KBR 2003 — Kompleksowe Badania Ruchu przeprowadzone w roku 2003

[3] Dla porównania, w systemie KRAKSIM przed modyfikacjami modelu skrzyżowania, do podobnego testu użyto wartości 100, 300 oraz 500. Modyfikacje wpłynęły na podniesienie realności zachowania modelu, a przede wszystkim znacząco zwiększyły przepustowość skrzyżowania z sygnalizacją świetlną.

Model miasta. Do eksperymentu użyto modelu i schematu ruchu opisanego w rozdziale 5.2 na stronie 100. Dokonano testów dla różnych natężeń ruchu. Natężeniem ruchu sterowano poprzez zmianę parametru n, który określa liczbę pojazdów generowanych na jedną trasę w ciągu 12 godzin symulacji. Pojazdy generowane są z rozkładem jednostajnym. Wartości parametru n użyte w testach to: 300, 500, 600, co odpowiada natężeniom ruchu: $225 poj/h$, $375 poj/h$ oraz $450 poj/h$.

Analiza wyników. Tabele 5.2, 5.3, porównują wyniki pomiarów o charakterze globalnym dla tych trzech przypadków. Wartość *długość symulacji* określa, ile tur (równocześnie sekund, bo jedna tura trwa sekundę) upłynęło od początku symulacji do momentu gdy ostatni pojazd opuścił model. Ilość wszystkich wygenerowanych pojazdów jest ściśle określona przez schemat ruchu.

	ilość samochodów na każdej trasie[n (p/h)]		
	300 (225)	500 (375)	600 (450)
długość symulacji [s]	44296	44386	49013
średnia prędkość [k/s (km/h)]	1,62 (43,7)	1,52 (41,0)	0,68 (18,4)

Tablica 5.2. Wyniki pomiarów o charakterze globalnym dla modułu oceny SOTL przy różnym natężeniu ruchu; model standardowy, długość stanu przejściowego: 8

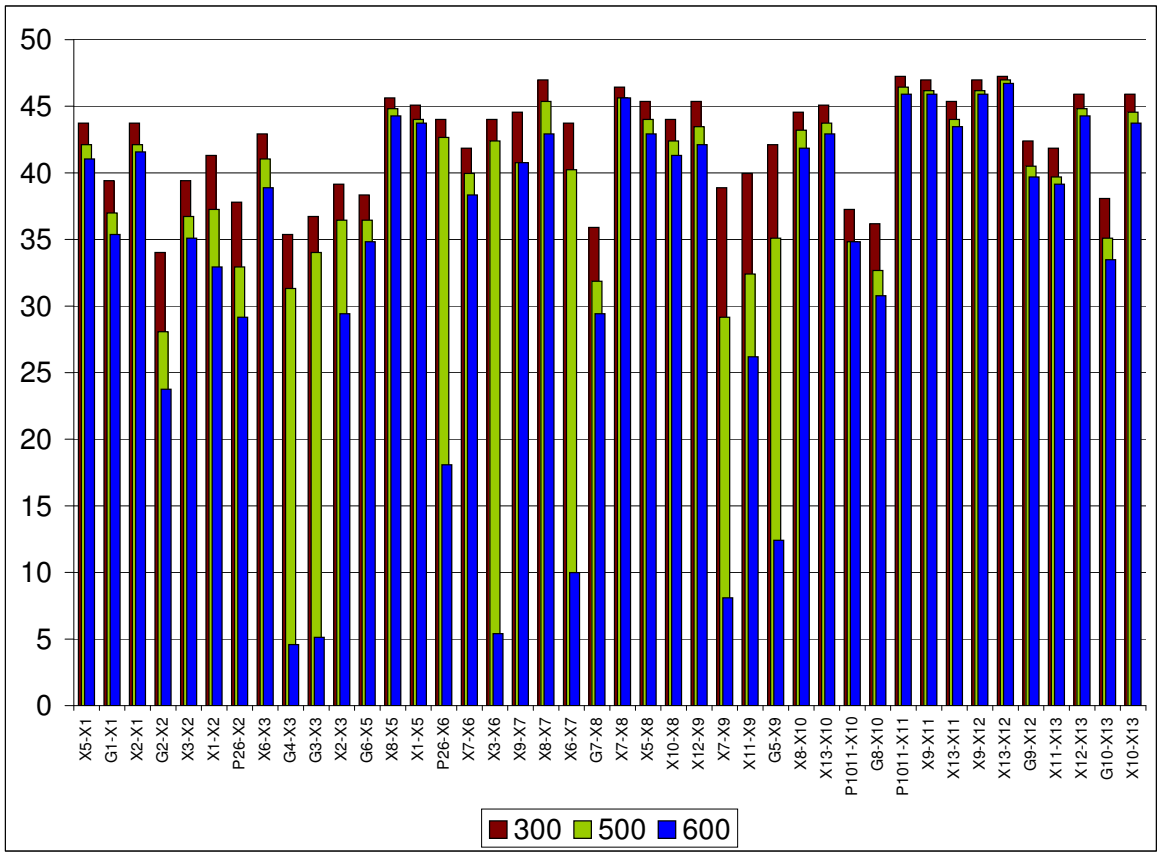
	ilość samochodów na każdej trasie [n (p/h)]		
	300 (225)	500 (375)	600 (450)
długość symulacji [s]	44362	44664	50695
średnia prędkość [k/s (km/h)]	1,44 (38,9)	1,37 (37,0)	0,59 (15,9)

Tablica 5.3. Wyniki pomiarów o charakterze globalnym dla modułu oceny RL przy różnym natężeniu ruchu; model standardowy, długość stanu przejściowego: 8

Obydwa algorytmy sterowania sygnalizacją dają podobne rezultaty.Nieznacznie lepszy okazuje się algorytm SOTL. Jest to spowodowane między innymi tym, że w tym algorytmie funkcja zysku z zapalenia zielonego światła uwzględnia samochody, które jeszcze nie stoją, lecz dopiero dojeżdżają do skrzyżowania (przekroczyły pierwszy detektor). W metodzie RL samochód wnosi swój wkład do oceny dopiero, gdy znajduje się w kolejce przed czerwonym światłem. Różnica tych metod powoduje, że pierwsza potrafi włączyć zielone światło zanim samochód nadjedzie, tym samym zmniejsza ilość zatrzymań i zwiększa średnią prędkość przejazdu.

Fakt, że pod uwagę brany jest tylko końcowy odcinek pasa ruchu (w tym przypadku 20 komórek), a głównym czynnikiem podlegającym ocenie jest ilość pojazdów na tym

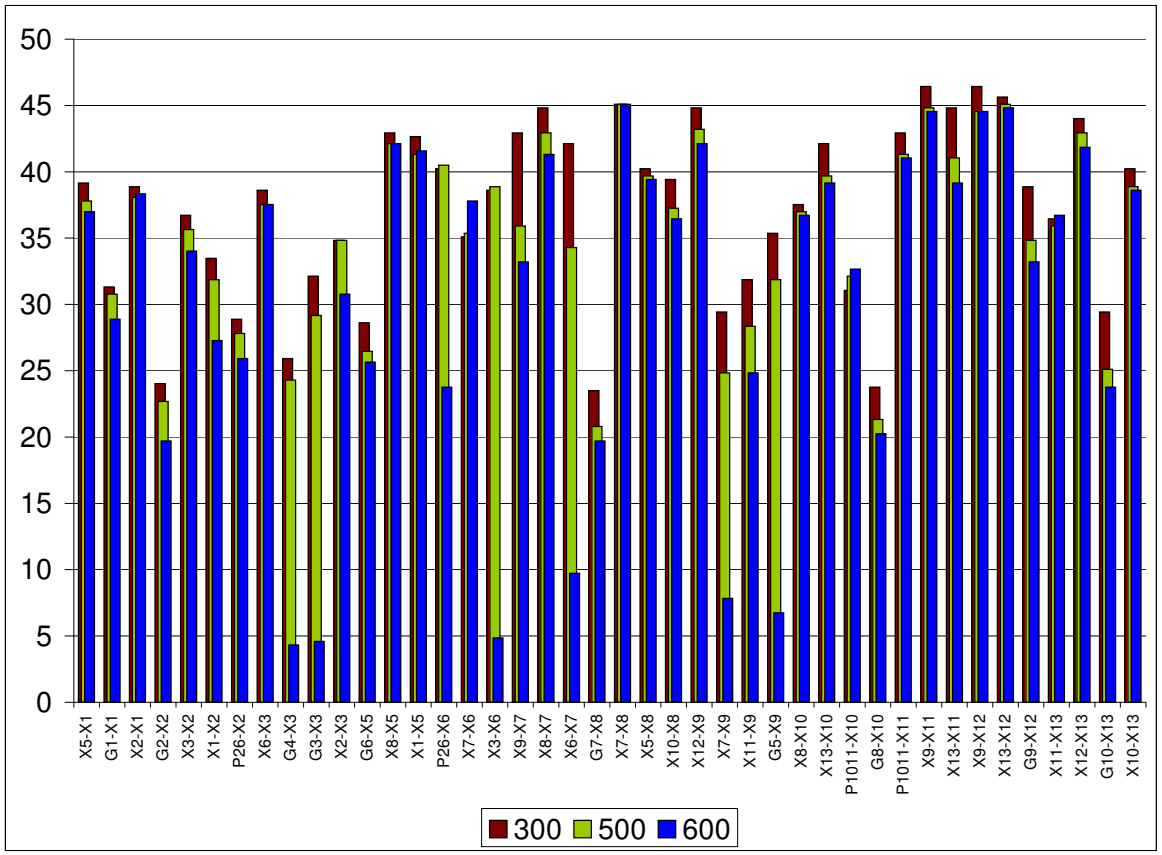
fragmencie jezdni sprawia, że obydwie metody zachowują się bardzo podobnie. Być może zwiększenie tej strefy na całą długość pasa spowoduje poprawę w działaniu metod sterowania. Problemem może być spadek wydajności obliczeń.



Rysunek 5.13. Średnia prędkość na drogach w modelu miasta, dla różnych poziomów natężenia ruchu. Metoda sterowania sygnalizacją świetlną SOTL.

Analizując szczegółowe dane można zauważyć, że średnia prędkość wyraźnie spada wraz ze wzrostem natężenia ruchu. Wykresy 5.13 oraz 5.14 pokazują średnią prędkość w zależności od poziomu natężenia ruchu dla metod sterowania światłami SOTL i RL. Dla schematów ruchu, w których generowane jest 300 pojazdów na każdą trasę, średnie prędkości są wysokie na poziomie ok. 1,52 komórki/s dla metody SOTL i 1,44 komórki/s dla metody RL. Gdy na każdą trasę wyjeżdża 500 pojazdów, średnie prędkości nieznacznie się obniżają. Gdy tę ilość zwiększy się do 600, sytuacja znacznie się pogarsza. Pokazuje to, że przekroczona jest maksymalna przepustowość sieci, a w szczególności region skrzyżowań X6 i X7.

Wiele tras ma na swojej drodze skrzyżowania X2, X3, X6, X7, X9. Przy nich



Rysunek 5.14. Średnia prędkość na drogach w modelu miasta, dla różnych poziomów natężenia ruchu. Metoda sterowania sygnalizacją świetlną RL.

należy spodziewać się największych korków. Dane zawarte w tabeli 5.4 na stronie 106 pokazują, że na drogach łączących się na tych skrzyżowaniach ruch jest najwolniejszy.

5.2.2. Porównanie do poprzedniej wersji systemu KRAKSIM

Tabele 5.5 oraz 5.6 przedstawiają wyniki eksperymentu przeprowadzonego na *modelu standardowym* w aktualnej wersji systemu KRAKSIM 2007 oraz w poprzedniej jego wersji tj. KRAKSIM 2006[4].

Obserwuje się wzrost średniej szybkości w symulowanym modelu (względem poprzedniej wersji). Jest to bezpośrednio związane z modyfikacją obsługi skrzyżowania. W poprzedniej wersji systemu zaimplementowany był niepełny sterownik sygnalizacji na skrzyżowaniu, który umożliwiał zapalenie zielonego światła dla tylko jednego strumienia ruchu. W ramach tej pracy zaimplementowano poprawki, które umożliwiają

[4] System nazywa się KRAKSIM, bez przyrostka określającego rok powstania. Dla odróżnienia dwóch wersji w tej pracy używa się nazwy KRAKSIM (również KRAKSIM 2007) tylko dla aktualnej wersji powstałej w związku z tą pracą, natomiast poprzednia wersja jest nazywana KRAKSIM 2006

KRAKSIM 2007 natężenie 600			
link	$\overline{t}$	$\sigma(\overline{t})$	$\overline{v}$
X5-X1	164.8	16.5	1.52
G1-X1	76.0	15.8	1.31
X2-X1	162.6	15.6	1.54
X1-X2	204.6	36.1	1.22
G2-X2	113.0	32.6	0.88
X6-X2	253.2	35.8	1.41
X3-X2	115.0	23.1	1.30
G3-X3	532.2	399.8	0.19
X2-X3	138.1	50.1	1.09
X6-X3	173.5	21.9	1.44
G4-X3	575.7	473.0	0.17
X1-X5	154.7	13.1	1.62
G6-X5	77.5	13.7	1.29
X8-X5	274.1	15.2	1.64
X2-X6	382.2	172.5	1.22
X3-X6	1268.8	809.7	0.20
X7-X6	105.4	14.4	1.42
X6-X7	409.6	171.4	0.37
X8-X7	409.8	36.7	1.59
X9-X7	132.3	13.3	1.51
G7-X8	92.1	22.3	1.09
X7-X8	385.3	18.4	1.69
X5-X8	283.6	19.1	1.59
X10-X8	228.7	23.9	1.53
X12-X9	352.6	32.6	1.56
X7-X9	670.7	198.7	0.30
X11-X9	256.5	63.7	0.97
G5-X9	759.2	329.9	0.46
X8-X10	226.5	18.0	1.55
G8-X10	87.9	20.2	1.14
X13-X10	282.7	24.2	1.59
X11-X10	272.6	17.9	1.54
X10-X11	262.1	11.1	1.75
X9-X11	146.8	8.5	1.70
X13-X11	93.2	9.6	1.61
X13-X12	317.7	9.8	1.73
X9-X12	322.8	12.7	1.70
G9-X12	67.8	9.2	1.47
X10-X13	277.0	17.0	1.62
X12-X13	336.0	23.4	1.64
X11-X13	103.4	14.4	1.45
G10-X13	80.5	16.8	1.24

KRAKSIM 2007 natężenie 500			
link	$\overline{t}$	$\sigma(\overline{t})$	$\overline{v}$
X5-X1	160.4	14.4	1.56
G1-X1	73.1	13.3	1.37
X2-X1	159.8	13.5	1.56
X1-X2	180.8	24.8	1.38
G2-X2	96.2	23.9	1.04
X6-X2	237.2	27.4	1.49
X3-X2	110.2	18.2	1.36
G3-X3	79.6	20.0	1.26
X2-X3	111.1	18.7	1.35
X6-X3	164.7	14.7	1.52
G4-X3	86.3	19.6	1.16
X1-X5	153.3	11.2	1.63
G6-X5	74.2	12.2	1.35
X8-X5	271.6	13.9	1.66
X2-X6	211.1	10.1	1.67
X3-X6	159.4	12.6	1.57
X7-X6	101.4	11.2	1.48
X6-X7	100.5	12.5	1.49
X8-X7	387.6	20.1	1.68
X9-X7	132.2	13.0	1.51
G7-X8	84.6	18.9	1.18
X7-X8	384.7	17.0	1.69
X5-X8	276.7	15.7	1.63
X10-X8	222.7	19.8	1.57
X12-X9	342.0	25.6	1.61
X7-X9	185.6	38.8	1.08
X11-X9	209.2	40.6	1.20
G5-X9	268.4	36.4	1.30
X8-X10	219.2	16.2	1.60
G8-X10	82.5	17.6	1.21
X11-X10	272,9	14.6	1.54
X13-X10	277.4	19.6	1.62
X10-X11	260.1	10.5	1.75
X9-X11	146.6	8.5	1.71
X13-X11	92.2	9.1	1.63
X13-X12	316.6	9.4	1.74
X9-X12	322.0	12.1	1.71
G9-X12	66.6	8.5	1.50
X10-X13	272.4	15.1	1.65
X12-X13	332.3	19.0	1.66
X11-X13	102.1	13.1	1.47
G10-X13	76.8	14.7	1.30

KRAKSIM 2006 natężenie 500			
link	$\overline{t}$	$\sigma(\overline{t})$	$\overline{v}$
X5-X1	193,8	27,1	1,29
G1-X1	110,6	28,8	0,90
X2-X1	191,5	30,1	1,31
X1-X2	242,0	41,7	1,03
G2-X2	160,5	47,2	0,62
X6-X2	302,3	61,2	1,16
X3-X2	203,7	60,4	0,74
G3-X3	135,5	38,0	0,74
X2-X3	174,7	51,2	0,86
X6-X3	214,4	49,0	1,17
G4-X3	135,4	36,9	0,74
X1-X5	172,7	21,1	1,45
G6-X5	89,0	19,5	1,12
X8-X5	289,7	23,2	1,55
X2-X6	503,2	590,4	0,70
X3-X6	333,9	363,9	0,75
X7-X6	244,4	185,5	0,61
X6-X7	293,6	209,9	0,51
X8-X7	931,9	1015,9	0,70
X9-X7	312,6	251,1	0,64
G7-X8	95,8	23,3	1,04
X7-X8	432,3	50,6	1,50
X5-X8	296,8	26,7	1,52
X10-X8	240,2	26,9	1,46
X12-X9	546,9	132,3	1,01
X7-X9	499,8	203,2	0,40
X11-X9	364,7	118,6	0,69
G5-X9	646,4	345,0	1,01
X8-X10	233,7	24,2	1,50
G8-X10	89,7	20,6	1,11
X11-X10	338,4	46,7	1,48
X13-X10	296,4	28,3	1,52
X13-X11	112,4	20,0	1,34
X10-X11	315,1	26,3	1,59
X9-X11	160,1	19,1	1,56
X13-X12	342,9	26,9	1,60
G9-X12	74,5	12,9	1,34
X9-X12	331,9	23,8	1,66
X11-X13	125,1	31,5	1,20
X10-X13	291,8	27,2	1,54
X12-X13	362,6	35,9	1,52
G10-X13	85,5	18,6	1,17

Tablica 5.4. Pomiary dla każdego połączenia; algorytm SOTL, długość stanu przejściowego 8, model standardowy, natężenie ruchu zapisane w nagłówku; połączenia posortowane wg ich końca - grupowane są połączenia zbiegające się w skrzyżowaniu.

natężenie ruchu: długość sygnału żółtego	KRAKSIM 2006	KRAKSIM 2007
	ilość tur (średnia prędkość)	
300:8	44234 (1,50)	44296 (1,62)
500:8	46795 (1,03)	44386 (1,52)
500:4	44489 (1,44)	44321 (1.58)

Tablica 5.5. Porównanie pomiarów o charakterze globalnym dla dwóch wersji systemu KRAKSIM; moduł oceny SOTL.

natężenie ruchu: długość sygnału żółtego	KRAKSIM 2006	KRAKSIM 2007
	ilość tur (średnia prędkość)	
300:8	44257 (1,38)	44362 (1,44)
500:8	46682 (1,05)	44664 (1,37)
500:4	44531 (1,28)	44309 (1.47)

Tablica 5.6. Porównanie pomiarów o charakterze globalnym dla dwóch wersji systemu KRAKSIM; moduł oceny RL.

zapalenie zielonego światła dla grupy strumieni określonej w konfiguracji skrzyżowania. Dzięki temu, że równocześnie przez skrzyżowanie mogą przejeżdżać pojazdy w różnych kierunkach (niekolidujących ze sobą), przepustowość skrzyżowania zwiększa się. Również dzięki temu można zbudować realistyczny plan fazowy.

Tablica 5.4 na stronie 106 pokazuje szczegółowe zestawienie wyników dla tego eksperymentu przeprowadzonego w poprzedniej wersji systemu dla dużego natężenia ruchu – 500 pojazdów na każdej trasie – oraz w aktualnej wersji systemu dla tego samego natężenia ruchu. Trzecia tabela prezentuje wyniki dla nowej wersji systemu i natężenia ruchu wynoszącego 600 pojazdów na każdą trasę. Dla tego samego natężenia ruchu średnie prędkości w nowej wersji systemu są o wiele większe. Wynika to bezpośrednio ze zmian sterownika sygnalizacji. Na najbardziej zatłoczonych trasach (np.: X6-X7, X3-X2) średnia prędkość wzrosła ponad dwukrotnie. Na trasach, na których ruch był luźny, średnia prędkość wzrosła tylko nieznacznie. Wynika to z ograniczeń prędkości oraz z tego, że te trasy są długie i czas przejazdu jest relatywnie długi w porównaniu do czasu oczekiwania na czerwonym świetle. Zmiany czasu trwania czerwonego światła mają niewielkie znaczenie na tych trasach, gdy ruch jest luźny.

Można porównać wyniki tego testu dla systemu KRAKSIM 2006 przy natężeniu 500 pojazdów na trasę z wynikami dla nowego systemu, przy natężeniu wynoszącym 600 pojazdów na trasę. Jest to poziom natężenia, który doprowadza obydwa systemy

do granicy przepustowości modelu. W przypadku dróg o niewielkim natężeniu ruchu, sytuacja w nowym systemie jest lepsza, natomiast dla dróg najbardziej zatłoczonych średnie prędkości są niższe. Ponieważ wszystkie skrzyżowania mają większą przepustowość, to z tych o niewielkim zatłoczeniu pojazdy szybciej wyjeżdżają, równocześnie pojazdy szybciej dojeżdżają do centrum (np.: najbardziej zatłoczona droga w centrum X6-X7). Dużo większa ilość pojazdów w centrum sprawia, że jest ono bardziej zatłoczone.

5.2.3. Modyfikacja sieci drogowej

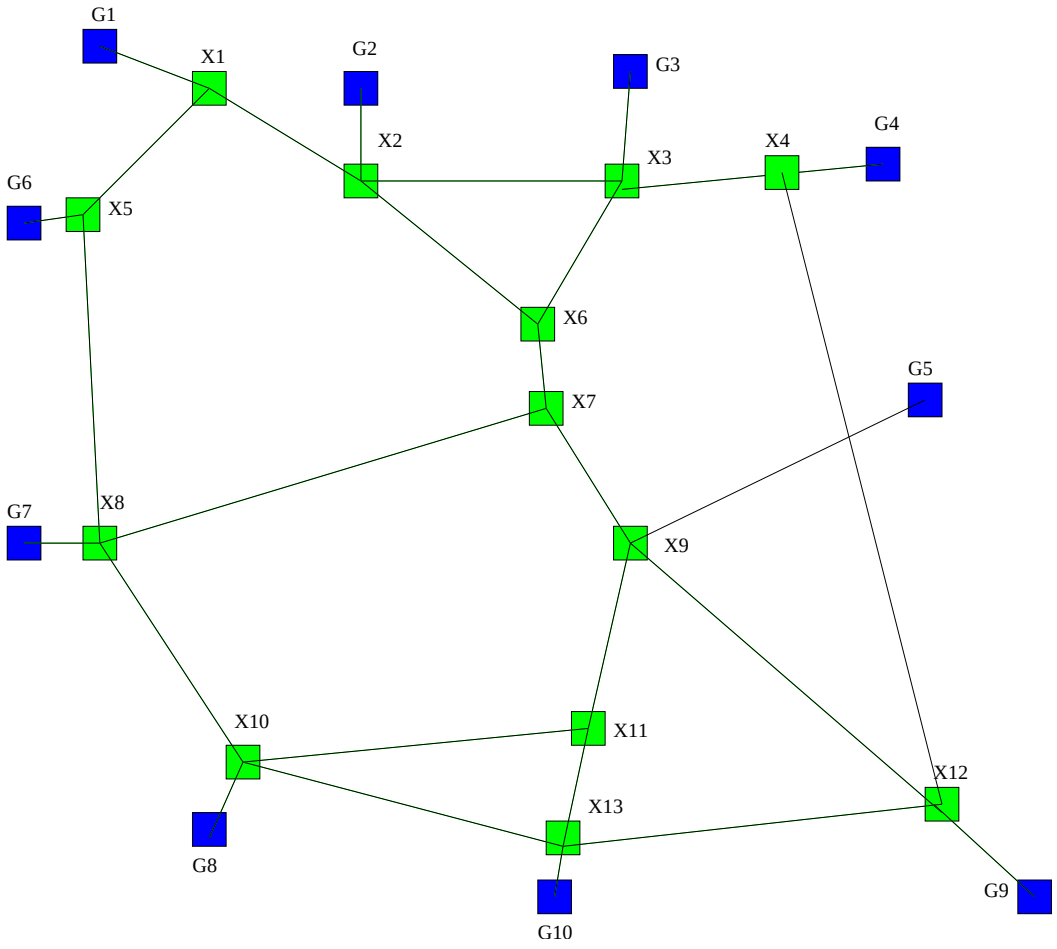
Celem tego eksperymentu było zbadanie zysku po modyfikacji sieci drogowej modelu. Do modelu dodano nowe skrzyżowanie X4 (pomiędzy X3 oraz G4) i nową drogę łączącą to skrzyżowanie ze skrzyżowaniem X12 (patrz model rozszerzony, rys. 5.15).

Model: Do eksperymentu użyto trzech modeli: model standardowy (patrz. 5.12 w rozdziale 5.2 na stronie 100), model rozszerzony z trasą X4-X12 o długości 1050 komórek(rys. 5.15) oraz model rozszerzony z trasą X4-X12 o długości 1000 komórek. Funkcjonalność dynamicznej zmiany trasy przez kierowców była wyłączona, aby uprościć analizę wyników.

Schemat ruchu: Schematu ruchu opisano w rozdziale 5.2. Dokonano testów dla natężenia ruchu wynoszącego 650 pojazdów na każdą trasę w trakcie całej symulacji.

Testy: Przeprowadzony został jeden eksperyment na modelu wyjściowym oraz jeden na modelu zmodyfikowanym. Zauważono, że bardzo mała liczba pojazdów wybiera nową trasę, co jest spowodowane tym, że router w podstawowej konfiguracji wybiera trasę tylko na podstawie jej długości. Dlatego, aby zmienić przebieg planowanych tras, wprowadzono kolejną wersję modelu rozszerzonego. Nowy model miasta różni się od modelu rozszerzonego tylko długością trasy X4-X12. Aby *zachęcić* router do wyznaczania podróży przez nową drogę, zmniejszono jej długość z 1050 komórek do 1000 komórek.

Analiza wyników i wnioski: Tablica 5.7 zawiera podsumowanie wyników tego eksperymentu. Jak można zauważyć wprowadzenie nowej drogi w odpowiednim miejscu w sieci może znacząco poprawić sytuację drogową. Średnia prędkość wzrosła



Rysunek 5.15. Rozszerzony model miasta

	model miasta		
	standardowy	rozszerzony	rozszerzony 2
długość symulacji [s]	51416	46220	44297
średnia prędkość [k/s (km/h)]	0,57 (15,4)	1,09 (29,4)	1,57 (42,4)

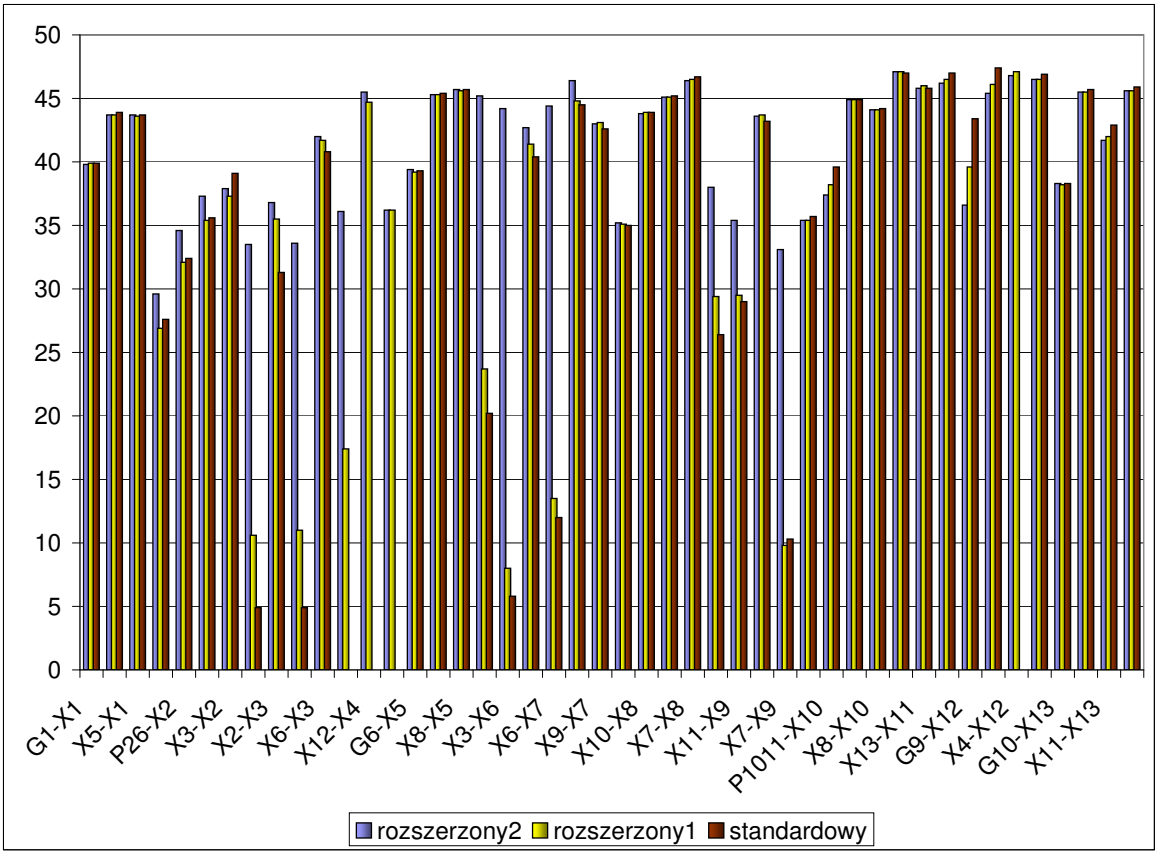
Tablica 5.7. Porównanie pomiarów o charakterze globalnym dla standardowego i rozszerzonego modelu miasta; moduł oceny SOTL, długość stanu przejściowego: 4, natężenie ruchu: 650 pojazdów na każdą trasę.

droga	średnia prędkość [k/s (km/h)]		
	model standardowy	model rozszerzony	model rozszerzony 2
P26-X6[a]	0,75 (20.2)	0,88 (23.7)	1,68 (45.2)
X4-X3[b]	0,18 (4.9)	0,41 (11)	1,24 (33.6)
G3-X3	0,18 (4.9)	0,39 (10.06)	1,24 (33.6)
X3-X6	0,22 (5.8)	0,30 (8)	1,64 (44.2)
X6-X7	0,45 (12)	0,5 (13.5)	1,65 (44.4)
X7-X9	0,38 (10.3)	0,36 (9.8)	1,22 (33.1)

[a] P26 jest punktem postawionym na trasie pomiędzy X2 i X6 w celu zachowania geometrii mapy podobnej do rzeczywistości. Nie wpływa on na cechy modelu a jedynie na sposób prezentacji wizualnej oraz, przez podział tego odcinka na dwie części, na wielkość wartości średnich.

[b] dla modelu standardowego jest to G4-X3

Tablica 5.8. Porównanie pomiarów o charakterze lokalnym dla kilku najbardziej zatłoczonych dróg standardowego oraz rozszerzonego modelu miasta; moduł oceny SOTL, długość stanu przejściowego: 4, natężenie ruchu: 650 pojazdów na każdą trasę.



Rysunek 5.16. Średnia prędkość przejazdu przez konkretne połączenia.

z 0,57 k/t[5] do 1,09 k/t, czyli niemal dwukrotnie. Równocześnie czas trwania symulacji skrócił się o 5196 tur, czyli o około 1 godzinę i 26 minut (z 51416 do 46220 tur). 2600 pojazdów skorzystało z nowej drogi, odciążając tym samym *centrum* miasta. Trzeci test, przeprowadzony na modelu, gdzie droga X4-X12 ma długość 1000 komórek, dał jeszcze lepsze wyniki. W tym przypadku z nowej drogi skorzystało 5200 pojazdów (jest to ok 8% wszystkich pojazdów), średnia prędkość wzrosła aż trzykrotnie (z 0,57 do 1,57 k/t), a czas trwania symulacji skrócił się o 7119 tur, czyli o prawie 2 godziny.

Szczegółowy wykres wartości średniej prędkości dla wszystkich dróg w sieci (pominięto tylko odcinki wyjazdowe, które są mało istotne dla tej analizy) dla tych trzech modeli zaprezentowano na rys. 5.16. Pozwala on zauważyć, że wąskim gardłem tego modelu są drogi: G3-X3, X4-X3, X2-X6, X3-X6, X6-X7, X7-X9. Droga prowadząca przez X4-X12 jest szybszą alternatywą dla trasy wiodącej przez *centrum* — X3-X6-X7-X9. Na kilku połączeniach średnia prędkość spadła (np. G9-X12), jednak są to niewielkie wahania, które nie zmieniają stanu ruchu drogowego.

Tablica 5.8 zawiera szczegółowe informacje dotyczące średniej prędkości na najbardziej zatłoczonych drogach modelu.

5.3. Różne metody sterowania sygnalizacją

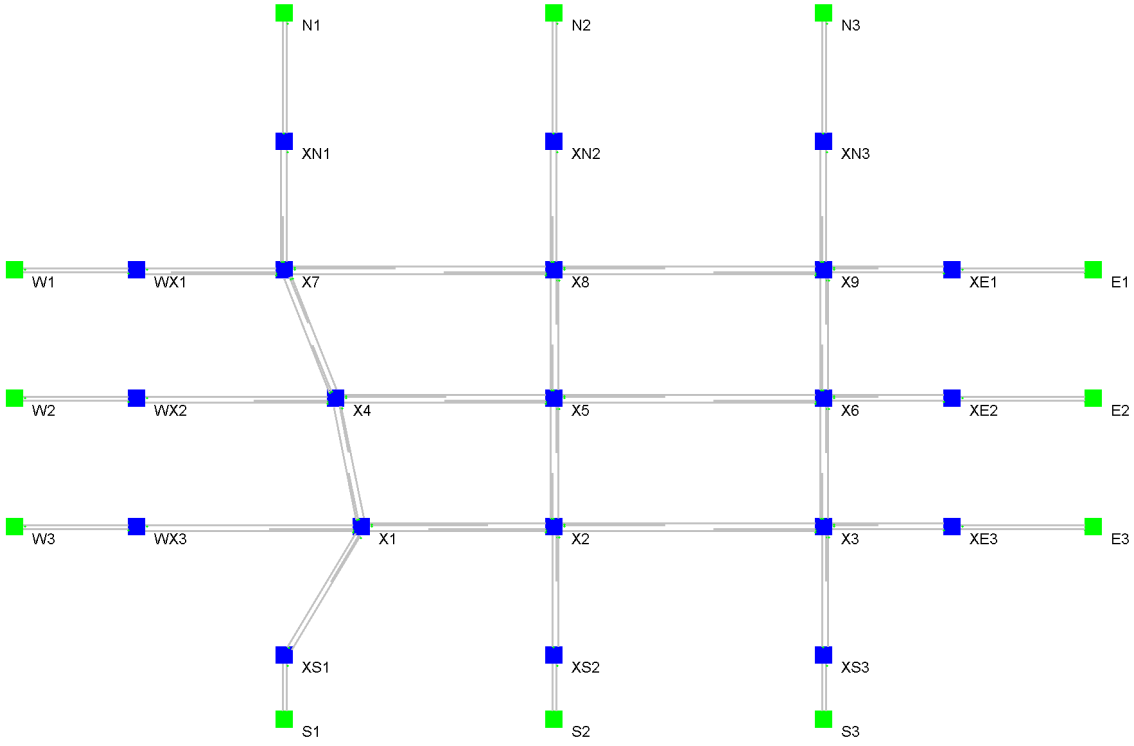
W celu porównania efektywności różnych metod sterowania sygnalizacją świetlną przeprowadzono eksperyment, w którym użyty został układ 9-ciu skrzyżowań z ustalonym planem ruchu. Na tym modelu przetestowano działanie różnych metod sterowania: sygnalizacja ustalona, sygnalizacja z dynamiczną synchronizacją[6] kierunku priorytetowego, sygnalizacja acykliczna sterowana metodą SOTL, sygnalizacja acykliczna sterowana metodą RL.

Model: Eksperyment przeprowadzany jest na kratowym (skrzyżowania dróg prostopadłych, stąd nazwa – manhattan) modelu dróg ze skrzyżowaniami, który jest przedstawiony na rys. 5.17. W modelu jest 12 węzłów wejścia/wyjścia oraz 9 skrzyżowań. Długość każdej drogi pomiędzy dwoma skrzyżowaniami wynosi 50 komórek (375 metrów). Wyjątek stanowią drogi X1-X2 oraz X4-X5, których długości to odpowiednio 30 komórek i 40 komórek. Dzięki temu planer tras zazwyczaj wybiera właśnie te drogi (gdy bierze pod uwagę koszt częściowy wynikający z długości rozważanej trasy). Każde

[5] komórki na turę

[6] gdzie kierunki synchronizacji wybierane są za pomocą agentowej metody mediacji

skrzyżowanie posiada pasy do skrętu w lewo o długości 20 komórek.



Rysunek 5.17. Model manhattan.

Schemat ruchu: Schemat ruchu dla tego eksperymentu składa się z 10-ciu strumieni. Przygotowano 3 wersje tego schematu.

- Pierwsza wersja (tab. 5.9) to schemat, w którym większość ruchu odbywa się na osi wschód-zachód.

	kierunki WE			kierunki NS	
	$W1 \leftrightarrow E1$	$W2 \leftrightarrow E2$	$W3 \leftrightarrow E3$	$N1 \leftrightarrow S1$	$N2 \leftrightarrow S2$
natężenie ruchu [pojazdów/h]	400	400	400	100	100

Tablica 5.9. Schemat ruchu nr 1, dominujące strumienie wschód-zachód.

- Drugi plan (tab. 5.10) zaprojektowano tak, że ruch o głównym natężeniu odbywa się na osi północ-południe. Taka konfiguracja pozwoli na pokazanie różnic w zachowaniu systemu dla różnych metod sterowania sygnalizacją i przy różnym charakterze ruchu.

	kierunki WE			kierunki NS	
	$W1 \leftrightarrow E1$	$W2 \leftrightarrow E2$	$W3 \leftrightarrow E3$	$N1 \leftrightarrow S1$	$N2 \leftrightarrow S2$
natężenie ruchu [pojazdów/h]	100	100	100	400	400

Tablica 5.10. Schemat ruchu nr 2, dominujące strumienie północ-południe.

- Trzecia wersja planu (tab. 5.11) ma zmienną charakterystykę w czasie. W połowie czasu zaplanowanego na symulację pojawiają się nowe strumienie ruchu. Ruch na trasach W1-E1, E1-W1, W2-E2, E2-W2, W3-E3 i E3-W3 jest zaplanowany na 300 pojazdów od początku eksperymentu do jego końca godzinę później (3600 tur symulacji), co daje 300 pojazdów na godzinę. Również od początku symulacji generowane są pojazdy z trasami przejazdu S2-N2 oraz N2-S2. Jest to ruch prostopadły do poprzedniego i ma natężenie 200 pojazdów na godzinę.
  Po upływie 30 minut symulacji, pojawia się ruch na trasie N1-S1, S1-N1 o natężeniu 200 pojazdów na godzinę (przez 30 minut), oraz ruch na trasie N1-S3, S3-N1 o natężeniu 200 pojazdów na godzinę.
  Przez pierwsze 30 minut, jest to ruch z dominującym kierunkiem wschód-zachód na wszystkich skrzyżowaniach. Po wprowadzeniu nowego strumienia ruchu na skrzyżowaniach X7, X4 oraz X1 ruch w prostopadłych kierunkach jest w miarę równomiernie rozłożony (z kierunków wschód-zachód ruch ma natężenie 300 pojazdów na godzinę, a z kierunków północ-południe 400 pojazdów na godzinę). Jako główny można wyróżnić kierunek północ-południe. Przez skrzyżowania X2 oraz X3, w kierunkach wschód oraz zachód, prowadzi trasa połączonych strumieni N1-S3 i W3-E3, co sprawia że wyraźnie dominuje ruch w tym kierunku (wschód-zachód).

	kierunki WE			kierunki NS startujące z opóźnieniem		kierunki NS
	$W1 \leftrightarrow E1$	$W2 \leftrightarrow E2$	$W3 \leftrightarrow E3$	$N1 \leftrightarrow S1$	$N1 \leftrightarrow S3$	$N2 \leftrightarrow S2$
natężenie ruchu [pojazdów/h]	300	300	300	200	200	200

Tablica 5.11. Schemat ruchu nr 3, charakter ruchu zmienny, trudno wyróżnić globalny kierunek dominujący.

Ten schemat ma sprawdzić jak zachowują się różne systemy sterowania przy zmianie charakterystyki ruchu. Oraz w przypadku ruchu w dwóch prostopadłych do siebie kierunkach, gdy trudno jest wybrać jeden kierunek priorytetowy.

Testy: Przeprowadzono symulacje ruchu na modelu kratowym, dla następujących metod sterowania:

A. sygnalizacja ustalona,
B. sygnalizacja z dynamiczną synchronizacją,
C. sygnalizacja acykliczna sterowana metodą SOTL,
D. sygnalizacja acykliczna sterowana metodą RL.

Wyniki pomiarów uzyskane podczas tych symulacji przedstawia tablica 5.12. Przeprowadzono 3 eksperymenty:

- ruch o większym natężeniu w kierunku WE
- ruch o większym natężeniu w kierunku NS
- ruch zróżnicowany, ze zmianami natężenia dla poszczególnych strumieni ruchu w trakcie trwania symulacji.

	czas trwania symulacji (średnia prędkość)		
metoda sterowania	Ruch WE	Ruch NS	Ruch zróżnicowany
A	5850 (0,27)	3994 (0,98)	6837 (0,22)
B	3939 (1,03)	3914 (1,01)	6708 (0,47)
C	3754 (1,63)	3751 (1,65)	3831 (1.52)
D	3848 (1,57)	3766 (1,62)	3838 (1.44)

Tablica 5.12. Porównanie pomiarów o charakterze globalnym dla czterech metod sterowania sygnalizacją świetlną na skrzyżowaniach w systemie KRAKSIM.

Analiza wyników i wnioski: Synchronizacja ustalona działa optymalnie tylko dla warunków dla jakich została zaprojektowana. W tym eksperymencie sygnalizacja świetlna została ustalona dla ruchu o znacznie większym natężeniu na kierunku północ–południe. Średnia prędkość w modelu, gdy ruch o głównym natężeniu przebiega zgodnie z kierunkiem ustawienia sygnalizacji, wynosi 0,98 k/t (w tab. 5.12 warto porównać również czasy trwania kolejnych testów). Jednak gdy główny kierunek ruchu to wschód–zachód lub gdy ruch jest zmienny i zróżnicowany, to średnia prędkość wynosi odpowiednio 0,27 dla ruchu WE oraz 0,22 k/t ruchu zróżnicowanego.

W przypadku sygnalizacji z dynamiczną synchronizacją kierunku priorytetowego, sytuacja jest o wiele lepsza. W pierwszym i drugim eksperymencie średnia prędkość wynosi odpowiednio 1,03 k/t oraz 1,01 k/t. Jest to wynik podobny do optymalnego dla sygnalizacji ustalonej na stałe, gdy główny kierunek ruchu jest zgodny z zaplanowanym. W tym przypadku nie ma problemu zgodności kierunku zaplanowanego z tym występującym w eksperymencie, ponieważ synchronizacja ustalana jest dynamicznie

na podstawie danych z detektorów ruchu. Trzeci eksperyment dla tej metody sterowania zakończył się z wynikiem 0,47 k/t. Jest prędkość o wiele niższa niż w pierwszych dwóch testach, ale równocześnie dwukrotnie wyższa (lepsza) niż w tym samym eksperymencie dla metody z sygnalizacją ustaloną. Można się domyślać, że spowodowane jest to okresami adaptacji, gdy sygnalizacja nie pracuje optymalnie, tzn. gdy występują zmiany w charakterystyce ruchu i cały system musi się do tego dostosować.

Serie eksperymentów przeprowadzone dla metod autonomicznych z sygnalizacją acykliczną są do siebie bardzo podobne. Obydwie acykliczne metody sterowania sygnalizacją (SOTL, oraz RL) dają lepsze rezultaty niż poprzednio badane metody cykliczne. Różnice pomiędzy kolejnymi eksperymentami są nieznaczne i we wszystkich średnia prędkość jest wyższa niż 1,44 k/t. Warto zauważyć, że metody te radzą sobie dobrze również ze zróżnicowanym i zmiennym ruchem w trzecim teście. Główną przewagą tych dwóch metod jest właśnie acykliczność sterownika sygnalizacji. Długość poszczególnych sygnałów jest dynamicznie dostosowywana do aktualnych warunków i jeżeli nie ma potrzeby, to sygnał może zostać pominięty[7] (to daje dużą oszczędność czasu).

5.4. Wpływ systemu informacji o korkach

Celem tego eksperymentu jest zbadanie wpływu zaimplementowanego prototypu systemu informacji o korkach na sytuację drogową w mieście. Prototyp działa na zasadzie rozgłośni radiowej (nie zaimplementowano systemu znaków o zmiennej treści), gdzie informacje dostępne są globalnie.

Ponieważ nie wszyscy kierowcy słuchają radia oraz nie wszyscy są skłonni zmienić trasę pod wpływem takich informacji, wprowadzono dwa parametry: k – określający procentową ilość kierowców skłonnych do zmiany trasy oraz d – określający prawdopodobieństwo zmiany trasy przez kierowcę. Sam system informacyjny może serwować informacje o różnym poziomie granularności (średnia z ostatniej minuty, średnia z ostatnich XX minut). Poprzez parametr u można określić przedział czasu w jakim dane są uaktualniane (w milisekundach).

[7] Ze względów bezpieczeństwa pomijany sygnał powinien zostać wyemitowany co ustalony okres. Zabezpieczy to przed chaotycznym działaniem sygnalizacji w przypadku awarii detektorów. Ta funkcjonalność nie została zaimplementowana, dlatego prezentowane tu wyniki przedstawiają idealny teoretyczny system. Szczegółowe zasady mogą być narzucone przez przepisy obowiązujące w regionie dla którego przeprowadza się symulację

5.4.1. Kratowy model sieci – *manhattan*

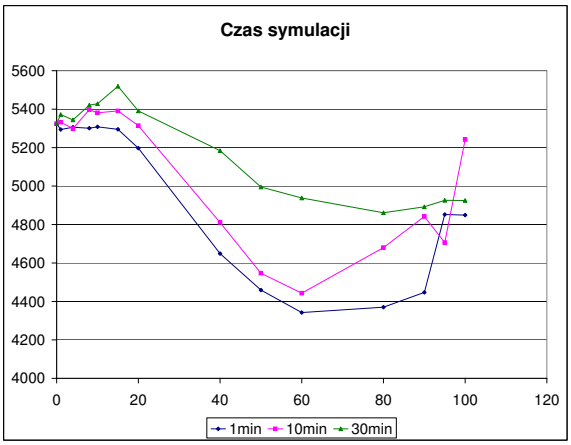
Model: Eksperyment przeprowadzany jest na kratowym modelu dróg ze skrzyżowaniami (patrz 5.3 na stronie 111).

Schemat ruchu: Schemat ruchu dla tego modelu został zaplanowany tak, aby zakorkować ruch w południowej części (przez skrzyżowania X1, X2, X3).

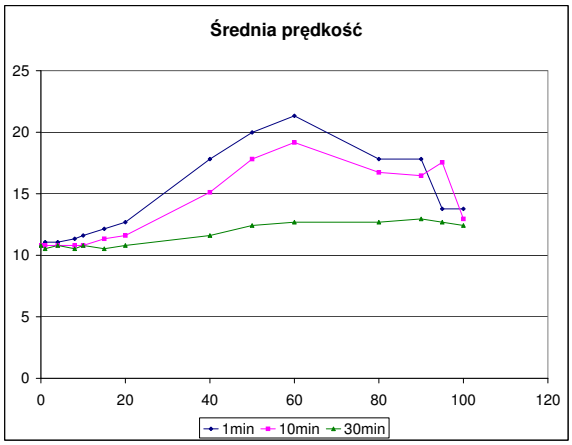
Główny ruch w tym schemacie przebiega z węzła W1 do węzła S3 najkrótszą trasą czyli: W1-X7-X4-X1-X2-X3-S3. Pojazdy pojawiają się co 3 sekundy przez okres jednej godziny (jest to 1200 pojazdów na godzinę). Po upływie 500 sekund (ponad 8 minut) zaczyna się ruch o dużym natężeniu na trasie z W3 do E3. Po upływie 1500 sekund od rozpoczęcia eksperymentu (25 minut) wzmaga się ruch na trasie z W2 do E2.

Strumienie ruchu, które pojawiają się później, mają za zadanie przeszkodzić kierowcom jadącym z W1 do S3 poprzez skrzyżowanie X1. Jeżeli kierowcy nie wiedzą o zatorze, który się wtedy zaczyna tworzyć, to nie zmieniają swojej trasy.

Testy: Poniżej wyniki eksperymentu przeprowadzone dla wartości u wynoszącej 1, 10 oraz 30 minut i różnych wartości k od 0% do 100%.



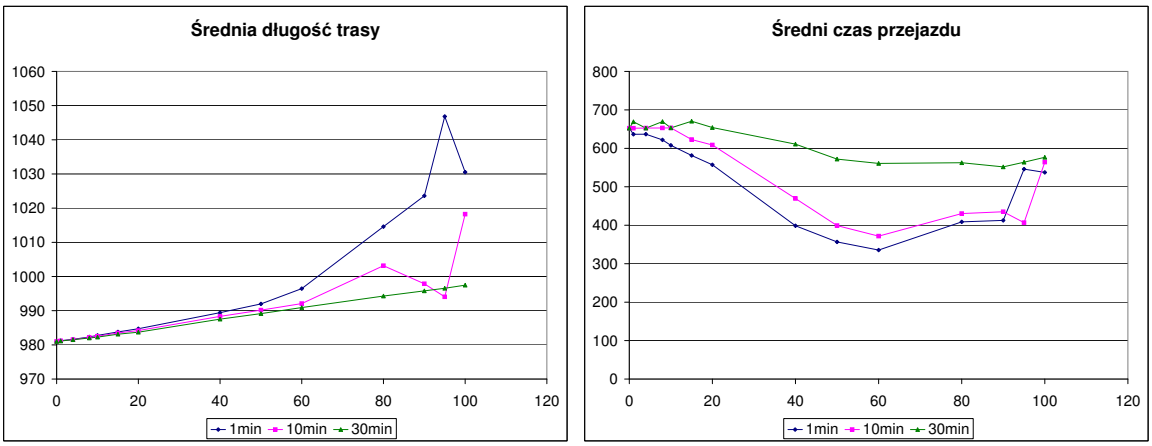
(a) czas trwania symulacji



(b) średnia prędkość

Rysunek 5.18. System informacji o zatłoczeniach: średnia prędkość i czas symulacji dla modelu manhattan. Na osi X procent kierowców reagujących na informacje zmianą trasy.

Analiza wyników i wnioski: Wszystkie próby pokazują, że aktualność danych z systemu informacyjnego ma bardzo duże znaczenie dla sytuacji drogowej. Średnia



(a) średnia długość trasy

(b) średni czas podróży

Rysunek 5.19. System informacji o zatłoczeniach: średni czas podróży i średnia długość podróży dla modelu manhattan. Na osi X procent kierowców reagujących na informacje zmianą trasy.

prędkość oraz przepustowość w testach z czasem aktualizacji wynoszącym jedną minutę jest wyższa niż w pozostałych (patrz 5.18(b)).

Analizując zależność pomiędzy ilością kierowców, którzy decydują się na zmianę trasy, a średnią prędkością i średnim czasem przejazdu (rys. 5.18(b),5.19(b)), zauważono pewną prawidłowość. Wzrost wartości k od 0% do pewnego poziomu powoduje wyraźną poprawę charakterystyki ruchu. W tym eksperymencie wartość graniczna leży pomiędzy 60% i 80%. Średni czas przejazdu spada z około 650s, gdy żaden kierowca nie zmienia trasy, do ok. 350s, gdy 60% kierowców wybiera objazd. To są wyniki dla czasu odświeżania danych 1 minuta. Podobnie jest dla 10 minut.

Średnia długość trasy zwiększa się o ok. 5%, a więc kierowcy wybierają dłuższe trasy, ale dzięki szybszemu przejazdowi całkowity czas podróży znacznie się skraca. Wzrost długości trasy jedynie o ok. 5% przy równoczesnym skróceniu czasu przejazdu o 50% to bardzo dobry wynik. Natomiast przy 30 minutowym czasie agregacji danych poprawa nie jest już tak duża, czas przejazdu spada do ok. 550s (daje to jedynie 15% poprawę).

Wzrost wartości k powyżej poziomu 80% powoduje pogorszenie badanych charakterystyk. Średni czas przejazdu w dalszym ciągu jest krótszy niż początkowe 650 sekund, ale zwiększa się długość trasy. Najbardziej zauważalne jest to dla danych o rozdzielczości jednej minuty, gdzie średnia długość trasy wynosi ok. 980, 995, 1015 oraz 1048 komórek odpowiednio dla liczby kierowców wybierających objazdy równej 0%, 60%, 80% i 95%. Jest to wzrost długości trasy o ok. 10%, przy zysku czasu przejazdu na

poziomie od 15 do 40%, a więc wynik w dalszym ciągu dobry[8]. Dane dla informacji z ostatnich 30-minut pokazują mniejsze zmiany głównych charakterystyk. Gdy 90% kierowców może wybierać objazdy, to średnia długość trasy rośnie zaledwie o ok. 2%, a czas przejazdu spada o 15%, co jest dosyć dobrym wynikiem.

Spadek efektywności systemu drogowego powyżej poziomu 80% może być spowodowany tym, że w przypadku wysłania informacji o zatorze na ulicy A, kierowcy decydują się na objazd ulicą B, co skutkuje zatorem na ulicy B. Częsta zmiana trasy i szukanie objazdów powodują wydłużenie długości trasy. To nawet przy większej średniej prędkości może wydłużyć czas przejazdu (rys. 5.19(a)).

Idealna sytuacja ma miejsce wtedy, gdy tylko część kierowców decyduje się na objazd. W tym przypadku pierwotnie planowana trasa przejazdu zostaje odciążona i zator się nie powiększa, a trasa alternatywna nie zostanie zakorkowana.

5.4.2. Uproszczony model miasta

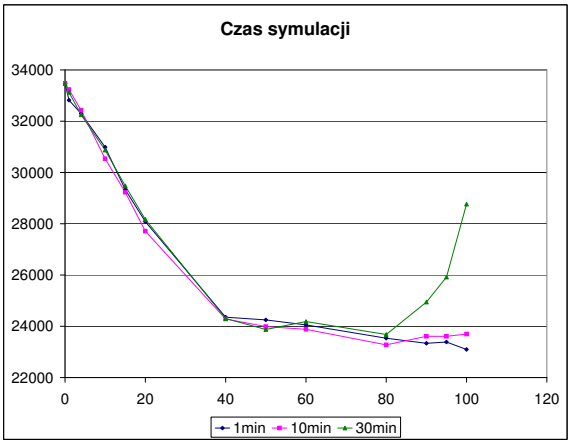
Model: Eksperyment przeprowadzany jest na uproszczonym modelu miasta. W modelu jest 10 węzłów wejścia/wyjścia oraz 13 skrzyżowań. Układ dróg jest modelem głównych arterii miasta Kraków wraz z obwodnicą oraz jedną drogą dodatkową, która pozwala ominąć centrum od wschodu (patrz model 5.15 na stronie 109). Układ dróg sprawia, że dla większości tras najkrótsza ścieżka prowadzi przez centrum. Przy dużym natężeniu ruchu łatwo dochodzi do utworzenia zatoru w rejonie centrum (drogi X9-X7-X6).

Schemat ruchu: Schemat ruchu dla tego modelu zawiera 90 tras o lekkim natężeniu (400 pojazdów na 6 godzin) pomiędzy każdą parą węzłów oraz 7 tras o średnim natężeniu (od 400 pojazdów na 5,5h do 500 pojazdów na 2 godziny). Cały plan ustalony jest na ok. 6 godzin. Na wszystkie trasy podstawowe ruch generowany jest z jednostajnym rozkładem prawdopodobieństwa od początku symulacji przez 6 godzin. 7 tras dodatkowych zaplanowano dla następujących węzłów źródło-cel: G2-G9, G3-G9, G3-G10, G6-G10, G8-G2, G9-G2, G10-G6. Ruch na tych trasach zaczyna się z opóźnieniem (aby wprowadzić zmiany do ustabilizowanego już systemu i sprawdzić reakcję systemu informacji oraz reakcję kierowców na dane z tego systemu). Wprowadzenie tego dodatkowego ruchu powoduje całkowite zablokowanie centrum, gdy jako kryte-

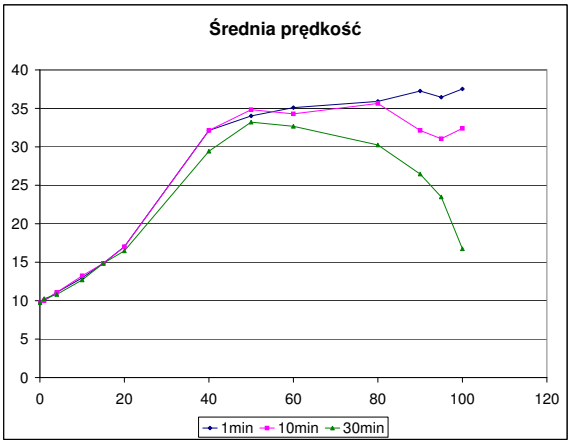
[8] Gdyby badać charakterystyki ekonomiczne (średnie spalanie pojazdów) oraz ekologiczne mogłoby się okazać, że jest to próg opłacalności.

rium wyboru trasy stosowana jest jej długość.

Testy: Poniżej wyniki eksperymentu przeprowadzone dla wartości u wynoszącej 1, 10 oraz 30 minut i różnych wartości k od 0% do 100%.

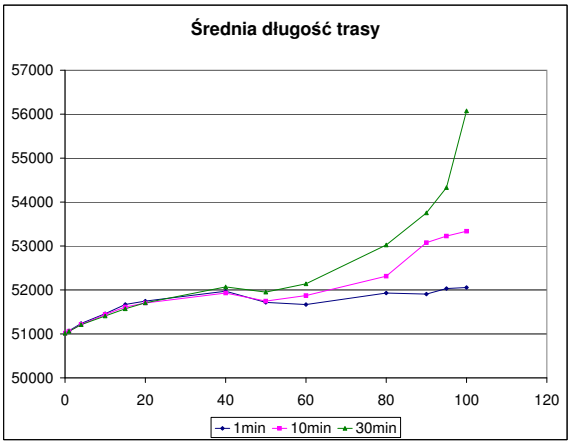


(a) czas trwania symulacji

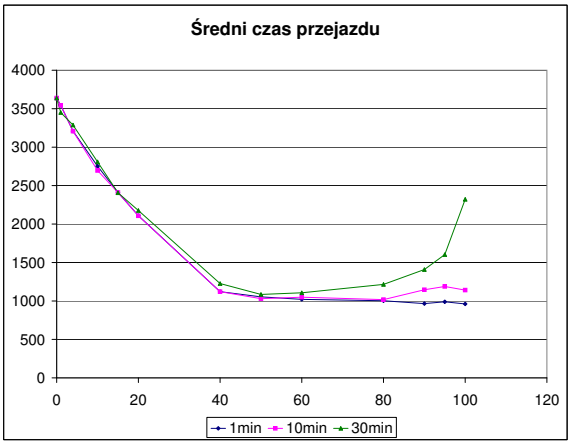


(b) średnia prędkość

Rysunek 5.20. System informacji o zatłoczeniach: średnia prędkość i czas symulacji dla modelu miasta. Na osi X procent kierowców reagujących na informacje zmianą trasy.



(a) średnia długość podróży



(b) średni czas przejazdu

Rysunek 5.21. System informacji o zatłoczeniach: średni czas podróży i jej długość dla modelu miasta. Na osi X procent kierowców reagujących na informacje zmianą trasy.

Analiza wyników i wnioski: Testy przeprowadzone na modelu miasta potwierdzają wyniki uzyskane już na modelu kratowym. Również tutaj można zauważyć, że

ilość kierowców, którzy decydują się na zmianę trasy ma bezpośredni wpływ na stan ruchu drogowego w modelu. Tutaj optymalny przedział wartości określającej procentową ilość kierowców zmieniających pierwotnie obraną trasę jest szerszy niż w poprzednim eksperymencie i wynosi od ok. 40% do ok. 85% (patrz rys. 5.20, 5.21). W tym przedziale czas symulacji (to znaczy czas, w którym wszystkie pojazdy przejadą swoje trasy) skrócił się z ok. 9-ciu do ok. 6-ciu godzin. Jest to poprawa o ponad 30%.

Jeżeli chodzi o wpływ aktualności danych informacyjnych na stan systemu, to podobnie jak w poprzednim eksperymencie im aktualniejsze dane, tym lepsze jest zachowanie systemu. Dla przedziałów o wielkości 1 minuta oraz 10 minut charakterystyki pozostają dobre w zakresie od 40% do 100%, natomiast dla przedziału 30-sto minutowego po przekroczeniu 85% widać spadek zysków. Gdy wszyscy kierowcy mogą wybierać objazdy, średnia długość trasy rośnie o 12% (z 51000 komórek do 56000 komórek), czas przejazdu spada o 32%. Jest to wynik gorszy niż dla informacji aktualizowanych co minutę, jednak w dalszym ciągu dobry.

Analizując wyniki z obydwu eksperymentów dla systemu informacji o zatorach, stwierdza się, że nie można wyznaczyć jednoznacznych wartości optymalnych dla ilości pojazdów szukających objazdu. Zależy to od dokładności i aktualności informacji drogowych oraz od topologii sieci drogowej i charakterystyki ruchu, a więc jest specyficzne dla każdego miasta. Wprowadzenie takiego systemu przyniesie korzyści pod warunkiem, że informacje będą skierowane do odpowiednich grup kierowców i kierowcy tym informacjom zaufają. Pewnym problemem może być sytuacja, kiedy zbyt dużo kierowców wybierze objazd.

Rozdział 6

Podsumowanie

Zaczynając prace dysponowano prototypem systemu do modelowania i optymalizacji ruchu drogowego „KRAKSIM". W ramach prowadzonej pracy magisterskiej udało się znacznie poprawić poziom realizmu w rozwijanym systemie symulacji.

Osiągnięte rezultaty są wystarczające, aby powiedzieć, że cele stawiane tej pracy dyplomowej zostały osiągnięte.

System sterowania sygnalizacją świetlną pozwalał na ruch tylko jednego strumienia jednocześnie oraz nie miał sterowania na podstawie zegara (nie można bez tego zbudować prostego sterownika cyklicznego). Sterownik sygnalizacji świetlnej został poprawiony: w konfiguracji skrzyżowania definiuje się, na których kierunkach równocześnie może zapalić się zielone światło, co pozwala na równoczesne przepuszczenie przez skrzyżowanie kilku strumieni ruchu. Do systemu dodane zostały nowe metody optymalizacji, w tym: system informacji o sytuacji drogowej, możliwość dynamicznej zmiany trasy przez kierowców, sygnalizacja cykliczna, sygnalizacja z dynamiczną synchronizacja kierunku priorytetowego[1]. Poprawiono również metody sterowania sygnalizacją SOTL oraz RL. Rozdział 3 stanowi opracowanie koncepcji systemu z tymi

[1] sygnalizatory posiadają plany przygotowane dla wzmożonego ruchu w kierunku północ-południe, lub wschód-zachód; na sąsiednich skrzyżowaniach wybierany jest wspólny kierunek priorytetowy i stosowany jest plan przeznaczony dla tego kierunku

wszystkimi poprawkami i rozszerzeniami, a rozdział 4 jest technicznym opisem jego realizacji.

Poprawiony i udoskonalony system został użyty w serii eksperymentów, które razem z wynikami oraz wnioskami również są efektami pracy (cały rozdział 5).

Zaproponowano pewne metody poprawy sytuacji ruchu drogowego w mieście oraz pokazano jak te rozwiązania mogą wpływać na wydajność infrastruktury drogowej w mieście.

W przyszłości projekt można dalej rozszerzać w celu uzyskania profesjonalnego narzędzia modelowania i optymalizacji ruchu drogowego. Proponowane kierunki rozwoju systemu:

- wygodny graficzny edytor topologii sieci drogowej oraz planu podróży,
- wielopasmowy model ruchu z regułami wyprzedzania,
- podział pojazdów na kategorie np.: autobusy miejskie, ciężarówki,
- implementacja znaków o zmiennym tekście,
- implementacja systemu zapobiegania sytuacjom kryzysowym (opisanego w koncepcji).

Bibliografia

[1] A. Adamski. *Inteligentne systemy transportowe: sterowanie, nadzór i zarządzanie*. Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH, 2003.

[2] R. Barlovic, L. Santen, A. Schadschneider, M. Schreckenberg. Metastable states in cellular automata for traffic flow. *European Physical Journal B*, 5:793–800, 1998.

[3] O. Biham, A. A. Middleton, D. Levine. Self-organization and a dynamical transition in traffic-flow models. *Physical Review A*, 46:6124, 11 1992.

[4] Elmar Brockfeld, Robert Barlovic, Andreas Schadschneider, Michael Schreckenberg. Optimizing traffic lights in a cellular automaton model for city traffic. *Phys. Rev. E*, 64(5):056132, 10 2001.

[5] R. Chrobok, T. Grunewald, A. Pottmeier, M. Schreckenberg. Analysis and validation of variable message signs using cellurar automaton traffic simulations. M. Dell'Orco, M. Ottomanelli, redaktorzy, *9th Meeting of the EURO Working Group on Transportation, Intermodality, Sustainability and Intelligent Transportation Systems*, strony 469–472, Bari, 2002.

[6] Carlos Daganzo. The cell transmission model: Network traffic. *California Partners for Advanced Transit and Highways (PATH). Working Papers: Paper UCB-ITS-PWP-94-12.*, 1 1994.

[7] D. de Oliveira, A.L.C. Bazzan, V. Lesser. Using cooperative mediation to coordinate traffic lights: a case study. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, strony 463–470, 2005.

[8] Carlos Gershenson. Self-Orginizing Traffic Lights. *ArXiv Condensed Matter e-prints*, 02 2005.

[9] S.F. Hafstein, R. Chrobok, A. Pottmeier, M. Schreckenberg, F. Mazur. A high-resolution cellular automata traffic simulation model with application in a freeway traffic information system. *Computer-Aided Civil and Infrastructure Engineering*, 19(5):338–350, 2004.

[10] B. S. Kerner. Three-Phase Traffic Theory and Highway Capacity. *ArXiv Condensed Matter e-prints*, 11 2002.

[11] W. Knospe, L. Santen, A. Schadschneider, M. Schreckenberg. Towards a realistic microscopic description of highway traffic. *J. Phys. A: Math. Gen*, 33(48):L477–L485, 2000.

[12] W. Knospe, L. Santen, A. Schadschneider, M. Schreckenberg. A realistic two-lane traffic model for highway traffic. *Journal of Physics A Mathematical General*, 35:3369–3388, 04 2002.

[13] Wolfgang Knospe, Ludger Santen, Andreas Schadschneider, Michael Schreckenberg. Empirical test for cellular automaton models of traffic flow. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(1):016115, 2004.

[14] M. J. Lighthill, G. B. Whitham. On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 229(1178):317–345, 1955.

[15] H.K. Lo, W. Lin, L.C. Liao, E. Chang, J. Tsao. A Comparison Of Traffic Models: Part 1, Framework. Raport instytutowy, Research Report UCB-ITSPRR-96-22. Partners for Advanced Transit and Highways, University of California, Berkeley„ 1996.

[16] Roger Mailler, Victor Lesser. Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems. *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, wolumen 1, strony 446–453, New York, 2004. IEEE Computer Society.

[17] K. Nagel, M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221–2229, 12 1992.

[18] K. Nagel, P. Stretz, M. Pieck, R. Donnelly, C. L. Barrett. TRANSIMS traffic flow characteristics. *ArXiv Condensed Matter e-prints*, strona 10003, 10 1997.

[19] I. Prigogine, R. Herman. *Kinetic theory of vehicular traffic*. American Elsevier, 1971.

[20] P.I. Richards. Shock Waves on the Highway. *Operations Research*, 4(1):42–51, 1956.

[21] D. Robertson, R. Bretherton. Optimizing networks of traffic signals in real time-the SCOOT method. *Vehicular Technology, IEEE Transactions on*, 40(1):11–15, 1991.

[22] A. Schadschneider, D. Chowdhury, E. Brockfeld, K. Klauck, L. Santen, J. Zittartz. A new cellular automata model for city traffic. *Arxiv preprint cond-mat/9911312*, 1999.

[23] M. Wiering, J. van Veenen, J. Vreeken, A. Koopman. Intelligent traffic light control. *ERCIM News, European Research Consortium for Informatics and Mathematics*, 53:40–41, 2003.

[24] M. Wiering, J. Vreeken, J. van Veenen, A. Koopman. Simulation and optimization of traffic in a city. *Intelligent Vehicles Symposium, 2004 IEEE*, strony 453–458, 2004.

Dodatek A

Instrukcja obsługi

A.1. Zawartość dysku CD

Dysk CD dostarczony wraz z pracą dyplomową jako załącznik zawiera:

- plik `readme.txt` — spis zawartości dysku,
- folder `Dependecies` — zależności projektu, co umożliwia szybszą instalację,
  - `jre-6u4-linux-i586.bin`,
  - `jre-6u4-windows-i586-p.exe`,
- folder `Kraksim2008` — dystrybucja systemu Kraksim2008, wersja binarna z podstawową konfiguracją,
- folder `Sources` — źródła pracy dyplomowej,
  - folder `tex` — źródła tekstu pracy dyplomowej wraz z wszystkimi rysunkami oraz tabelami,
  - folder `java` — źródła systemu informatycznego Kraksim2008.

A.2. Instalacja

Aktualna wersja systemu jest przenośna pomiędzy różnymi platformami, działa wszędzie tam, gdzie można zainstalować wirtualną maszynę Java w wersji 1.5 lub wyższej.

System posiada właściwości aplikacji nomadycznej i nie wymaga instalacji. System można uruchomić nawet z płyty CD-ROM. Jedynym wymaganiem jest wcześniejsze zainstalowanie JRE 1.5.

Aby zainstalować system Kraksim2008 na komputerze, na którym nie ma środowiska Java należy:

1. Zainstalować wirtualną maszynę Java. Na dostarczonym dysku CD, w folderze Dependencies znajduje się wersja instalacyjna wirtualnej maszyny Java w wersji 1.6 dla systemów Windows oraz Linux.
2. Skopiować katalog Kraksim2008 do wybranej lokalizacji.
3. W celu uruchomienia systemu z graficznym interfejsem wystarczy kliknąć dwukrotnie ikonę pliku kraksim.jar, który znajduje się w folderze Kraksim2008. Program można uruchomić również z linii poleceń, używając do tego celu skryptów kraksim.bat lub kraksim.sh.

A.3. Program

A.3.1. Uruchamianie

Składnia wywołania programu jest wyświetlana, jeśli uruchomi się go bez żadnych opcji:

```
usage: kraksim [options] algConf modelFile trafficFile

       algConf selects and configures traffic light system driver.
       syntax: algConf = algName[:param1=val,param2=var,...]
       For the list of algorithms and their parameters see below.

options:
       -v                    : turns on visualization (default: off)
       -t transitionDuration : sets the duration of traffic lights'
                               transitional state (default: 8)
       -h                    : shows help
       -s modelSeed          : sets the seed of the traffic simulator
                               RNG (default: based on the system clock)
       -S genSeed            : sets the seed of the traffic generator
                               RNG (default: based on system clock)
       -l learnPhaseCount    : number of learning phases (default: 0)
```

```
       -o statFile           : statistics file name
                               (default: no statistics are generated)
       -r                    : enables dynamic routing (driver can
                               change route)
       -k isRoutingTh        : sets the percentage level of drivers that
                               are willing to change route when needed
       -u interval           : sets the interval for time table update
                               process

algorithms:
       sotl: Self Organizing Traffic Lights
        parameters:
              zone: length of metering zone (default: 50)

       rl: Reinforcement Learning
        parameters:
              discount: discount factor - gamma in RL equations
                        (default: 0.9)
              halve: how often halving RL counters takes place;
                     nonpositive value - never halve (default: 60)
   static: Statical Light plan from topology xml file
     sync: OptAPO priority direction synchronization (uses agent
           platform)
```

Warto zwrócić uwagę na możliwość wygenerowania pliku ze statystykami. Stworzony plik będzie zwykłym plikiem tekstowym czytelnym dla człowieka.

W celu ułatwienia pracy przygotowano szereg skryptów *.bat dla systemu MS Windows (przygotowanie identycznych dla systemów rodziny Unix/Linux nie powinno użytkownikowi sprawiać problemów).

kraksimVisual.bat uruchamia system w trybie graficznym, nie przyjmuje parametrów.

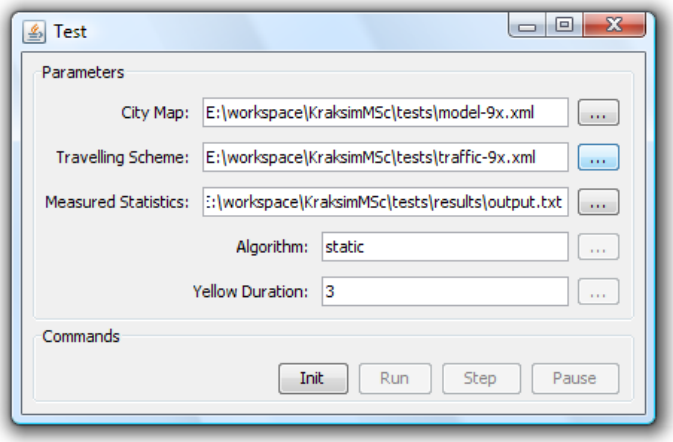
kraksimBatch.bat uruchamia system w trybie wsadowym, przyjmuje 6 parametrów:
%1 - nazwa i konfiguracja algorytmu,
%2 - część nazwy modelu sieci drogowej, tests/model-%2.xml
%3 - część nazwy planu ruchu, tests/traffic-%3.xml,
%4 - czas palenia się żółtego światła, w turach,

%5 - wartość określająca procent kierowców, którzy mogą zmieniać trasę w razie potrzeby,

%6 - wartość określająca okres odświeżania tabeli średnich czasów przejazdu.

Wywołanie `runtest.bat sotl 9X 9xa 4 100 60` spowoduje uruchomienie systemu kraksim2008 z algorytmem sterowania światłami SOTL, z modelem sieci drogowej tests/model-9x.xml i planem ruchu tests/traffic-9xa.xml. Czas sygnału żółtego jest ustawiony na 4s, 100% kierowców może zmieniać trasę w razie potrzeby, czas odświeżania tablicy średnich czasów przejazdu to 60 sekund.

A.3.2. Obsługa interfejsu graficznego

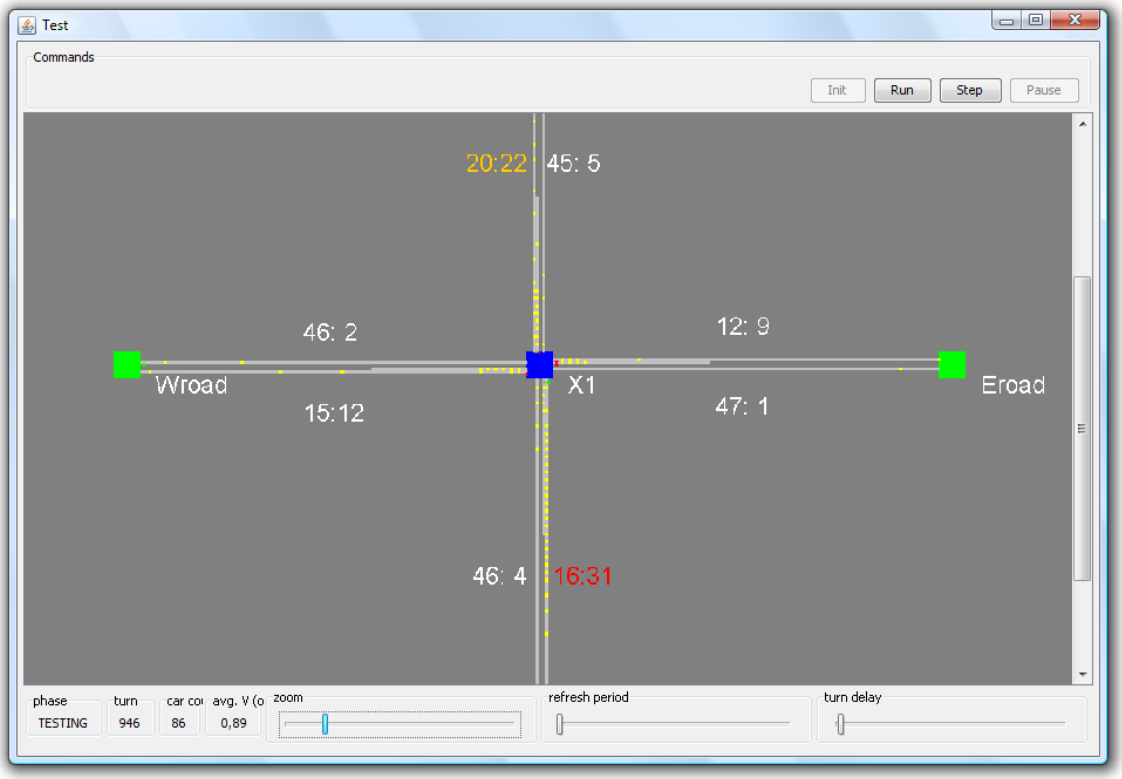


Rysunek A.1. Konfiguracja plików wejściowych

Po uruchomieniu aplikacji, pojawia się ekran konfiguracji (rys. A.1). Należy tu podać lokalizację pliku z opisem topologii miasta, lokalizację pliku ze schematem ruchu, lokalizację pliku, w którym będą zapisywane statystyki oraz wybrany algorytm sterowania światłami i czas trwania żółtego światła.

Po wybraniu podstawowych parametrów można kliknąć przycisk `init` w celu wczytania danych wejściowych i zainicjalizowania programu.

Jeżeli podano prawidłowe ścieżki, a dane wejściowe są prawidłowe, na ekranie pojawi się główne okno symulacji (rys. A.2). Główną część okna zajmuje wizualizacja przebiegu symulacji, która ma postać mapy z zaznaczoną aktualną pozycją pojazdów. Obok każdego połączenia wypisywane są statystyki określające średnią prędkość oraz ilość pojazdów. Dodatkowo na dolnej belce okna zamieszczono licznik pokazujący aktualną turę, ilość pojazdów w modelu oraz średnią prędkość w całym modelu. Na dolnej belce umieszczone są trzy suwaki, pierwszy służy do zmiany skali mapy, drugim



Rysunek A.2. Główne okno wizualizacji przebiegu symulacji

zmienia się czas odświeżania animacji a trzecim reguluje prędkość przebiegu symulacji (symulację można zwolnić do prędkości jednego kroku na sekundę).

Użytkownik ma do dyspozycji 4 przyciski sterujące przebiegiem symulacji: Init, Run, Step, Pause. Init służy do wczytania plików konfiguracyjnych i inicjalizacji modelu i jest dostępny tylko do pierwszego użycia. Run uruchamia proces symulacji w trybie pętli (po zakończeniu tury następuje rozpoczęcie następnej). Step służy do uruchomienia jednego kroku symulacji. Pause umożliwia wstrzymanie symulacji.

A.3.3. Formaty plików wejściowych

Pliki wejściowe dla systemu symulacji ruchu drogowego mają postać strukturalnych dokumentów xml

Sieć drogowa

Aby opisać konfigurację sieci drogowej należy utworzyć dokument xml z następującym nagłówkiem (standardowy nagłówek określający xml, format 1.0). Ten dokument zaczyna się od tagu `RoadNet`.

```
<?xml version="1.0"?>
<RoadNet>
```

Plik konfiguracji schematu sieci drogowej powinien zawierać informacje o rozmieszczeniu węzłów sieci, informacje opisujące parametry dróg (długość, pasy zjazdowe) oraz informacje opisujące skrzyżowania. Proces budowy modelu sieci drogowej wymusza określoną kolejność tych trzech bloków opisujących daną sieć drogową. Najpierw należy opisać węzły i ich położenie. Jest dwa rodzaje węzłów gateway - węzeł wejścia/wyjścia oraz intersection - skrzyżowanie.

```
<nodes>
        <gateway id="N" x="500" y="10"/>
        <gateway id="E" x="990" y="500"/>
        <gateway id="S" x="500" y="990"/>
        <gateway id="W" x="10"  y="500"/>
        <intersection id="intersect" x="500" y="500"/>

</nodes>
```

Drugi blok opisuje ulice. Każda ulica (a właściwie każdy fragment łączący dwa węzły) musi mieć unikalny identyfikator, oraz pola *from* i *to*, które określają *kierunek* ulicy oraz węzły(w tym skrzyżowania) będące dwoma końcami ulicy.

Poniższy przykład zawiera opis drogi, która biegnie od węzła N do węzła intersect (kierunek jest ważny dla ustalenia który pas jest lewy, a który prawy). Pas (połączenie), którym samochody poruszają się w kierunku z N do intersect to uplink, i ma długość 1000. Natomiast downlink to połączenie, którym pojazdy poruszają się w kierunku przeciwnym czyli z intersect do N. Identyfikatorem tego fragmentu drogi jest Nroad, a jej nazwą (oraz nazwę całej drogi do której należy ten fragment, nazwa drogi w przeciwieństwie do identyfikatora nie musi być unikalna i nie jest obowiązkowa) jest ulica Pionowa.

```
<roads>
        <road id="Nroad" street="ulica Pionowa" from="N" to="intersect" >
         <uplink>
                <main length="100"/>
                </uplink>
                <downlink>
                        <main length="100"/>
```

```
            </downlink>
        </road>
```

Definiując drogę należy określić, czy ma ona dodatkowe pasy do skrętu w lewo i/lub w prawo. Następny przykład zawiera opis drogi prawie identyczny jak ten wyżej. W tym przypadku połączenie biegnące z N do intersect posiada dodatkowo pas do skrętu w lewo o długości 5 jednostek (w naszej symulacji jednostka to 7,5 metra), a połączenie w przeciwnym kierunku posiada dodatkowo pasy do skrętu w lewo jak i w prawo, obydwa o długości 5 jednostek.

```
        <road id="Nroad" street="ulica Pionowa" from="N" to="intersect" >
            <uplink>
                <main length="100"/>
                <left length="5"/>
            </uplink>
            <downlink>
                <right length="5"/>
                <main length="100"/>
                <left length="5"/>
            </downlink>
        </road>

</roads>
```

Ostatni - trzeci blok opisowy to schemat budowy i działania skrzyżowania.

```
<intersectionDescriptions>
```

Zawiera on identyfikator skrzyżowania, oraz listę akcji dla każdego *ramienia*.

```
        <intersection id="intersect">
```

Każda akcja składa się z reguł nadrzędności.

```
            <armActions arm="Nroad">
```

Dla jednego połączenia wychodzącego ze skrzyżowania (dla danego ramienia) definiowana jest jedna akcja. Jedna akcja może zawierać od 0 do n reguł nadrzędności (zawierają one nazwy pasów nadrzędnych dla bieżącego), gdzie n jest mniejsze od liczby pasów wchodzących do skrzyżowania.

```
 <action lane="0" exit="Eroad">
        <rule entrance="Sroad" lane="0"/>
 </action>
 <action lane="0" exit="Sroad"/>
 <action lane="0" exit="Wroad"/>
</armActions>

<armActions arm="Eroad">
     <action lane="-1" exit="Sroad">
        <rule entrance="Nroad" lane="0"/>
        <rule entrance="Sroad" lane="0"/>
        <rule entrance="Wroad" lane="0"/>
     </action>
     <action lane="0" exit="Wroad">
        <rule entrance="Nroad" lane="0"/>
        <rule entrance="Sroad" lane="0"/>
 </action>
 <action lane="0" exit="Nroad">
        <rule entrance="Sroad" lane="0"/>
 </action>
</armActions>

<armActions arm="Sroad">
    [...]
</armActions>
```

Akcja dotyczy jednego pasa - lane(może to być np. pas do skrętu w lewo). Akcja ma podany cel (tutaj nazwany exit). Taka akcja określa z którego pasa ruchu i w którym kierunku możliwy jest ruch pojazdu. Każda reguła zawiera numer dokładnie jednego pasa jednej drogi(ulicy), który ma wyższy priorytet niż pas którego dotyczy dana akcja.

Opis skrzyżowania powinien zawierać listę faz sygnalizacji świetlnej.

```
           <phase num="1" duration="20" name="NS">
<inlane arm="Nroad" lane="0" state="green" />
<inlane arm="Nroad" lane="-1" state="red" />
```

```
<inlane arm="Sroad" lane="0" state="green" />
<inlane arm="Sroad" lane="-1" state="red" />

<inlane arm="Wroad" lane="0" state="red" />
<inlane arm="Wroad" lane="-1" state="red" />

<inlane arm="Eroad" lane="0" state="red" />
<inlane arm="Eroad" lane="-1" state="red" />
</phase>

            <phase ...>
                   [...]
            </phase>
```

Opis jednej fazy sygnalizacji składa się z listy wszystkich pasów wchodzących do skrzyżowania wraz z kolorami świateł dla tych pasów. Każda faza ma swój numer.

```
           <plan name="WE" >
<phase num="1" duration="20" name="NS" />
<phase num="2" duration="10" name="lNS" />
<phase num="3" duration="68" name="WE" />
<phase num="4" duration="10" name="lWE" />
</plan>
<plan name="NS" >
<phase num="1" duration="68" name="NS" />
<phase num="2" duration="10" name="lNS" />
<phase num="3" duration="20" name="WE" />
<phase num="4" duration="10" name="lWE" />
</plan>
        </intersection>
</intersectionDescriptions>
```

Na zakończenie pliku wpisujemy tag:

```
</RoadNet>
```

Pełne przykłady plików dostarczone są wraz z projektem w zestawach testowych. Do zestawów, dołączone są krótkie opisy, rysunki i wykresy. Jeśli ten opis czegoś nie wyjaśnia to na pewno można to zrozumieć z zestawów testowych.

Konfiguracja generatorów ruchu

Konfiguracja generatorów ruchu, czyli opis wzorów generowania pojazdów dla potrzeb symulatora jest prostszy niż budowa sieci dróg. Plik zaczyna się podobnie, na początku nagłówek xml, wersja 1.0 oraz tag rozpoczynający dokument, w tym przypadku `<traffic>`.

```
<?xml version="1.0"?>
<traffic>
```

Główną część tego pliku stanowią schematy ruchu. Pojedynczy schemat posiada atrybut określający ilość pojazdów – `count`. Schemat to po prostu lista węzłów (typu gateway) wejściowych, z wyszczególnieniem pory pojawienia się pojazdów.

```
    <scheme count="250">
        <gateway id="N">
```

Czas pojawienia się pojazdów określa się podając jeden z trzech możliwych rozkładów, oraz jego parametry. Dostępne rozkłady to rozkład normalny, rozkład jednorodny oraz dokładny punkt czasu, w którym grupa pojazdów ma się pojawić.

```
            <uniform a="0" b="2700"/>
        </gateway>
```

Aby zdefiniować rozkład punktowy wystarczy podać tylko punkt w jednowymiarowej przestrzeni czasu `y` (jednostką jest sekunda, początek jest o północy - początek doby).

```
    <point y="1200"/>
```

Rozkład normalny określa się podając czas y , oraz wartość odchylenia standardowego dev:

```
<normal y="1200" dev="30"/>
```

Rozkład jednostajny określa się podając punkt startu i końca a i b.

```
<uniorm a="1000" b="1300"/>
```

Należy pamiętać że ostatni węzeł w każdym schemacie nie posiada informacji o czasie, gdyż jest to węzeł docelowy – wyjściowy i w nim nie generuje się pojazdów do symulacji. W ostatnim węźle pojazdy są „niszczone”.

```
        <gateway id="E">
        </gateway>
```

```
    </scheme>
    <scheme count="250">
        <gateway id="N">
            <point y="200"/>
        </gateway>
        <gateway id="S">
        </gateway>
    </scheme>
    <scheme count="1000">
        <gateway id="N">
            <normal y="2700" dev="20"/>
        </gateway>
        <gateway id="W">
        </gateway>
    </scheme>
</traffic>
```

A.3.4. Format pliku statystyk

Dla potrzeb pracy magisterskiej zaimplementowano przykładowy moduł statystyk MiniStat. W trakcie symulacji generowane są statystyki szczegółowe i za pomocą klasy pomocniczej SatsUtil są one zapisywane do pliku tekstowego z rozszerzeniem txt. Po zakończeniu symulacji obliczane są statystyki sumaryczne i zapisywane są do drugiego pliku statystyk, tym razem z rozszerzeniem txt.sum. Pliki statystyk zawierają wartości oddzielone znakami odstępu.

Format pliku statystyk szczegółowych. Plik statystyk szczegółowych zaczyna się nagłówkiem.

```
<LINK_1> <LINK_2> ... <LINK_N> #of_travels #of_cars avg_velocity
```

Kolejne pola `<LINK_1>` w nagłówku to nazwy skierowanych połączeń, wartości w kolejnych wierszach odpowiadające tym nagłówkom pokazują liczbę pojazdów, które przejechały przez dane połączenie. `#of_travels` oznacza ilość zakończonych podróży. `#of_cars` to ilość pojazdów w systemie symulacji, czyli ilość pojazdów, które aktualnie przemierzają drogi modelu zsumowana z ilością pojazdów, które już wystartowały, ale jeszcze nie są obecne w modelu, ponieważ czekają w kolejce za punktem wjazdu (ta wartość pomaga porównać różne przebiegi symulacji). `avg_velocity` oznacza średnią prędkość w systemie.

Format pliku statystyk sumarycznych. Ten plik podzielony jest na 3 części. Pierwsza części zawiera czas trwania symulacji w turach oraz średnią prędkość.

```
CITY STATS
==========
  sim. duration   avg. velocity
          43201            1,23
```

Druga część zawiera statystyki tras (gdzie trasa rozumiana jest jako podróż z miejsca początkowego do celu). W każdej linijce pojawiają się następujące pola: identyfikator węzła źródłowego, identyfikator węzła docelowego, liczba podróży, średni czas trwania podróży w turach, odchylenie standardowe, średnia prędkość dla tej trasy wyrażona w komórkach na turę oraz w kilometrach na godzinę.

```
ROUTE STATS
===========
  from  to count avg. duration <-std dev. avg. velocity  <-[kph]
    G4  G3   616         561,0      650,2          0,71     19,3
    G4  G6   609         954,5      638,6          1,10     29,7
```

Trzecia część zawiera statystyki dla połączeń. Znaczenie kolejnych pól jest dokładnie takie samo jak, dla statystyk tras. Oczywiście w tym przypadku tabela zawiera statystyki dla wszystkich połączeń modelu, a nie dla tras.

```
LINK STATS
==========
  from  to count avg. duration <-std dev. avg. velocity  <-[kph]
    G6  R6  5842          56,5        1,7          1,77     47,8
   X12  X9  3854         341,4       26,7          1,61     43,5
```