Akademia Górniczo – Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki
Katedra Informatyki

Praca magisterska

Elastyczne środowisko do modelowania i optymalizacji ruchu drogowego

Piotr Doliński

Promotor: dr inż. Jarosław Koźlak

Kraków wrzesień 2012

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 RYSUNKÓW

SPIS TABEL

1. WPROWADZENIE

Wzrost znaczenia i zróżnicowanie transportu jest naturalnym wynikiem rozwoju cywilizacji. Z biegiem czasu rosła wielkość i ilość przemieszczanych towarów, zwiększała się szybkość i zasięg podróży. Wynikiem tego procesu jest współczesny ruch drogowy, kolejowy, lotniczy i morski.

Z punktu widzenia ekonomii transport pełni bardzo ważną funkcję. Ważny jest tani i szybki przewóz towarów, pośrednio poprzez możliwość dogodnego przemieszczania osób ogranicza się koszty (lepsze kontakty rodzinne, biznesowe, praca w bardziej odległych miejscach zamieszkania). Z uwagi na dogodność transportu samochodowego, wzrasta jego powszechność, co powoduje jednocześnie duże zagęszczenia na drogach. Celem światowej gospodarki jest wyprodukowanie jak największej ilości towarów i usług, przy jak najmniejszym nakładzie kosztów i czasu pracy. Ulepszenie systemu drogowego sprawia, że krótszy czas podróży przekłada się na krótszy czas pracy kierowców. Dzięki temu ta sama grupa ludzi, może obsłużyć i dostarczyć większą ilość towarów. Należy tu podkreślić, że koszty transportu stanowią znaczną część ceny towarów.

Jedną z dziedzin prac naukowych związanych z zagadnieniem transportu jest optymalizacja ruchu drogowego. Badania nad optymalizacją prowadzone są od lat 50-tych dwudziestego wieku. Można m.in. optymalizować ruch w gęstej sieci drogowej miast. Wykonuje się to na bardzo wiele sposobów. Zmieniana może być konstrukcja sieci drogowej poprzez dodawanie nowych dróg lub nowych pasów ruchu, a także przez zmianę ich organizacji. Optymalizuje się ruch poprzez sygnalizację świetlną. Powstają systemy wpływania na decyzje kierowców poprzez przekazywanie im informacji drogą radiową lub systemami nawigacji GPS. Jest bardzo prawdopodobne, że w przyszłości kierowcy zostaną usunięci z pojazdów. Samochodami będzie sterował komputer, a ruch będzie w całości zarządzany przez oprogramowanie.

Jednym z najważniejszych i ciągle aktualnych problemów, z jakimi musi się zmagać transport, jest problem powstawania zagęszczeń i zatorów w nieposiadającej dostatecznej przepustowości sieci dróg. Autor zamierza przyjrzeć się temu problemowi i następnie zaproponować rozwiązanie, które

będzie głosem w dyskusji nad rozwiązaniem tego podstawowego problemu transportowego.

Celem niniejszej pracy jest opracowanie i zaimplementowanie zestawu rozwiązań służących do predykcji i optymalizacji ruchu drogowego modelowanego przez elastyczne środowisko symulacyjne. W celu jego realizacji, system symulacyjny Kraksim [1] [2] zostanie rozszerzony o nowy moduł implementujący efektywny algorytm przewidywania zagęszczeń w ruchu drogowym. Zaproponowane rozwiązanie ma dokładnie przewidywać powstawanie korków w gęstej sieci drogowej miast, w krótkich okresach czasu. W oparciu o dane uzyskane z symulacji zostanie zmierzona efektywność algorytmu przewidywania zagęszczeń. Udowodniona zostanie również możliwość skutecznego wykorzystania nowego modułu w procesie zapobiegania powstawania zagęszczeń w ruchu drogowym. Pozwoli to na zwiększenie płynności ruchu w systemie symulacyjnym.

Dodanie nowego modułu zwiększy wybór algorytmów umożliwiających optymalizację ruchu drogowego w systemie. Wzrośnie ilość opcji konfiguracyjnych i rodzajów statystyk opisujących sytuacje na drogach w czasie trwania symulacji. Dzięki temu będzie możliwe lepsze zrozumienie symulowanych sytuacji i podejmowanie decyzji co do efektywnego zarządzania ruchem ulicznym w mieście.

Praca magisterska rozpoczyna się przedstawieniem sposobów monitorowania parametrów ruchu drogowego. Następnie opisane są algorytmy i rozwiązania stosowane do przewidywania charakterystyk ruchu drogowego na podstawie wartości tych parametrów. Stanowi to treść drugiego rozdziału.

Trzeci rozdział omawia algorytmy klasyfikacji, które zostaną wykorzystane w procesie przewidywania korków w zaproponowanej w pracy metodzie przewidywania. Opisane zostanie również oprogramowanie Weka z którego pochodzić będą implementacje tych algorytmów.

Czwarty rozdział przedstawia różne podejścia do implementacji systemów symulujących ruch uliczny. Znajdzie się tam również opis systemu Kraksim, którego rozwój jest głównym zadaniem niniejszej pracy.

W rozdziale piątym sformułowano zestaw wymagań stawianych modułowi przewidywania zagęszczeń. Dalszą cześć rozdziału zawierać będzie opis koncepcji algorytmu przewidywania korków ulicznych.

Treść następnego rozdziału stanowić będzie opis implementacji zaproponowanego w rozdziale piątym rozwiązania. Opisane zostanie też jego miejsce w systemie Kraksim i współpraca z nim.

Siódma część rozprawy zawiera opis testów i ich wyniki, a także komentarz do nich.

Ostatnią część pracy stanowi podsumowanie, w którym autor zbierze najważniejsze wnioski i wyniki pracy. Na tej podstawie zostaną zaproponowane dalsze kierunki przyszłych działań, jakie można podjąć w oparciu o uzyskane wyniki.

2 PRZEGLĄD DZIEDZINY

W celu podniesienia efektywności i bezpieczeństwa transportu, dzięki rozwojowi technologii informatycznych, telekomunikacyjnych, pomiarowych i automatyki, tworzone są Inteligentne Systemy Transportowe (ang. ITS – Intelligent Transportation Systems) [3]. Sama nazwa została zdefiniowana w 1994 roku w Paryżu, ale prace nad optymalizacja transportu prowadzone są od wielu lat. Już w latach 30-tych dwudziestego wieku w Stanach Zjednoczonych próbowano za pomocą równań matematycznych opisać ruch autostradowy. Dziś jest to szeroka dziedzina badań uniwersyteckich. Istnieje też szeroki rynek produktów oferowanych firmom, administracji publicznej i samorządom.

Część badań nad systemami ITS stanowią prace nad zrozumieniem i rozwiązaniem problemu zagęszczeń w ruchu miejskim. Rodzaj prowadzonych prac jest bardzo szeroki. Obejmuje on między innymi:

- analizę metod monitorowania i pozyskiwania danych o ruchu drogowym,
- analizę pozyskanych danych w celu znalezienia charakterystycznych wzorców,
- wytworzenie skutecznych metod prognozowania zagęszczeń,
- analizę przydatności różnych rodzajów danych do prognozowania,
- wykorzystanie wyników prognozowania w metodach zapobiegającym powstawaniu zagęszczeń.

2.1 MONITOROWANIE RUCHU DROGOWEGO

Monitorowanie ruchu drogowego stanowi część technik dostarczania danych dla inteligentnych systemów transportowych. Dużą ich część stanowią automatyczne systemy sterowania np. sygnalizacją świetlną. Dlatego też pozyskiwane dane muszą być aktualne i dostarczane praktycznie natychmiast. Wymagana jest też dostateczna dokładność zbieranych informacji, uzyskiwana pomimo wpływu warunków zewnętrznych (np. pogodowych), anormalnych sytuacji, awarii i zawodności sprzętu.

Wśród sposobów monitorowania ruchu wyróżnić można [4] [5]:

Czujniki nacisku:

- Sensory pneumatyczne – są to gumowe przewody wbudowane w drogę wykrywające samochody w oparciu o zmiany ciśnienia gazu w przewodzie spowodowane naciskiem osi pojazdu. Wadą tego podejścia jest słaba dokładność uzyskiwanych danych, mocno zależna od warunków atmosferycznych.
- Czujniki piezoelektryczne – czujniki wykorzystujące zjawisko piezoelektryczne. Nacisk kół pojazdu powoduje zmianę ilości ładunku elektrycznego na powierzchni materiału piezoelektrycznego. Urządzenie monitoruje szybkość i wielkość zmiany ładunku. Zaletą tego podejścia jest jego dokładność. Możliwy jest nie tylko pomiar ilości pojazdów, ale też ich masy i prędkości.
- Pętle indukcyjne – czujnik składa się z przewodu wtopionego w asfalt podłączonego do źródła prądu i czujnika zmian poziomu indukcji elektrycznej. Są najlepszą i najczęściej stosowaną metodą detekcji pojazdów. Ulegają jednak często zniszczeniu przez przeładowane pojazdy.

Zliczanie na odległość:

- Liczenie manualne – najprostsza technika, ale niemożliwa do zastosowania na dużą skalę i nie dostarcza aktualnych danych dla systemów teleinformatycznych.
- Radar mikrofalowy – zliczanie pojazdów odbywa się w oparciu o efekt Dopplera. Ten sposób pomiaru jest całkowicie niezależny od warunków atmosferycznych.
- Czujniki podczerwieni – detektor emituje promieniowanie podczerwone i mierzy czas do powrotu odbitego sygnału. Krótszy czas oznacza obecność samochodu w badanej strefie.
- Radary akustyczne – emitują dźwięk o częstotliwości od 20 do 300 KHz. Detektor mierzy czas powrotu odbitego sygnału lub zmianę jego częstotliwości wynikającą z efektu Dopplera.

- Wykrywanie pojazdów w oparciu o metody rozpoznawania obrazów w zapisie z kamery video.

Flota pojazdów testowych:

W przypadku tej metody pewna, w miarę duża ilość pojazdów wyposażana jest w urządzenia, które przekazują informacje o pozycji i prędkości pojazdu do centrali. Najczęściej wykorzystywane są tutaj gotowe systemy korporacji taksówkowych, komunikacji miejskiej lub nawigacji samochodowej. Wśród metod przekazywania informacji do centrali można wyróżnić następujące metody:

- Zliczanie oparte o systemy GPS.
- Zliczanie oparte na systemach telefonii komórkowej.
- Zliczanie oparte o nadajniki radiowe (Systemy AVL i AVI).

Systemy te dostarczają bardzo dokładnych danych o ruchu, jednak problemem w ich zastosowaniu jest wysoka cena związana z zapewnieniem wystarczająco dużej ilości pojazdów próbkujących. Aktualnie rozwijać będą się systemy oparte o telefonię komórkową i systemy nawigacji samochodowej. Firmy obsługujące systemy nawigacji samochodowej, poprzez aplikacje mobilne, będą zdobywać aktualne i bardzo dokładne informacje o ruchu. Dostęp do danych będzie sprzedawany m.in. zarządcom ruchu w miastach. Cena dostępu do danych może być na tyle atrakcyjna, że może wyeliminować z rynku inne metody monitorowania.

Monitorowane parametry

Najprostszym do monitorowania parametrem ruchu jest ilość pojazdów, jednak w celu wyciągnięcia jak najbardziej dokładnych wniosków o sytuacji w sieci drogowej, wymagane jest zebranie dokładniejszych danych. Dlatego monitoruje się również:

- Prędkość.
- Stopień zatłoczenia drogi.
- Czas przejazdu między dwoma punktami.
- Długość kolejki pojazdów przed skrzyżowaniem, czas oczekiwania w kolejce, ilość pojazdów, które muszą zatrzymać się przed skrzyżowaniem.

- Klasyfikację pojazdów (samochody osobowe, autobusy, lekkie i ciężkie samochody ciężarowe, samochody specjalne).

Kolekcjonowane są też inne dane nie określające bezpośrednio ruchu drogowego, ale mogące mieć na niego wpływ: stan nawierzchni i warunki pogodowe, rodzaj drogi, dzień tygodnia, niespodziewane zdarzenia, wypadki i blokady dróg.

Wartości zbieranych parametrów mierzone są w różnych odstępach czasowych. Dla gęstych sieci drogowych w centrach miast jest to okres 30 czy 60 sekundowy. Dla głównych arterii miejskich okres obserwacji jest dłuższy i może wynosić 15, 30 czy 60 minut. Część systemów wykorzystuje tylko dane aktualne, ale istnieją też systemy, gdzie informacja jest gromadzona i proces przewidywania czy odkrywania wzorców odbywa się zarówno na danych aktualnych jak i historycznych. Częstotliwość monitorowania parametrów zależy w dużej mierze od możliwości ich wykorzystania.

2.2 METODY OPTYMALIZACJI I ZAPOBIEGANIA POWSTAWANIU ZAGĘSZCZEŃ W RUCHU DROGOWYM

2.2.1 METODY OPARTE O PRZEWIDYWANIE CZASU PRZEJAZDU

Istnieje wiele podejść badawczych do problemu optymalizacji ruchu drogowego. Jednym z nich jest zapobieganie powstawaniu zagęszczeń poprzez przekazywanie kierowcom informacji o prognozowanych czasach przejazdu ulicami miast. Przykładem takiej metody jest **przewidywanie czasu przejazdu samochodów w oparciu o rozpoznawanie wzorców ruchu** [6]**.** Przewidywany czas przejazdu został użyty w systemach znaków o zmiennym tekście i w systemach nawigacji samochodowej w celu dostarczenia kierowcom użytecznych informacji, w oparciu o które będą podejmować lepsze decyzje o wyborze tras przejazdu.

Algorytm został przetestowany na danych zebranych z tokijskiej sieci dróg szybkiego ruchu. Według badaczy najlepiej ruch na takich drogach opisuje parametr prędkości samochodów. Dane z detektorów prędkości zapisywane są w bazie danych. Na podstawie tych danych historycznych tworzone są wzorce

ruchu. Wzorzec stanowi zbiór obserwacji zebranych z detektorów z danej drogi w kilku kolejnych okresach.

Przeszukanie całej bazy danych w poszukiwaniu wzorców ruchu jest niemożliwe, ze względu na zbyt długi czas wyszukiwania. Badacze przyjmują [7], że ruch o tych samych parametrach pojawia się cyklicznie o tych samych godzinach w kolejnych dniach tygodnia. W bazie danych wyszukuje się więc wzorce nie bardziej odległe w czasie niż pewna liczba minut od danej godziny i dnia tygodnia.

Proces wyszukiwania podobnych wzorców posiada kilka parametrów:

- długość okresu czasu – określa wielkość wzorca. Jest to ilość kolejnych pomiarów wykonanych przez detektory w tym okresie czasu,
- liczba wyszukiwanych wzorów na podstawie których następuje przewidywanie,
- rozmiar okna wyszukiwania – okres czasu, w jakim próbuje się znaleźć podobny wzorzec.

Problemem jest ustalenie najlepszych wartości tych parametrów. W celu ich określenia twórcy artykułu stosują algorytm genetyczny.

Model zbudowany w oparciu o wyliczone parametry wykorzystywany jest do przewidywania aktualnego czasu przejazdu. W celu weryfikacji poprawności rozwiązania, badacze przeprowadzili test uzyskanego modelu dla 12km drogi z 40 detektorami prędkości. W zależności od dnia, dla którego przeprowadzono walidację podejścia, uzyskano średni błąd względny predykcji czasu przejazdu od 4.7% do 17%.

Inną metodą optymalizacji ruchu [8] jest **wykorzystanie wirtualnego systemu przewidującego korki uliczne w oparciu o flotę pojazdów testowych i wykorzystanie podejścia „feromonowego”**. Artykuł prezentuje zdecentralizowane podejście do przewidywania zagęszczeń w ruchu. Dane do systemu zbierane są przez specjalne samochody testowe wyposażone w odpowiednie czujniki monitorujące prędkość, częstotliwość użycia hamulca i odległość od poprzedzającego samochodu. Każdy z tych parametrów zamieniany

jest na informację feromonową, którą oznaczane są punkty na wirtualnej mapie. Informacja pozostawiona przez samochód z biegiem czasu ulega zatarciu. Samochody poruszając się wzdłuż sieci drogowej odczytują pozostawione na drodze informacje. Na ich podstawie następuje próba przewidzenia czasu przejazdu.

Zaproponowaną metodę przewidywania czasu przejazdu porównano z przewidywaniem za pomocą średniej ruchomej. Dla jednej drogi przewidywano czas co jedną minutę w ciągu godziny. Współczynnik korelacji przewidywania i wartości rzeczywistej w przypadku średniej wyniósł 0.57, a w przypadku badanej metody 0.67.

Autorzy mają nadzieję, że w przyszłości dzięki ich metodzie samochody będą w stanie przewidywać korki na podstawie informacji dostarczanych przez poprzedzające je samochody.

Podejście przedstawione przez autorów tej pracy jest zdecydowanie inne od poprzedniego. Nie wymaga ono wykorzystania danych historycznych, a co za tym idzie dużych mocy obliczeniowych.

Do przewidywania czasu przejazdu można również użyć **algorytm sieci neuronowych** [9]. W artykule opisywana jest metoda przewidywania czasu przejazdu na podstawie różnych typów danych, takich jak: prędkość, informacje o incydentach na drodze czy wielkość przepływu samochodów. Najlepsze wyniki uzyskiwane są dla parametru prędkości. Co ciekawe, próba nauczenia sieci neuronowej kilkoma typami danych pogarsza wyniki prognoz.

2.2.2 STEROWANIE SYGNALIZACJĄ ŚWIETLNĄ

Jeszcze inną metodą eliminacji korków ulicznych jest **metoda skutecznego zarządzania sygnalizacją świetlną** [10]**.** Rozwiązanie wykorzystuje dynamiczną sieć bayesowską do przewidywania rozkładu prawdopodobieństwa ilości samochodów na ulicy, w kolejnych okresach działania sygnalizacji świetlnej. Na podstawie tych przewidywań algorytm zmienia czas działania faz sygnalizacji świetnej.

Jeżeli prawdopodobieństwo, że w przyszłym okresie na ulicy przed skrzyżowaniem będzie czekało więcej samochodów niż wartość Smax, jest większe od α, należy wydłużyć czas świecenia się zielonego światła. Jeżeli prawdopodobieństwo, że w przyszłym okresie na ulicy będzie czekało mniej samochodów niż wartość Smin jest większe niż β, to czas świecenia się zielonego światła dla tej drogi jest zmniejszany. Po tej zmianie prawdopodobieństwa rozkładu samochodów na ulicach obliczane są na nowo i cały proces jest powtarzany, aż spełnione będą podane warunki.

W wyniku zastosowania metody, liczba samochodów stojąca przed skrzyżowaniem spadała o 37 procent w stosunku do symulacji, w której używano sygnalizacji świetlnej ze stałym okresem palenia się świateł.

Wśród metod optymalizacji ruchu poprzez sterowanie sygnalizacją świetlną można również wyróżnić algorytmy SOTL, RL, OptAPO. Zostaną one omówione w rozdziale 4.2.4. w ramach przedstawiania systemu symulacyjnego Kraksim.

3 METODY REGRESJI W OPROGRAMOWANIU WEKA

Regresja statystyczna jest procesem przewidywania nieznanych wartości jednych wielkości na podstawie znanych innych wartości [11]. Cechy, których wartości znamy, to zmienne objaśniające (niezależne), natomiast cechy, których wartości będziemy szukać, określane są mianem objaśnianych (zależnych). Regresja, w której badamy zależność jednej zmiennej zależnej, czasami zwanej też krytyczną, od wielu zmiennych niezależnych nazwana jest regresją wieloraką.

3.1 WEKA

Weka [12] jest oprogramowaniem napisanym w języku Java, które służy do odkrywania wiedzy (ang. data mining) w zbiorach danych. Oprogramowanie pozwala na użycie wielu metod, przygotowujących zbiór danych do analizy. Możliwe jest przeprowadzenie selekcji cech, filtrowania, dyskretyzacji czy normalizacji danych. Następnie Weka umożliwia wykorzystanie algorytmów uczenia maszynowego do przeprowadzenia procesów klasyfikacji, klastrowania lub znajdowania wzorców częstych.

Część algorytmów klasyfikacji jest również algorytmami, które mogą budować model regresji. Jest to możliwe, gdyż klasyfikacja i regresja to procesy bardzo podobne, różniące się jedynie rodzajem zmiennej objaśnianej. W klasyfikacji przyjmuje ona tylko wartości dyskretne, natomiast w regresji jest zmienną ciągłą. W oprogramowaniu Weka zmienne określane są mianem atrybutów. Typ atrybutu objaśnianego, ciągły bądź dyskretny, decyduje czy mamy do czynienia z regresją czy klasyfikacją.

3.2 ALGORYTM K-NAJBLIŻSZYCH SĄSIADÓW

Algorytm K-najbliższych sąsiadów [13] jest metodą typu Instance Based Learning. Algorytm nie buduje modelu zbioru danych. Można powiedzieć, że sam zbiór danych uczących jest modelem w przypadku tego podejścia. W czasie procesu testowania każda instancja zbioru testowego jest porównywana z instancjami modelu przy użyciu pewnej metryki odległości. Najbliższa instancja pod względem tej metryki determinuje wartość atrybutu objaśnianego dla testowanego punktu. Ten prosty proces zwany jest algorytmem najbliższego

sąsiada. Algorytm k-najbliższych sąsiadów jest jego prostą modyfikacją. Wartość klasy instancji testowej określana jest na podstawie k-najbliższych punktów według zadanej metryki. Wartość szukanego atrybutu dla badanej instancji określana jest poprzez obliczenie średniej arytmetycznej.

Wadą algorytmu k-najbliższych sąsiadów jest duże zapotrzebowanie na pamięć operacyjną. Model klasyfikatora zbudowany jest ze wszystkich instancji uczących. Natomiast proces uczenia przebiega praktycznie natychmiastowo. Do odpowiedniej struktury danych kopiowane są przykłady uczące lub referencje do nich.

Algorytm k-najbliższych sąsiadów w oprogramowaniu Weka oznaczony jest nazwą IBk[1].

3.3 DRZEWA REGRESJI

Struktura drzewa [13]**.** Drzewo zbudowane jest z węzłów i liści. Węzeł tworzy test na jednym z atrybutów, który dzieli zbiór uczący na dwa poddrzewa. Poddrzewo może rozpoczynać się kolejnym węzłem lub być liściem. Liść determinuje wartość wyniku regresji.

Jeżeli atrybutem klasyfikacyjnym jest atrybut nominalny, klasą liścia jest najliczniejsza klasa instancji, wyznaczających ten liść. Gdy atrybut jest liczbą rzeczywistą, obliczana jest średnia arytmetyczna z wszystkich wartości atrybutu objaśnianego dla wszystkich instancji uczących jakie reprezentuje dany liść. Takie drzewo nazywane jest drzewem regresyjnym.

Biorąc instancję testową i przechodząc przez wszystkie testy w węzłach drzewa, od korzenia do jednego z liści, otrzymujemy klasyfikację tej instancji. Dla atrybutu nominalnego test jest porównaniem z jedną z możliwych wartości atrybutu. W przypadku atrybutu rzeczywistego obliczana jest pewna stała wartość, która tworzy podział zbioru liczb rzeczywistych na dwa przedziały. Test polega na sprawdzeniu w którym z tych przedziałów znajduje się badany atrybut testowanej instancji.

Proces budowy drzewa. Tworzenie drzewa rozpoczyna się od stworzenia korzenia. Wybór atrybutu, na którym dokonywany jest podział zbioru uczącego

[1] Weka javadoc - http://weka.sourceforge.net/doc/index.html?weka/classifiers/lazy/IBk.html

determinowany jest maksymalną wartością zysku informacyjnego. Podział dokonywany jest do momentu, aż w każdym liściu znajdzie się tylko jedna instancja i dalszy podział jest już niemożliwy.

Przycinanie drzewa (ang. pruning). Proces budowy pełnego drzewa jest bardzo kosztowny. Niektóre algorytmy zaprzestają więc rozwijania kolejnych węzłów, jeżeli maksymalna wartość zysku informacyjnego z wszystkich atrybutów w rozwijanym węźle jest zbyt mała. Jednak większość algorytmów tworzy pełne drzewo decyzyjne, a następnie próbuje zmniejszać jego rozmiar poprzez przycinanie.

Najprostszą metodą przycinania drzewa jest zamiana pewnego poddrzewa w liść. Drugą metodą jest operacja podnoszenia poddrzewa (ang. subtree raising). Dany węzeł A w drzewie zastępowany jest poprzez jego poddrzewo B, a wszystkie pozostała poddrzewa i liście węzła A stają się nowymi poddrzewami i liśćmi węzła B. Algorytm stara się wykonać obie operacje na każdym węźle drzewa rozpoczynając od jego liści i poruszając się w kierunku korzenia.

Decyzja o tym, czy wykonać jedną z dwóch powyższych operacji, podejmowana jest na podstawie obliczanego dla każdego węzła błędu przewidywania. Drzewo budowane jest w oparciu tylko o część zbioru treningowego. Pozostałe instancje używane są do obliczania błędu przewidywania przy procesie przycinania drzewa. Dla każdego węzła obliczany jest więc błąd przewidywania przed i po przeprowadzeniu jednej z dwóch operacji przycinania. Jeżeli błąd po przeprowadzeniu jednej z dwóch operacji jest mniejszy, drzewo jest przycinane.

W oprogramowaniu Weka opisany powyżej algorytm zaimplementowany jest pod nazwą REPTree[2].

3.4 Reguły klasyfikacyjne

Drzewo modeli (ang. model tree) budowane jest w ten sam sposób jak drzewo regresji. W swoich liściach posiada, zamiast pojedynczej wartości będącej średnią z wartości atrybutu objaśnianego instancji uczących, liniowy model

[2] Weka javadoc - http://weka.sourceforge.net/doc/index.html?weka/classifiers/trees/REPTree.html

regresji dla tego atrybutu. Model ten budowany jest dla instancji, które dany liść reprezentuje.

Proces budowy drzewa modeli rozpoczyna się rekurencyjnie od korzenia. W każdym węźle podział zbioru przykładów uczących następuje w oparciu o minimalizację odchylenia standardowego dla atrybutu zmiennej zależnej. W tym celu w węźle buduje się liniowy model regresji przewidujący wartość tego atrybutu. Następnie dla każdej zmiennej niezależnej wykonuje się podział zbioru treningowego na dwa podzbiory i sprawdza jak ten podział wpłynął na wartość odchylenia standardowego tych podzbiorów. Wśród wszystkich tych podziałów wybiera się ten, który najbardziej zminimalizował odchylenie standardowe całego zbioru. Na dwóch powstałych podzbiorach algorytm znów spróbuje dokonać podziału na nowe węzły drzewa. Proces podziału kończy się, jeżeli w węźle pozostało tylko kilka instancji uczących albo odchylenie standardowe tych instancji stanowi tylko niewielki procent odchylenia standardowego całego zbioru uczącego.

Proces przycinania drzewa przebiega rekurencyjnie od liści do korzenia. Dla danego węzła i jego poddrzew obliczany jest błąd bezwzględny przewidywania dla podzbiorów zbioru uczącego, które każdy z tych węzłów reprezentuje. Jeżeli błąd ten jest mniejszy w węźle, niż w jego poddrzewach, następuje proces przycinania, w którym badany węzeł staje się liściem.

Generowanie reguł. Gdy tworzone są reguły na podstawie drzewa modeli, nie jest tworzone pełne drzewo. Gdy tworzony jest liść, liść ten zamieniany jest na regułę, która po swojej prawej stronie posiada model regresji dla przykładów uczących, które ten liść i regułę reprezentują. Następnie te przykłady usuwane są ze zbioru danych uczących używanych do tworzenia drzewa modeli. Algorytm przystępuje więc znowu do próby utworzenia kolejnej reguły z następnego utworzonego liścia w drzewie.

Algorytmem implementującym generowanie reguł regresyjnych z drzewa modeli jest algorytm M5Rules[3]. W systemie występuje też jego podobna wersja,

[3] Weka javadoc - http://weka.sourceforge.net/doc/index.html?weka/classifiers/rules/M5Rules.html

gdzie do przeprowadzania regresji wykorzystywane jest bezpośrednio drzewo modeli. Można go znaleźć pod nazwą M5P[4].

3.5 INNE ALGORYTMY

Weka zawiera też wiele innych algorytmów regresji takich jak: SMOReg, który implementuje algorytm regresji oparty o maszynę wektorów nośnych (SVM – ang. suport vector machine). Istnieje też implementacja sieci neuronowej ze wsteczną propagacją błędów – MultilayerPerceptron i implementacja bardziej skomplikowane algorytmu typu Instance Based nazwana KStar.

Możliwe do użycia jest też wiele klasyfikatorów zbiorczych (ang. ensemble methods), gdzie przewidywana wartość wyliczana jest w oparciu o decyzje zbioru prostych klasyfikatorów lub modeli regresji. Zaimplementowane są wszystkie najpopularniejsze ich rodzaje: boosting, bagging, stacking.

Algorytmy te są bardzo dokładne w swoich przewidywaniach i osiągają znacznie lepsze wyniki niż proste modele regresji. Wymagają jednak znacznie większej mocy obliczeniowej procesora lub znacznie większego czasu obliczeń.

[4] Weka javadoc - http://weka.sourceforge.net/doc/index.html?weka/classifiers/rules/M5Rules.html

4 SYMULACJA RUCHU DROGOWEGO

Tworzenie i wdrażanie systemów ITS jest bardzo kosztowne i trwa dość długo. Wymagane jest więc, aby móc wcześniej ocenić opłacalność planowanych rozwiązań. W tym celu wykorzystuje się różnego rodzaju symulatory ruchu drogowego. Pozwalają one na ocenę ruchu drogowego, identyfikację problemów, które w nim zachodzą oraz znalezienie ich przyczyn. Symulacje pozwalają również na tanie sprawdzenie skuteczności nowych pomysłów na rozwiązanie problemów z ruchem drogowym. Poniższy rozdział zawiera ogólny opis systemów symulacyjnych i prezentuje system Kraksim, który będzie rozwijany w ramach tej pracy magisterskiej.

4.1 MODELE RUCHU

Głównym kryterium podziału systemów symulacyjnych jest poziom szczegółowości z jakim reprezentowane są w systemie elementy świata rzeczywistego. Podział taki wytworzył się w związku z brakiem odpowiednich mocy obliczeniowych potrzebnych do zasymulowania ruchu drogowego w najdrobniejszych szczegółach. W zależności od rozmiaru sieci drogowej, jaka potrzebna jest do przeprowadzenia badań, używa się symulatora implementującego inny rodzaj modelu ruchu [14] [15]:

Mikroskopowy. W podejściu mikroskopowym każdy samochód reprezentowany jest bezpośrednio w środowisku symulacyjnym. Każdy z nich posiada pozycję, prędkość, może przyspieszać i zwalniać. Środowisko w którym poruszają się pojazdy też modelowane jest bardzo szczegółowo. Na skrzyżowaniach zmieniają się światła sygnalizacji, a samochody w systemie zmieniają swoje zachowanie pod wpływem znaków drogowych. Wszystkie elementy reagują ze sobą zmieniając stan symulacji. Typowym przykładem modulu mikroskopowego jest model Nagla-Schreckenberga [16] opisany w podpunkcie 4.2.1. tej pracy.

W ramach modeli mikroskopowych lub jako oddzielną kategorię można wyróżnić modele **nanoskopowe**. Rzeczywistość w nich modelowana jest jeszcze bardziej szczegółowo niż w typowych modelach mikroskopowych. Modele te w dokładny sposób symulują kierowców pojazdów. Mowa tu o sposobach

podejmowania przez nich decyzji i popełnianych przez nich błędach. Implementowana jest ich interakcja ze środowiskiem zewnętrznym. Kierowcy reagują na zmienne warunki atmosferyczne, czy też nietypowe sytuacje na drodze. Takie symulatory używane są do testowania zagadnień związanych z bezpieczeństwem ruchu drogowego.

Wśród symulatorów implementujących model mikroskopowy można wymienić: CORSIM, SimTraffic, AIMSUN, VISSIM.

Mezoskopowy. Ten rodzaj modeli znajduje się pomiędzy modelami mikroskopowymi i makroskopowymi, jeżeli chodzi o szczegółowość reprezentacji elementów rzeczywistości w symulacji. Pojedyncze samochody mogą być reprezentowane bezpośrednio w systemie, ale są łączone w grupy i tak też reprezentowany jest ich ruch. Istnieją też modele w których symuluje się tylko przejazdy samochodów przez skrzyżowanie. Pasy ruchu nie są symulowane, w sposób uproszczony obliczany jest jedynie czas przejazdu od jednego skrzyżowania do drugiego. Przykładami mezoskopowych systemów symulacyjnych są: INTEGRATION i DynaMIT.

Makroskopowy. W modelach makroskopowych ruch miejski na ulicach opisany jest za pomocą abstrakcyjnych parametrów takich jak prędkość, wielkość przepływu i gęstość samochodów na ulicy. Takie uproszczenie modelu pozwala na symulowanie dużych sieci drogowych i jest wykorzystywane raczej w rozwiązywaniu problemów transportowych niż problemów ruchu drogowego. Systemami które wykorzystują model makroskopowy symulacji ruchu są: KRONOS i KWaves.

4.2 System Kraksim

Niniejsza praca magisterska opiera się na systemie symulacyjnym Kraksim [1] [2] rozwijanym od kilku lat na Katedrze Informatyki. Aplikacja ta jest dobrą bazą do implementacji i testowania własnych algorytmów i koncepcji. W tym podrozdziale zostaną przedstawione najważniejsze pojęcia związane z tym systemem symulacyjnym.

4.2.1 *Mikroskopowy Model Kraksima*

W systemie Kraksim zaimplementowany jest model Nagla-Schreckenberga [16]. Jest to powszechnie stosowany model. Jego zaletą jest prostota, sprawiająca że ilość obliczeń jest nieduża. Umożliwia to symulowanie dużej i skomplikowanej sieci drogowej. Jednocześnie model ten jest wystarczający do modelowania skomplikowanego zachowania pojazdów na drodze.

W podstawowej wersji modelu drogę stanowi jeden pas ruchu składający się z jednowymiarowej tablicy komórek. Mają one jednakowy rozmiar odpowiadający wielkości jaką zajmuje stojący na drodze samochód osobowy wraz z odległością do innych pojazdów. Każda z komórek może być zajęta przez maksymalnie jeden pojazd. Pojazdy poruszają się wzdłuż tablicy zajmując kolejne pola. Ich prędkość wyraża się w ilości komórek na turę symulacji. Tura symulacji składa się z czterech kroków:

- Przyspieszanie – jeżeli samochód n nie jedzie z prędkością maksymalną i odległość d_n od poprzedzającego go pojazdu jest większa o jedną komórkę od odległości jaką pojazd pokona w aktualnej turze, prędkość pojazdu wzrasta według wzoru: $v_n = v_n + 1$.
- Zwalnianie – jeżeli odległość d_n od poprzedzającego pojazdu jest mniejsza niż odległość jaką mógłby samochód n pokonać przy prędkości $v_n + 1$ to prędkość jest zmniejszana o jeden.
- Czynnik losowy – z pewnym prawdopodobieństwem p prędkość v_n pojazdu n zmniejszana jest o jeden.
- Ruch – samochody przemieszczane są wzdłuż tablicy zgodnie ze swoją wartością prędkości v_n.

Model ruchu Nagla-Schreckenberga jest jednak nie wystarczający do budowy i symulacji rzeczywistej sieci drogowej. Istnieje realna potrzeba rozszerzenia tego modelu w celu symulacji ruchu drogowego na ulicach z wieloma pasami ruchu. Takie ulice symuluje się poprzez połączenie kilku równoległych pasów modelowanych jako podstawowe modele jednopasmowe.

Rozszerzany jest również zestaw kroków wykonywanych w jednej turze symulacji [17] [18]. Dodane są dwa dodatkowe kroki wykonywane przed czterema krokami oryginalnego modelu Nagla-Schreckenberga. Symulują one zmiany pasów

ruchu przez kierowców. Pierwszym krokiem jest próba zmiany pasa w celu wykonania manewru wyprzedzania. W drugim kroku samochód wykonuję próbę zmiany pasa w celu ustawienia się na wybranym pasie z którego chce wjechać na skrzyżowanie.

4.2.2 MODEL MIASTA

Model sieci drogowej miasta składa się z węzłów wlotowych, skrzyżowań i połączeń między nimi. Poprzez węzły wlotowe („bramy") samochody pojawiają się i opuszczają symulację systemu. Ruch pojedynczego samochodu odbywa się pomiędzy dwoma węzłami wlotowymi poprzez połączenia (drogi) i skrzyżowania. Każde połączenie implementowane jest jako wielopasmowy model Nagla-Schreckenberga. Jedna tura modelu odpowiada jednej sekundzie w rzeczywistości, a jedna komórka pasa ruchu posiada rozmiar 7,5 metra. Do zasymulowania typowej ulicy dwukierunkowej potrzebne są dwa takie połączenia.

4.2.3 RUCH MIEJSKI

Czas trwania symulacji w systemie Kraksim ustalany jest przez definicje ruchu drogowego. Definicja ta zawiera zbiór informacji o tym jak duża ilość pojazdów ma przejechać przez sieć drogową między dwoma punktami węzłów wlotowych. Symulacja trwa przez cały okres generowania pojazdów w węzłach bram i nie kończy się dopóki ostatni pojazd nie dojedzie do swojej bramy docelowej. Po zakończeniu symulacji następuje zapisanie różnych parametrów ruchu miejskiego zebranych w trakcie jej działania. Zadaniem użytkownika jest dobranie tak ilości pojazdów poruszających się między parami bram, tak aby ruch tych samochodów w symulacji odpowiadał ruchowi drogowemu, który interesuje użytkownika systemu.

4.2.4 ALGORYTMY STEROWANIA SYGNALIZACJĄ ŚWIETLNĄ

Algorytmy sterowania sygnalizacją świetlną są jednym z dwóch sposobów optymalizacji ruchu drogowego w systemie. Wraz ze zmianą algorytmu sterowania można zaobserwować wyraźne zmiany w charakterystyce i statystykach ruchu w systemie.

4.2.4.1 Stała konfiguracja

Użytkownik systemu umieszcza w pliku konfiguracyjnym planu miasta informacje o działaniu sygnalizacji świetlnej. Dla każdego skrzyżowania należy dostarczyć opis każdej kolejnej fazy sygnalizacji świetlnej. Opis ten zawiera czas trwania fazy i jaki rodzaj światła będzie palił się na każdym z pasów ruchu. Definicja kolejnych faz tworzy plan, który będzie uruchamiany cyklicznie w czasie symulacji.

4.2.4.2 SOTL

Podstawowa wersja algorytmu *Sotl-request* [19] opiera się o zliczanie wartości K_i. Gdy świeci jest czerwone światło wartość ta zwiększana jest o ilość samochodów na drodze. Kiedy wartość ta przekroczy pewien ustalony poziom θ następuje zmiana fazy sygnalizacji świetlnej. Na ulicy, dla której przekroczono ten poziom, zapala się zielone światło. Na pozostałych zaś pali się czerwone.

Algorytm z pierwszym usprawnieniem działania nosi nazwę *Sotl-phase*. Wprowadza ono parametr minimalnego czasu palenia się jednej fazy sygnalizacji świetlnej. Dopóki licznik tur nie doliczy do wartości tego parametru, nie może dojść do zmiany faz sygnalizacji świetlnej.

Trzecia wersja algorytmu nazwana została *Sotl-platoon*. Usprawnienie w niej też przeciwdziała częstym zmianom świateł na skrzyżowaniu. Kiedy poziom θ decydujący o zmianie fazy sygnalizacji świetlnej zostanie osiągnięty dla jednej z dróg na skrzyżowaniu, sprawdzany jest jeszcze jeden warunek. Jeżeli dla drogi z zielonym światłem w odległości mniejszej niż ω wykryto zbliżający się pojazd, również nie następuje zmiana świateł na skrzyżowaniu. Oczywiście taki warunek mógłby doprowadzić do sytuacji, w której nigdy nie doszłoby do zmiany świateł na sygnalizatorach. W metodzie tej wprowadzono kolejny parametr μ ilości samochodów stojących na ulicy z czerwonym światłem, po przekroczeniu którego powyższy warunek nie jest brany pod uwagę przy zmianie sygnalizacji świetlnej.

W systemie Kraksim zaimplementowana jest druga wersja algorytmu SOTL (sot-phase). Wprowadzono też do niej kilka modyfikacji. Przede wszystkim obliczanie wartości K_i odbywa się nie dla całej drogi, ale dla pojedynczego pasa

ruchu. Wartość ta nie jest też wyliczana z całej jego długości, ale tylko dla pewnej strefy znajdującej się przy skrzyżowaniu.

4.2.4.3 RL

W pracy Intelligent traffic Light Control [20] przedstawiono implementację podejścia uczenia ze wzmocnieniem do kontroli sygnalizacji świetlnej. Celem tego podejścia jest uzyskanie jak najlepszego wyboru fazy sygnalizacji świetlnej dla danego skrzyżowania na podstawie aktualnych informacji o ruchu.

Każdy samochód na danej ulicy może znajdować się w danym stanie s, gdzie s jest parą [*cell, dir*]. Gdzie *cell* jest pozycją w komórce jednej z jezdni modelu Nagla-Schreckenberga, a *dir* oznacza pas ruchu. W celu jak najlepszej oceny sytuacji wokół skrzyżowania dla każdego samochodów wprowadza się funkcję kosztu R(s, s'), który oznacza karę za niewykonanie ruchu w danej turze. R(s, s') jest równe 1 gdy s = s' i 0 gdy s ≠ s'.

Wybór aktualnie najlepszej fazy sygnalizacji świetnej A_j^{opt} dla danego skrzyżowania j odbywa się na podstawie maksymalnej wartości oceny:

$$A_j^{opt} = \arg\max_{A_j} \sum_{i \in M_j} \sum_{s \in queue_i} Q(s, red) - Q(s, green)$$

Pierwsze sumowanie odbywa się po wszystkich sygnalizatorach M_j skrzyżowania j. Następne sumuje się po ocenach stanów wszystkich samochodów stojących w kolejce $queue_i$ przed sygnalizatorem i.

Ocena stanu odbywa się poprzez obliczenie wartości funkcji Q(s, l). Określa ona wielkość kary na samochód znajdujący się w stanie s przed sygnalizatorem gdy znany jest kolor l ∈ { $red, green$} światła na sygnalizatorze.

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

Sumowanie odbywa się po wszystkich stanach s' w jakich znaleźć się może samochód w kolejnej turze. Prawdopodobieństwo P(l, s, s') że samochód znajdzie się w stanie s' przy danym świetle l, mnożone jest przez funkcję kary dla tego stanu. Parametr $0 < \gamma < 1$ reguluje ważność wpływów krótko lub długoterminowych.

Funkcja V(s) jest równa:

$$V(s) = \sum_{l \in \{red, green\}} P(l|s)Q(s,l)$$

Reprezentuje ona prawdopodobieństwo że dla stanu s światło będzie miało kolor zielony lub czerwony. P(l|s) w powyższym wzorze jest liczbą określającą prawdopodobieństwo palenia się światła l dla samochodu, który może znaleźć się w kolejnej turze w stanie s.

Tak samo jak w przypadku algorytmu SOTL implementacja algorytmu nie bierze pod uwagę wszystkich samochodów znajdujących się na drodze przy skrzyżowaniu. Badane są tylko samochody znajdujące się w pewnej długości strefie przed skrzyżowaniem.

4.2.4.4 OptAPO

Algorytm Optimal Asynchronous Partial Overlay – OptAPO [21] zakłada że na każdym ze skrzyżowań działają sygnalizatory posiadające tylko dwie fazy świetlne. Fazy te umożliwiają przejazd na jednym z kierunków północ-południe, wschód-zachód.

Działanie algorytmu polega na wyborze dla skrzyżowania jednego z tych dwóch ustawień sygnalizatorów. Dokonywany jest podział skrzyżowań na połączone drogami grupy, które tworzą jeden kanał przepływu wzdłuż jednego z kierunków. Na całej długości kanału uruchamiane jest zielone światło.. Celem stworzenia takich kanałów jest zapewnienie niezakłóconego przepływu pojazdów przez jak najdłuższe odcinki sieci drogowej.

Każde skrzyżowanie kontrolowane jest przez agenta, który zlicza ilość pojazdów zbliżającą się do skrzyżowania z każdego z kierunków. Na podstawie tych danych i informacji uzyskanych od sąsiednich agentów obliczana jest wartości funkcji kosztu dla agenta zarządzającego tym skrzyżowaniem. Wzór na funkcję kosztu to suma wartości obliczanych dla każdego z czterech sąsiadujących agentów/skrzyżowań. Wartość ta jest równa:

- 0 - jeżeli zarówno dla tego agenta jak i sąsiedniego jest uruchomiona faza sygnalizacji świetlnej dla kierunku z większym natężeniem ruchu.
- Ilość pojazdów jadąca od sąsiedniego skrzyżowania podzielona przez wszystkie pojazdy zbliżające się do badanego skrzyżowania. Taka wartość

zwracana jest gdy badany agent ma fazę świetlną ustawioną zgodnie z większym natężeniem ruchu, a sąsiedni przeciwną w stosunku do większego natężenia na swoim skrzyżowaniu.

- Podwojona ilość pojazdów jadąca od sąsiedniego skrzyżowania podzielona przez wszystkie pojazdy zbliżające się do badanego skrzyżowania. Wartość ta zwracana jest, gdy zielone światło pali się dla kierunku przeciwnego niż dla kierunku z większym natężeniem ruchu.

Algorytm OptAPO poprzez mediację agentów ustala jak ustawić kierunki faz świetlnych na sygnalizatorach tak, aby minimalizować globalną funkcję kosztu obliczaną jako suma po wszystkich agentach.

4.2.5 PODSYSTEM ZNAKÓW O ZMIENNYM TEKŚCIE

Podsystem znaków o zmiennym tekście jest drugim elementem systemu służącym do optymalizacji ruchu drogowego. Znaki te, ustawione przy drogach lub zawieszone nad nimi, służą do prezentowania kierowcom różnych aktualnych informacji, które mogą okazać się dla nich przydatne. Znaki te prezentują informacje o sytuacji pogodowej, objazdach czy utrudnieniach w ruchu. Z punktu widzenia optymalizacji przepływu pojazdów najbardziej wartościowe jest prezentowanie kierowcom takich informacji, na podstawie których w sposób świadomy będą oni podejmować decyzje o wyborze tras przejazdu. Takimi informacjami są na przykład przewidywany czas przejazdu daną ulicą czy informacja o poziomie ruchu jaki napotka kierowca dalej jadąc tą trasą.

W systemie Kraksim idea ta zaimplementowana jest w sposób uproszczony. Kierowcy otrzymują na bieżąco informacje o średnim czasie przejazdu przez wszystkie drogi. Na tej podstawie algorytmem Dijkstry obliczana jest najkrótsza trasa do punktu docelowego. Tą trasą dalej będzie poruszał się kierowca. W ten sposób symulowane są znaki o zmiennym tekście. Aby urealnić ten mechanizm jako parametru symulacji podaje się jaki procent kierowców i jak często ma dokonywać próby przeliczenia i ponownego wybrania najkrótszej trasy. Oznaczać to ma, że kierowcy co pewien czas uznają informacje o czasach przejazdu za użyteczne i wykorzystują je prowadząc samochód.

Taka implementacja znacznie lepiej jednak reprezentuje symulację wyznaczania trasy przejazdu dla kierowców poprzez systemy nawigacji GPS.

5 MODUŁ WYKRYWANIA I ZAPOBIEGANIA ZAGĘSZCZENIOM RUCHU DROGOWEGO W SYSTEMIE KRAKSIM

5.1 CEL PRAC

Celem projektu jest stworzenie efektywnego algorytmu przewidującego zagęszczenia w ruchu drogowym, co umożliwia jego reorganizację, a w konsekwencji poprawę jego płynności. Zaproponowany algorytm identyfikacji zagęszczeń został zaimplementowany jako moduł systemu symulacyjnego Kraksim. Dzięki temu możliwe jest pokazanie jego skuteczności dla wielu rodzajów ruchu, różnych sieci drogowych i sposobów sterowania sygnalizacją świetlną. Część implementacji stanowi integrację nowego modułu z podsystemem znaków o zmiennym tekście i podsystemem sterowania sygnalizacją świetlną. Dzięki niej możliwe jest wykorzystanie wyników przewidywania do zapobiegania powstawaniu zagęszczeń.

Integracja z systemem symulacyjnym umożliwia podanie odpowiedzi na wiele pytań związanych z przewidywaniem i zapobieganiem korkom ulicznym. Praca da odpowiedź na pytanie, jakie typy informacji o ruchu drogowym są użyteczne przy przewidywaniu zagęszczeń. Następnie, czy sposób zdefiniowania zagęszczenia w ruchu drogowym może mieć wpływ na wyniki prognozowania. Określone zostanie również, na jaki okres czasu możliwe jest prognozowanie ruchu w gęstej sieci drogowej miast.

5.2 WYMAGANIA

Moduł predykcji musi być uniwersalny. Powinien pozwalać na użycie różnych definicji zagęszczenia ruchu. Możliwe musi być również wykorzystanie w procesie przewidywania wielu różnych typów informacji opisujących ruch drogowy. Kolejne wymaganie stanowi możliwość wygładzania danych opisujących sytuację na drogach poprzez użycie średniej kroczącej. Konfigurowalny powinien być również okres w jakim te dane są zbierane i jaki ich zakres używany jest w procesie predykcji.

Algorytm powinien mieć możliwość przewidywania zagęszczeń na różną ilość okresów czasu w przód.

W oparciu o dane uzyskane w procesie przewidywania moduł powinien próbować zapobiegać ich powstawaniu, wywierając wpływ na elementy systemu symulacyjnego sterujące ruchem tj. sygnalizację świetlną i znaki o zmiennym tekście. Moduł powinien mieć możliwość zapobiegania zagęszczeniom poprzez oba moduły jednocześnie jak i poprzez każdy z osobna.

Podstawowym wymaganiem stawianym przed implementacją jest umożliwienie użytkownikowi zdefiniowania w głównym pliku konfiguracyjnym tego, czy nowy moduł będzie wykorzystywany w symulacji, czy też nie.

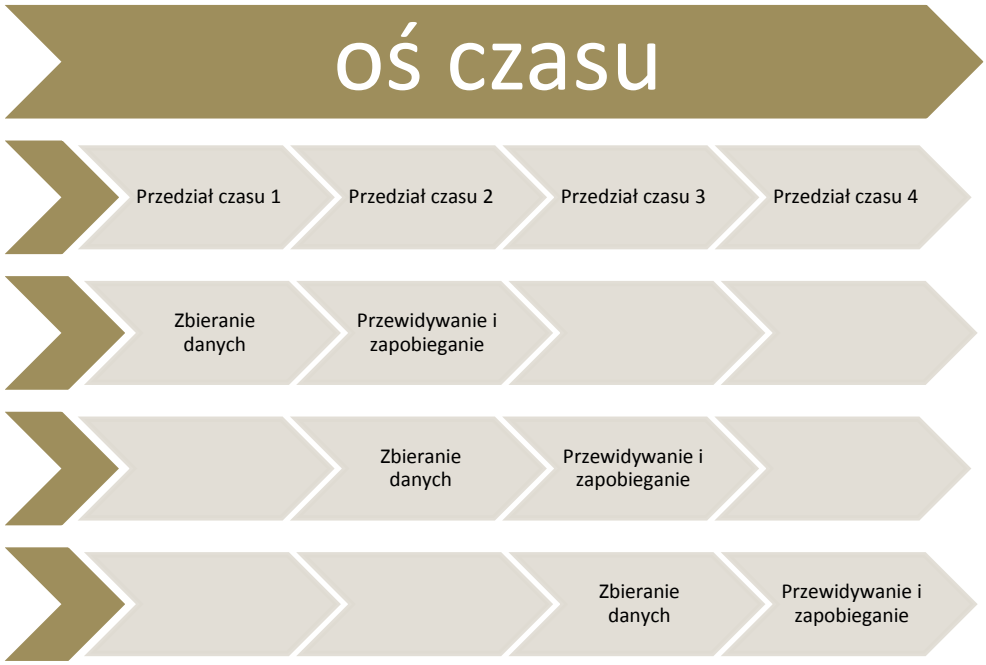
Sam moduł powinien korzystać z oddzielnego pliku konfiguracyjnego, gdzie użytkownik będzie mógł wyspecyfikować wszystkie parametry modułu. W celu określenia skuteczności algorytmu konieczne jest zapisywanie do pliku wyników jego działania. Powinno być możliwe określenie w którym momencie symulacji dojdzie do zapisania statystyk modułu.

Ponieważ zaproponowana metoda wykorzystuje zewnętrzną bibliotekę algorytmów regresji, możliwe powinno być użycie wielu z nich.

5.3 KONCEPCJA DZIAŁANIA MODUŁU PREDYKCJI

Metoda przewidywania działa w dwóch etapach. W pierwszej następuje zbieranie informacji o ruchu drogowym. Po zakończeniu tego etapu, w oparciu o zdobyte dane, nastąpi moment uczenia algorytmu. Gdy uczenie się zakończy, rozpocznie się drugi etap, w którym wykonywane jest już przewidywanie zagęszczeń i zapobieganie ich powstawaniu. Ponieważ ruch uliczny zmienia się w czasie, potrzebna jest ponowna inicjalizacja algorytmu predykcji nowymi danymi. Aby zapewnić ciągłość działania algorytmu pierwszy i drugi etap działania muszą nakładać się na siebie w jednym przedziale czasu. Mechanizm ten przedstawia rysunek 5.1.

W pierwszym okresie następuje tylko proces zbierania danych. W drugim okresie przewidywane są zagęszczenia na podstawie danych zapisanych z okresu pierwszego i zbierane są dane którymi zostanie nauczony algorytm przewidywania w okresie trzecim. W okresie trzecim oprócz przewidywania następuje zapisywanie danych, na podstawie których będzie następować przewidywanie w okresie następnym.



Rysunek 5.1 Cykl działania modułu predykcji

Dane uczące zbierane w pierwszym etapie działania algorytmu. Informacje przydatne do przewidywania zagęszczeń są pobierane z systemu symulacyjnego i odnoszą się do skrzyżowań lub dróg je łączących. Przyjęto, że jedna ulica składa się z dwóch dróg, jednej prowadzącej ruch w jednym kierunku i drugiej prowadzącej w kierunku przeciwnego skrzyżowania. Dane te są pobierane co pewien stały przedział czasu i opisują cechy przepływu pojazdów i infrastruktury drogowej. Do zbioru tych danych, dla dróg możemy zaliczyć:

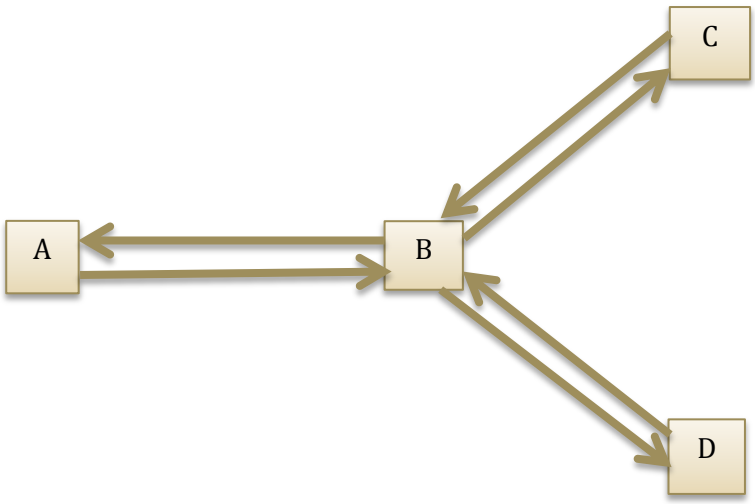
- gęstość samochodów na drodze – ilość pojazdów na drodze podzieloną przez iloczyn długości drogi przez ilość pasów ruchu,
- liczbę samochodów na drodze,
- liczbę samochodów, która opuściła drogę poprzez skrzyżowanie w ostatnim okresie,
- liczbę samochodów, która wjechała na drogę ze skrzyżowania,
- średni czas przejazdu samochodów zmierzony w ostatnim okresie.

W przypadku skrzyżowań będzie to:

- numer fazy sygnalizacji świetnej włączonej pod koniec okresu,
- liczba sekund jaką działa aktualna faza sygnalizacji świetnej,
- liczba sekund jaką będzie działać aktualna faza sygnalizacji świetlnej.

Dane te zebrane w jednym okresie dla wszystkich dróg i skrzyżowań **tworzą opis stanu sieci drogowej w danej chwili**. Długość trwania takiego okresu może być różna i będzie możliwa do ustalenia jako parametr modułu predykcji. Biorąc pod uwagę, że symulacji będzie podlegać miejska sieć drogowa, wartość tego parametru może wynosić od 15 sekund do kilku minut.

Przykład. Dla prostej sieci drogowej przedstawionej na rysunku 5.2. dane opisujące sieć drogową pobierane są dla 4 skrzyżowań: A, B, C, D i 6 dróg: AB, BA, BC, CB, BD i DB. Przykłady prezentowane w dalszej części rozdziału będą opierać się o prosty schemat dróg z rysunku 5.2. Przyjęto w nich, że do zbioru danych uczących należy gęstość samochodów na drodze oznaczona jako [GS] i średni czas przejazdu przez daną drogę oznaczony jako [CP].



Rysunek 5.2 Prosta sieć drogowa

Pomiary stanu sieci drogowej zapisywane są w systemie w **tabeli historii**. Zawiera ona informacje o tym, jak wyglądał stan sieci drogowej w kolejnych momentach symulacji. Dane w tabeli zapisywane są przez dłuższy okres, który może trwać od kilku godzin do wielu dni. Ograniczeniem jest tutaj możliwość przetworzenia przez algorytm predykcji odpowiednio dużej tabeli historii.

Przykład. Poniższa tabela 5.1. jest tabelą historii dla sieci drogowej z rysunku 5.2. zawierającą wartości gęstości samochodów na drodze [GS] i średnich czasów przejazdu [CP], zmierzonych w czterech okresach czasu trwających po 60 sekund.

Tabela 5.1 Przykładowa tabela historii

	BA[GS]	BA[CP]	AB[GS]	AB[CP]	CB[GS]	CB[CP]	…	DB[GS]	DB[CP]	…	…
t_i = 240	0.3	14	0.1	24	0.2	23	…	0.1	15	…	…
t_{i-1}= 180	0.2	23	0.3	25	0.4	26	…	0.2	16	…	…
t_{i-2} = 120	0.4	12	0.2	15	0.1	27	…	0.2	13	…	…
t_{i-3}=60	0.1	34	0.4	23	0.3	32	…	0.3	18	…	…

Wygładzanie danych. Dane uzyskiwane z dróg w gęstej sieci ulic są bardzo zaburzone przez działanie sygnalizacji świetlnej. Okres zbierania informacji jest bardzo krótki, na przykład ilość samochodów, jakie opuściły ulicę poprzez skrzyżowanie, będzie wynosić zero, jeżeli paliło się czerwone światło przez całą długość trwania okresu zbierania danych. Aby dane lepiej oddawały prawdziwy obraz ruchu drogowego, są one wygładzane przed umieszczeniem w tabeli historii. Do wygładzanie używana jest metoda prostej średniej kroczącej. Wygładzaniu podlegają wszystkie rodzaje danych zbierane dla dróg.

Uczenie algorytmu przewidywania. [Algorytm 1] Gdy zakończy się okres zbierania i zapisywania danych w tablicy historii, uruchamiany jest proces uczenia algorytmu przewidywania. Dla każdej drogi i skrzyżowania, i dla każdego typu danych (spośród wybranych na poziomie konfiguracji), które je opisuje, uczona jest nowa instancja algorytmu regresji. Algorytm ten ma przewidywać wartość tego typu danych w najbliższym okresie, na podstawie danych z okresów poprzednich. Dane uczące tworzą tabelę transakcji.

Algorytm 1. Pseudokod metody tworzącej modele regresji

```
function stwórz modele regresji (historia)
---- foreach droga do
-------- foreach parametr drogi do
---------- tabela transakcji = stwórz tabelę transakcji (historia, droga, parametr drogi)
---------- model regresji = stwórz model regresji (tabela transakcji)
---- foreach skrzyżowanie do
-------- foreach parametr skrzyżowania do
---------- tabela transakcji = stwórz tabelę transakcji (historia, skrzyżowanie, parametr
skrzyżowania)
---------- model regresji = stwórz model regresji (tabela transakcji)
```

Tabela transakcji. Tabela transakcji zawiera kolejne instancje uczące dla algorytmu regresji. Transakcja składa się ze zmiennej zależnej i zbioru zmiennych niezależnych. Zmienną zależną stanowi jeden z parametrów drogi lub skrzyżowania zmierzony w czasie t_i, gdzie i oznacza numer kolejnego pomiaru stanu sieci drogowej.

Zmiennymi niezależnymi są wartości parametrów dróg i skrzyżowań, dróg „sąsiednich". Dla skrzyżowań za sąsiednie przyjęto tylko drogi bezpośrednio wprowadzające ruch na to skrzyżowanie. Dla dróg, za sąsiednie należy uznać, te drogi i skrzyżowania, przez które prowadzić będzie ruch pojazdów w kierunku drogi, którą reprezentuje badana zmienna zależna. Jest to wynikiem założenia, że strumień pojazdów z tych dróg i skrzyżowań pojawi się po pewnym czasie na drodze, której parametr staramy się przewidzieć. Istnieć może więc zależność między parametrami opisującymi te drogi i skrzyżowania. Odległość „sąsiadów" od badanej drogi, mierzona jest ilością skrzyżowań o jakie oddalone są od niej elementy sieci drogowej. Wartości parametrów opisujących „sąsiednie" elementy sieci drogowej stanowią wartości zmiennych niezależnych w tablicy transakcji. Dodatkowo za zbioru zmiennych niezależnych zalicza się wartości parametrów opisujących drogę lub skrzyżowanie którego parametr jest przewidywany.

Wysokość historii (h) określa jak odległe w czasie pomiary (wartości parametrów) stanu sieci drogowej brane są pod uwagę przy przewidywaniu.

Przykład. Tabele 5.1. 5.2. 5.3. prezentują sposób budowy tabeli transakcji na podstawie tabeli historii. Przewidywanym parametrem jest wartość gęstości pojazdów [GS] na drodze łączącej skrzyżowania B i A z rysunku 5.2. Na niebiesko zaznaczone są wartości parametru zależnego, a na czerwono wartości parametrów niezależnych. Wysokość historii h w tym przykładzie wynosi dwa. Wartość parametru sąsiedztwa wynosi jeden. Brane są więc pod uwagę tylko najbliższe drogi (skrzyżowania nie uwzględniono).

Tabela 5.2 Tabela historii z przykładową transakcją dla okresu zakończonego w turze 240

	BA[GS]	BA[CP]	AB[GS]	AB[CP]	CB[GS]	CB[CP]	…	DB[GS]	DB[CP]	…	…
t_i = 240	0.3	14	0.1	24	0.2	23	…	0.1	15	…	…
t_{i-1}= 180	0.2	23	0.3	25	0.4	26	…	0.2	16	…	…
t_{i-2} = 120	0.4	12	0.2	15	0.1	27	…	0.2	13	…	…
t_{i-3}=60	0.1	34	0.4	23	0.3	32	…	0.3	18	…	…

Tabela 5.3 Tabela historii z przykładową transakcją dla okresu zakończonego w turze 180

	BA[GS]	BA[CP]	AB[GS]	AB[CP]	CB[GS]	CB[CP]	…	DB[GS]	DB[CP]	…	…
t_i = 240	0.3	14	0.1	24	0.2	23	…	0.1	15	…	…
t_{i-1} = 180	0.2	23	0.3	25	0.4	26	…	0.2	16	…	…
t_{i-2} = 120	0.4	12	0.2	15	0.1	27	…	0.2	13	…	…
t_{i-3}= 60	0.1	34	0.4	23	0.3	32	…	0.3	18	…	…

Tabela 5.4 Przykładowa tablica transakcji

	BA [GS] $[t_i]$	BA [GS] $[t_{i-1}]$	BA [CP] $[t_{i-1}]$	BA [GS] $[t_{i-2}]$	BA [CP] $[t_{i-2}]$	CB [GS] $[t_{i-1}]$	CB [CP] $[t_{i-1}]$	CB [GS] $[t_{i-2}]$	CB [CP] $[t_{i-2}]$	DB [GS] $[t_{i-1}]$	DB [CP] $[t_{i-1}]$	DB [GS] $[t_{i-2}]$	DB [CP] $[t_{i-2}]$
t=240	0.3	0.2	23	0.4	12	0.4	26	0.1	27	0.2	16	0.2	13
t=180	0.2	0.4	12	0.1	34	0.1	27	0.3	32	0.2	13	0.3	18

Proces przewidywania. [Algorytm2] Algorytm może przewidywać wartości parametru definiującego zagęszczenie na jeden lub więcej kroków czasowych w przyszłości. Kroki czasowe są liczone od zera. Jeżeli przewidywany jest j-ty okres czasowy, gdzie j > 0, potrzebne jest przewidzenie wartości parametrów z okresów poprzedzających ostatecznie prognozowany okres. W pętli przewidywane są więc wszystkie parametry sieci drogowej dla okresów $k \in \{0, 1, \dots j-1\}$. Zbiór przewidywań dla każdego kolejnego okresu umieszczany jest w tabeli historii. Pozwala to na zbudowanie transakcji przewidujących na następny okres.

Algorytm 2. Pseudokod metody wykonującej właściwy proces przewidywania

```
function wykonaj przewidywanie sytuacji na drogach (historia, przewidywany parametr dróg)
---- while ilość kroków czasowych >  0 do
-------- tymczasowy stan sieci drogowej = new tymczasowy stan sieci drogowej
-------- foreach droga do
---------- foreach parametr drogi do
---------- transakcja przewidująca = stwórz transakcję przewidującą (historia, droga,
parametr drogi)
---------- przewidziany parametr = wykonaj przewidywanie (model regresji, transakcja
przewidująca)
---------- tymczasowy stan sieci drogowej.dodaj(przewidziany parametr)
-------- foreach skrzyżowanie do
---------- foreach parametr skrzyżowania do
---------- transakcja przewidująca = stwórz transakcję przewidującą (historia, skrzyżowanie,
parametr skrzyżowania)
---------- przewidziany parametr = wykonaj przewidywanie (model regresji, transakcja
przewidująca)
---------- tymczasowy stan sieci drogowej.dodaj(przewidziany parametr)

-------- historia.dodaj(tymczasowy stan sieci drogowej)
-------- ilość kroków czasowych = ilość kroków czasowych - 1

---- foreach droga do
-------- transakcja przewidująca = stwórz transakcję przewidującą (historia, droga,
przewidywany parametr dróg)
-------- przewidziany parametr = wykonaj przewidywanie (model regresji, transakcja
przewidująca)

---- usuń z tabeli historii przewidziane tymczasowo stany sieci drogowej
```

Przykład. Przewidywanie na jeden krok czasowy w przód. Tabela 5.5. prezentuje zaznaczony przewidywany parametr gęstości samochodów dla drogi BA. Na czerwono zaznaczono wartości parametrów na podstawie których będzie odbywać się przewidywanie. Tabela 5.6. zawiera zbudowaną z tych parametrów transakcję testową. Dzięki ewaluacji transakcji testowej na modelu regresji otrzymamy przewidywaną wartość parametru.

Tabela 5.5 Tabela historii z przykładową transakcją testową przewidującą wartość w okresie kończącym się w turze nr 300

	BA[GS]	BA[CP]	AB[GS]	AB[CP]	CB[GS]	CB[CP]	…	DB[GS]	DB[CP]	…	…
t_{i+1}= 300	???										
t_i = 240	0.3	14	0.1	24	0.2	23	…	0.1	15	…	…
t_{i-1}= 180	0.2	23	0.3	25	0.4	26	…	0.2	16	…	…
t_{i-2} = 120	0.4	12	0.2	15	0.1	27	…	0.2	13	…	…
t_{i-3}=60	0.1	34	0.4	23	0.3	32	…	0.3	18	…	…

Tabela 5.6 Przykładowa transakcja przewidująca na jeden okres w przód

	BA [GS] $[t_i]$	BA [GS] $[t_{i-1}]$	BA [CP] $[t_{i-1}]$	BA [GS] $[t_{i-2}]$	BA [CP] $[t_{i-2}]$	CB [GS] $[t_{i-1}]$	CB [CP] $[t_{i-1}]$	CB [GS] $[t_{i-2}]$	CB [CP] $[t_{i-2}]$	DB [GS] $[t_{i-1}]$	DB [CP] $[t_{i-1}]$	DB [GS] $[t_{i-2}]$	DB [CP] $[t_{i-2}]$
t = 300	???	0.3	14	0.2	23	0.2	23	0.4	26	0.1	15	0.2	16

Przykład. Przewidywanie na dwa kroki czasowe w przód. Aby przewidzieć wartości parametrów średniej gęstości samochodów dla okresu z tury 360 potrzebne jest przewidzenie parametrów z okresu wcześniejszego [Tabela 5.7.]

Tabela 5.7 Tabela historii przed uruchomieniem algorytmu przy przewidywaniu na dwa okresy w przód.

	BA[GS]	BA[CP]	AB[GS]	AB[CP]	CB[GS]	CB[CP]	…	DB[GS]	DB[CP]	…	…
t_{i+2}=360	???										
t_{i+1}= 300	?	?	?	?	?	?	…	?	?	…	…
t_i = 240	0.3	14	0.1	24	0.2	23	…	0.1	15	…	…
t_{i-1}= 180	0.2	23	0.3	25	0.4	26	…	0.2	16	…	…
t_{i-2} = 120	0.4	12	0.2	15	0.1	27	…	0.2	13	…	…
t_{i-3}=60	0.1	34	0.4	23	0.3	32	…	0.3	18	…	…

Następuje więc przewidywanie wszystkich parametrów dróg i skrzyżowań dla okresu z tury 300. Tabela 5.7. zawiera transakcję przewidującą jeden z parametrów pośrednich potrzebnego do budowy transakcji przewidujących parametry dla okresu z tury 360.

Tabela 5.8 Przykładowa transakcja przewidująca na jeden okres w przód. Przewidująca jedną z wartości okresu pośredniego.

	BA [GS] $[t_i]$	BA [GS] $[t_{i-1}]$	BA [CP] $[t_{i-1}]$	BA [GS] $[t_{i-2}]$	BA [CP] $[t_{i-2}]$	CB [GS] $[t_{i-1}]$	CB [CP] $[t_{i-1}]$	CB [GS] $[t_{i-2}]$	CB [CP] $[t_{i-2}]$	DB [GS] $[t_{i-1}]$	DB [CP] $[t_{i-1}]$	DB [GS] $[t_{i-2}]$	DB [CP] $[t_{i-2}]$
t = 300	?	0.3	14	0.2	23	0.2	23	0.4	26	0.1	15	0.2	16

Tabela historii 5.8 zawiera przewidziane parametry pośrednie dla wszystkich dróg i skrzyżowań (tura 300). W tej chwili możliwe jest już prognozowanie parametrów odległych o dwa kroki czasowe (tura 360).

Tabela 5.9 Tabela historii z przewidzianymi wszystkimi wartościami parametrów pośrednich, wymaganych do zbudowania transakcji przewidujących na okres 360.

	BA[GS]	BA[CP]	AB[GS]	AB[CP]	CB[GS]	CB[CP]	…	DB[GS]	DB[CP]	…	…
t_{i+2}=360	???										
t_{i+1}= 300	? =0.3	? = 15	?=0.4	?=25	? =0.2	?=34	…	?=0.2	?=17	…	…
t_i = 240	0.3	14	0.1	24	0.2	23	…	0.1	15	…	…
t_{i-1}= 180	0.2	23	0.3	25	0.4	26	…	0.2	16	…	…
t_{i-2} = 120	0.4	12	0.2	15	0.1	27	…	0.2	13	…	…
t_{i-3}=60	0.1	34	0.4	23	0.3	32	…	0.3	18	…	…

Tabela 5.9. prezentuje transakcję prognozującą parametr drogi na dwa kroki czasowe w przód. Jak widać zawiera ona przewidziane wcześniej wartości parametrów na jeden krok czasowy w przód.

Tabela 5.10 Transakcja przewidująca na dwa okresy w przód, Zawierajaca przewidziane wartości pośrednie.

	BA [GS] $[t_i]$	BA [GS] $[t_{i-1}]$	BA [CP] $[t_{i-1}]$	BA [GS] $[t_{i-2}]$	BA [CP] $[t_{i-2}]$	CB [GS] $[t_{i-1}]$	CB [CP] $[t_{i-1}]$	CB [GS] $[t_{i-2}]$	CB [CP] $[t_{i-2}]$	DB [GS] $[t_{i-1}]$	DB [CP] $[t_{i-1}]$	DB [GS] $[t_{i-2}]$	DB [CP] $[t_{i-2}]$
t = 360	???	0.3	15	0.3	14	0.2	34	0.2	23	0.2	17	0.1	15

Definicja zagęszczenia ruchu. W pliku konfiguracyjnym użytkownik specyfikuje jaki typ parametru będzie określał korek. Algorytm przewiduje wartość tego parametru dla wszystkich dróg w symulacji. Użytkownik systemu określa również jaka wartość tego parametru będzie oznaczać zagęszczenie. Jeżeli system wykonując proste porównanie, uzna że ma do czynienia z zagęszczeniem na jednej z ulic, to informacja o tym zostanie zapisana w systemie.

Statystyki przewidywania. W systemie zostanie również zapisana rzeczywista wartość parametru określającego korek uliczny i informacja o tym czy określna ona zagęszczenie (na ulicy w symulacji pojawiło się zagęszczenie w ruchu). Obie informacje będą dalej przechowywane, aż konieczne będzie ponowne nauczenie algorytmu predykcji. W tym momencie nastąpi obliczenie częściowych informacji o skuteczności przewidywania korków jak i skuteczności przewidywania samego parametru charakteryzującego ruch. W przypadku zagęszczeń mierzona jest procentowa poprawność przewidywania, jak i błędy

pierwszego i drugiego rodzaju, a także poprawność przewidzenia, że nie wystąpiło zagęszczenie.

W przypadku przewidywania dokładnej wartości parametru określającego korek, dokładność przewidywania określana jest poprzez obliczanie średniego absolutnego błędu procentowego (MAPE – ang. mean absolute percentage error)

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{A_i - P_i}{A_i} \right|$$

Liczba n oznacza ilość wszystkich obserwacji parametru charakteryzującego zagęszczenie. A_i to rzeczywista, zmierzona w symulacji, wartość tego parametru, a P_i to przewidziana przez moduł wartość.

5.4 ZAPOBIEGANIE WYSTĘPOWANIA ZAGĘSZCZEŃ W RUCHU DROGOWYM

Informacje o korkach, które pojawią się w najbliższej przyszłości na drogach, można w prosty sposób wykorzystać do przeciwdziałania ich powstawaniu. Wystarczy odpowiednio wykorzystać te dane w systemach mających wpływ na sytuację na drogach.

5.4.1 WSPOMAGANIE SYSTEMU SYGNALIZACJI ŚWIETLNEJ

Moduł sterujący sygnalizacją świetlną w systemie Kraksim podejmuje decyzję o wyborze odpowiedniej fazy sygnalizacji świetnej na podstawie informacji uzyskanych z modułu oceny. Moduł oceny, badając sytuację na danym pasie ruchu drogi, przypisuje mu pewną wartość, mówiącą o tym, jak bardzo konieczne jest zapalenie zielonego światła dla tego pasa. Sterownik sygnalizatora wybiera fazę sygnalizacji dla pasa ruchu z największą wartością oceny.

Ponieważ moduł oceny operuje tylko na aktualnych danych, można ocenić, że nie jest wystarczająco dokładny. Możliwe jest więc, że posiadając wiedzę o możliwości pojawienia się zagęszczenia na drodze, będzie możliwe niedopuszczenie do pojawienia się korka ulicznego, poprzez wybór innej fazy sygnalizacji świetlej, która nie dopuści do jego powstania.

Jeżeli moduł predykcji przewiduje pojawienie się zatoru dla danej drogi, to następuje przemnożenie wartości ewaluacji każdego pasa tej drogi przez wartość parametru zapobiegania.

Faz świetlnych zapalających zielone światło dla pasów danej drogi może być wiele. Tak samo jedna faza świetlna może zapalać zielone światło na kilku drogach prowadzących do jednego skrzyżowania. Rozwiązanie to więc może wiele zepsuć w sterowaniu ruchem na skrzyżowaniu i najlepiej będzie się sprawdzać na skrzyżowaniach, gdzie jedna faza sygnalizacji świetlnej zapala zielone światło dla wszystkich pasów ruchu jednej drogi.

5.4.2 INFORMACJE PRZEKAZYWANE KIEROWCY

Kierujący pojazdem mogą otrzymywać informacje o niepożądanych sytuacjach na drodze na różne sposoby. Należą do nich: tablice świetlne ze zmiennym tekstem, stacje radiowe, CB radio, urządzenia nawigacji samochodowej GPS. Te ostatnie mają tą właściwość, że mogą automatycznie podawać zmienioną postać trasy kierowcy.

W systemie Kraksim jest już zaimplementowany system, który symuluje wykorzystywanie przez kierowców informacji o aktualnym czasie przejazdu. Możliwe jest zmodyfikowanie tego rozwiązania. Jeżeli przewidzimy, że na danej ulicy pojawi się korek, możemy dla niej zwielokrotnić szacowany czas przejazdu o pewien współczynnik. Prezentowana kierowcom wartość znacznie wzrośnie. Ci z nich, którzy uznają, że czas przejazdu tą ulicą jest za długi, mogą postanowić ją ominąć i możliwe, że nie pojawi się zagęszczenie w ruchu drogowym.

5.5 KONFIGUROWALNE PARAMETRY MODUŁU

W pliku konfiguracyjnym modułu należy określić wartości parametrów wymaganych do jego działania. Są to ogólne parametry modułu:

- `outputMainFolder` – ścieżka do katalogu w którym mają zostać zapisane wyniki działania modułu;

- `writeDataSetToFile` – przyjmuje wartości true lub false. Określa, czy zbiory uczące algorytmy regresji mają zostać zapisane do późniejszej analizy w środowisku graficznym oprogramowania Weka;
- `worldStateUpdatePeriod` – oznaczająca ilość tur symulacji co którą moduł będzie zbierał statystyki o sytuacji w sieci drogowej. Informacje te zostaną zapisane jako nowe pole w tablicy historii;
- `timeSeriesUpdatePeriod` – liczba tur symulacji co którą moduł ma przebudować swój model predykcji na podstawie nowych danych zebranych w tablicy historii;
- `statisticsDumpTime` – czas po jakim moduł ma zapisać wyniki swojego działania w ścieżce podanej w parametrze `outputMainFolder;`
- `averageSize` – liczba określająca rozmiar średniej kroczącej służącej do wygładzania danych.

Parametry związane z przewidywaniem zagęszczeń:

- `maxNumberOfInfluencedLinks` – liczba naturalna określająca jak odległe drogi mogą mieć wpływ na ruch na badanej drodze;
- `maxNumberOfInfluencedTimesteps` – wysokość historii. Parametr określa jak odległe dane brać pod uwagę w procesie przewidywania;
- `predictionSize` – parametr liczony od zera, mówiący na ile kroków w przód ma następować przewidywanie zagęszczeń;
- `regressionDataType` – typ danych pobieranych z systemu, dla którego moduł predykcji będzie prognozował zagęszczenia. Parametr ten może przyjmować jedną z 3 wartości:
  - `carsDensity` – korek uliczny określany jest w oparciu o gęstość samochodów na ulicy,
  - `carsOn` – korek uliczny określany jest w oparciu o liczbę samochodów na ulicy,
  - `durationLevel` – korek określany jest w oparciu o średni czas przejazdu daną drogą.
- `congestionValue` – wartość dla typu danych określonego w regressionDataType, która oznacza poziom określający zaistnienie korka ulicznego;

- `regressionAlgorithm` – algorytm regresji używany w rozwiązaniu. Parametr może przyjmować wartości:
  - `smoreg` – algorytm oparty o maszynę wektorów nośnych,
  - `repTree` – algorytm drzew regresji,
  - `ibk` – algorytm k-najbliższych sąsiadów,
  - `m5p` – algorytm budujący drzewo modeli,
  - `m5rules` – algorytm wykorzystujący budowę drzewa modeli do stworzenia reguł.
- `ibkNeighbours` – parametr dla algorytmu IBk (k-najbliższych sąsiadów), określający ilość sąsiadów branych pod uwagę przy regresji.

Zbiór parametrów związanych z typami danych o ruchu drogowym, które mają zostać użyte do budowy modeli regresji. Każdy z parametrów przyjmuje wartość true lub false:

- `carsDensity` - gęstość samochodów na ulicy;
- `carsOut` – ilość samochodów, która opuściła skrzyżowanie w ostatnim okresie czasu;
- `carsIn` - ilość samochodów, która wjechała na ulicę ze skrzyżowania w ostatnim okresie czasu;
- `carsOn` – zmierzona liczba samochodów znajdująca się na ulicy pod koniec kroku czasowego;
- `durationLevel` – średni czas przejazdu przez ulicę w ostatnim okresie
- `evaluation` – wartość oceny zwracana przez moduł oceny sterujący sygnalizacją świetlną(SOTL, RL);
- `greenDuration` – moduł sterujący sygnalizacją świetlną określa jak długo będzie używana dana faza sygnalizacji świetlnej. Jest to wartość mierzona liczbą tur symulacji;
- `phase` – numer aktualnie używanej fazy sygnalizacji świetlnej;
- `phaseWillLast` – planowany czas palenia się aktualnej fazy sygnalizacji; świetlnej;
- `phaseLast` – liczba tur przez jaką pali się aktualna faza sygnalizacji świetlnej.

Parametry związane z zapobieganiem powstawaniu zagęszczeń:

- `timeTableMultiplier` – liczba rzeczywista większa lub równa jeden. Parametr określający jak bardzo ma zostać zwiększony czas przejazdu daną drogą prezentowany kierowcom przez system znaków o zmiennym tekście. Czas przejazdu szacowany na podstawie aktualnych danych jest przemnażany przez tą wartość w module predykcji. Wartość 1 oznacza brak zapobiegania zagęszczeniom.
- `evaluationMultiplier` - liczba rzeczywista większa lub równa jeden. Parametr określający jak bardzo wzrosnąć ma prawdopodobieństwo wyboru fazy świetlnej zapalającej zielone światło dla drogi z przewidywanym korkiem. Wartość 1 oznacza brak zapobiegania zagęszczeniom.

6 PROJEKT SYSTEMU I JEGO IMPLEMENTACJA

6.1 KRAKSIM

6.1.1 MIEJSCE NOWEGO MODUŁU W ARCHITEKTURZE SYSTEMU

Główną klasą systemu Kraksim, **sterownikiem symulacji**, jest klasa *Simulation* implementująca główną pętlę symulacji. Zawiera ona w sobie instancję *SampleModuleConfiguration*, która przechowuje główne moduły aplikacji biorące udział w działaniu systemu. Symulacja w systemie Kraksim składana jest z kilku rodzajów modułów:

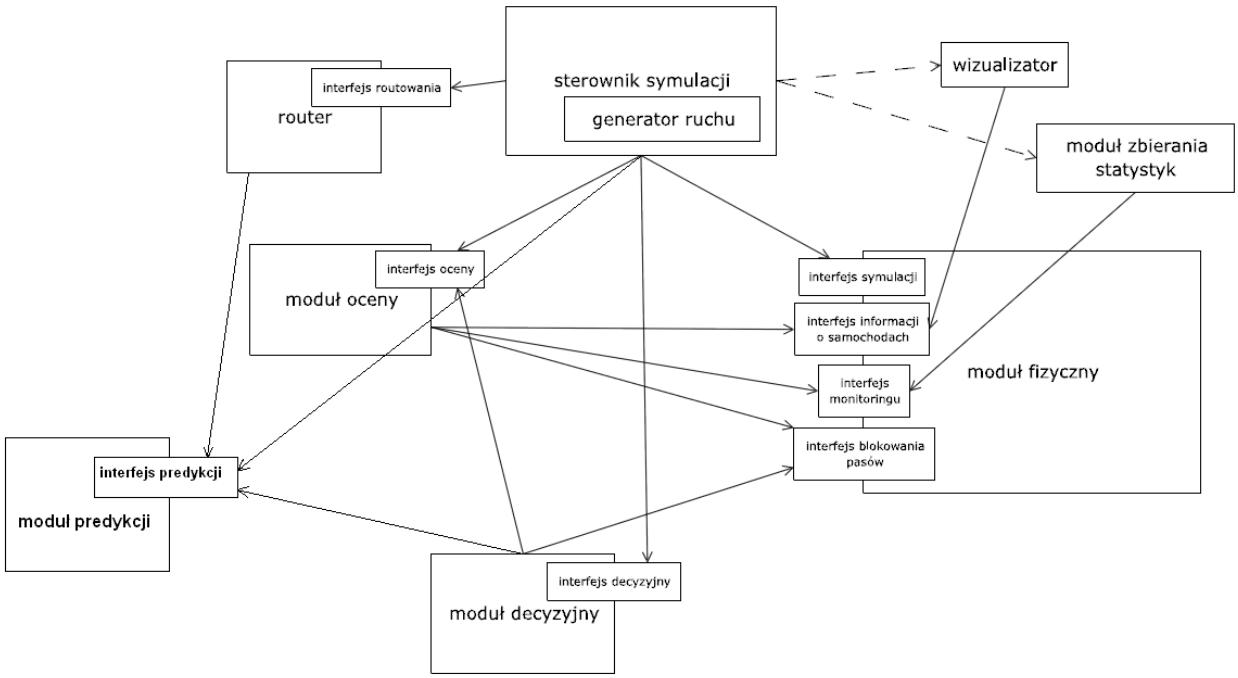
- **Moduł fizyczny** implementujący właściwą symulację ruchu miejskiego. Posiada on jedną implementację znajdującą się w pakiecie: *pl.edu.agh.cs.kraksim.real.*
- **Router** – służący do wyznaczania trasy przejazdu samochodów w symulacji. Posiada dwie implementacje statyczną i dynamiczną. Druga z nich może być wykorzystana do symulacji działania systemu znaków o zmiennym tekście, tak jak zostało to opisane w punkcie 4.2.5 dokumentacji.
- **Wizualizator** – dzięki niemu możliwe jest obserwowanie przebiegu symulacji.
- **Moduł zbierania statystyk** – moduł implementujący zbieranie podstawowych statystyk o ruchu drogowym w systemie. Posiada jedną implementacje *pl.edu.agh.cs.kraksim.ministat*.
- **Moduł oceny** – implementujący algorytmy sterowania sygnalizacją świetlną:
  - RL – pakiet *pl.edu.agh.cs.kraksim.rl.*
  - SOTL – pakiet *pl.edu.agh.cs.kraksim.sotl.*
  - OptApo – pakiet *pl.edu.agh.cs.kraksim.optapo.*
- **Moduł decyzyjny** – moduł właściwie zarządzający sygnalizatorami, decyzje o zmianie faz sygnalizacji świetlnej podejmuje na podstawie informacji uzyskanych z modułu oceny. Posiada dwie implementacje:
  - *pl.edu.agh.cs.kraksim.simpledecision* – współpracującą z algorytmem RL i SOTL.

- *pl.edu.agh.cs.kraksim.dsyncdecision* – współpracującą z algorytmem OptAPO.

Nowym modułem systemu jest moduł predykcji. Wywoływany jest on przez trzy elementy systemu [Rysunek 6.1.]. Pierwszym modułem korzystającym z jego interfejsu jest sterownik symulacji, który wywołuje funkcję *WekaPredictionModule.turnEnded()* w głównej pętli symulacji. Metoda ta odpowiedzialna jest za wykonanie całego procesu działania modułu. Odpowiada za uczenie modułu, przewidywanie zagęszczeń i zbieranie statystyk.

Pozostałe dwa wywołania wykonywane są w procesie zapobiegania powstawaniu zagęszczeń. Pierwsze z nich wykonywane jest przez moduł dynamicznego routingu przez klasę *TimeTableRules*, w chwili kiedy pobierany jest z tabeli aktualny czas przejazdu przez daną ulicę. To oszacowanie czasu może zostać zmienione przez moduł predykcji. Jeżeli moduł predykcji przewiduje korek na tej ulicy, wartość ta zostanie zwiększona o daną wartość parametru zapobiegania `timeTableMultiplier`.

Drugie wywołanie pochodzi z modułu decyzyjnego *pl.edu.agh.cs.kraksim.simpledecision*. Druga implementacja modułu decyzyjnego nie współpracuje z modułem predykcji. Moduł decyzyjny SimpleDecision pobiera z jednego z modułów oceny (SOTL, RL) wartość mówiącą o tym, czy konieczne jest zapalenie zielonego światła dla danego pasa ruchu drogi. Następnie wywoływany jest moduł predykcji. Jeżeli pas ruchu należy do drogi na której pojawi się zagęszczenie, to wartość ta zostanie zwiększona o parametr zapobiegania `evaluationMultiplier`.



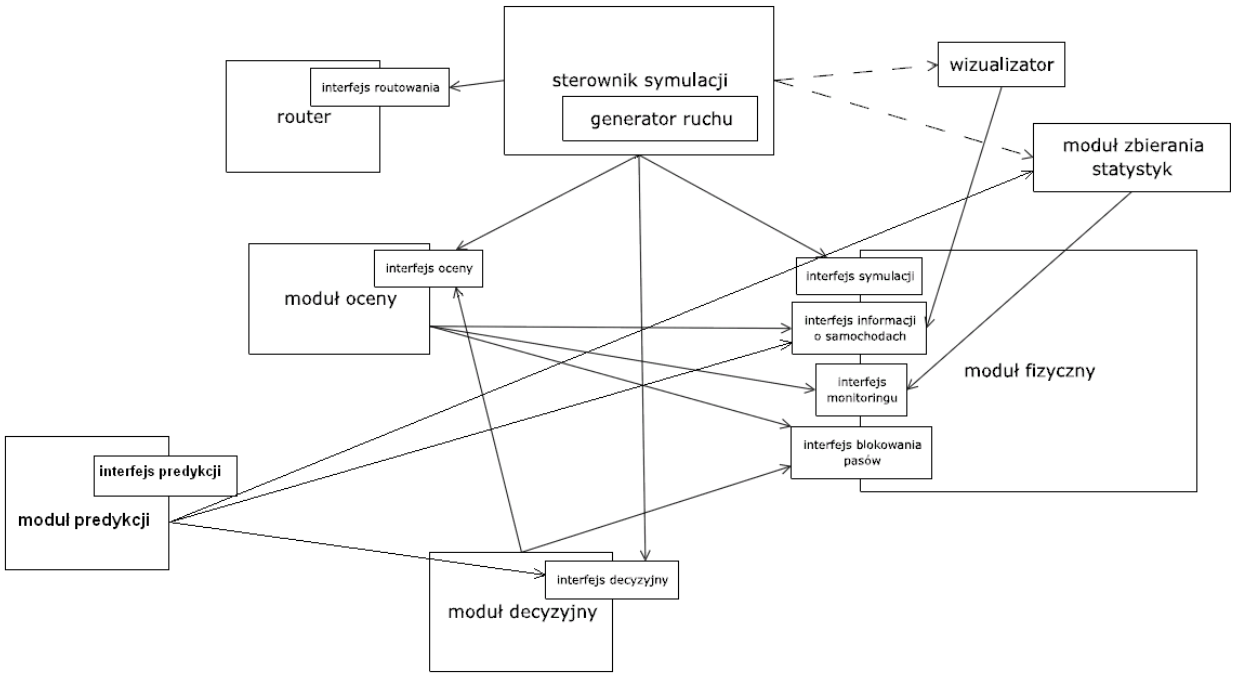
Rysunek 6.1 Odwołania pozostałych modułów systemu do modułu predykcji. Diagram architektury symulatora systemu Kraksim

6.2 MODUŁ PRZEWIDYWANIA I ZAPOBIEGANIA KORKOM ULICZNYM

6.2.1 POZYSKIWANIE DANYCH

System predykcji pobiera dane potrzebne do przewidywania zagęszczeń z innych modułów systemu [Rysunek 6.2.]. Informacje o ruchu drogowym pobierane są z modułów:

- *pl.edu.agh.cs.kraksim.ministat* – moduł zbierania statystyk,
- *pl.edu.agh.cs.kraksim.simpledecision* – moduł decyzyjny. Moduł predykcji nie współpracuje z drugim modułem decyzyjnym: *pl.edu.agh.cs.kraksim.dsyncdecision*. Nie można więc go użyć w symulacji wraz z algorytmem OptAPO,
- *pl.edu.agh.cs.kraksim.real* – moduł fizyczny.



Rysunek 6.2 Odwołania modułu predykcji do innych modułów systemu. Diagram architektury symulatora systemu Kraksim

Z konkretnych modułów pobierane są następujące informacje o ruchu drogowym:

gęstość samochodów na ulicy:

Element mapy: droga

Źródło danych: moduł *pl.edu.agh.cs.kraksim.real*

Czas mierzenia: Wartość mierzona po zakończeniu badanego okresu czasu

Wartość: ilość samochodów podzielona przez sumę długości wszystkich pasów

ilość samochodów na ulicy:

Element mapy: droga

Źródło danych: moduł *pl.edu.agh.cs.kraksim.real*

Czas mierzenia: Wartość mierzona po zakończeniu badanego okresu czasu

Wartość: ilość samochodów znajdująca się aktualnie na wszystkich pasach drogi

ilość samochodów która opuściła skrzyżowanie w ostatnim okresie

Element mapy: droga

Źródło danych: moduł *pl.edu.agh.cs.kraksim.ministat*
Czas mierzenia: Wartość mierzona przez badany okres czasu (np.: 60 sekund)
Wartość: Ilość samochodów które opuściły drogę i przejechały przez skrzyżowanie, gdy paliło się zielone światło

ilość samochodów która wjechała na drogę z początkowego skrzyżowania
Element mapy: droga
Źródło danych: moduł *pl.edu.agh.cs.kraksim.ministat*
Czas mierzenia: Wartość mierzona przez badany okres czasu (np.: 60 sekund)
Wartość: Ilość samochodów które wjechały na drogę z początkowego skrzyżowania

średni czas przejazdu w ostatnim okresie samochodów które opuściły skrzyżowanie
Element mapy: droga
Źródło danych: moduł *pl.edu.agh.cs.kraksim.ministat*
Czas mierzenia: Wartość mierzona po zakończeniu badanego okresu czasu
Wartość: średni czas przejazdu samochodów, które opuściły skrzyżowanie

aktualnie włączona faza sygnalizacji świetlnej
Element mapy: skrzyżowanie
Źródło danych: moduł *pl.edu.agh.cs.kraksim.simpledecision*
Czas mierzenia: Wartość mierzona po zakończeniu badanego okresu czasu
Wartość: numer fazy sygnalizacji świetlnej, której używa sygnalizator na zakończenie badanego okresu

czas jaki pozostał do zakończenia aktualnie używanej fazy sygnalizacji świetlnej
Element mapy: skrzyżowanie
Źródło danych: moduł *pl.edu.agh.cs.kraksim.simpledecision*
Czas mierzenia: Wartość mierzona po zakończeniu badanego okresu czasu
Wartość: liczba tur po której najwcześniej może nastąpić zmiana sygnalizacji świetlnej

długość okresu czasu jaki już używana jest aktualna faza sygnalizacji świetlnej

Element mapy: skrzyżowanie

Źródło danych: moduł *pl.edu.agh.cs.kraksim.simpledecision*

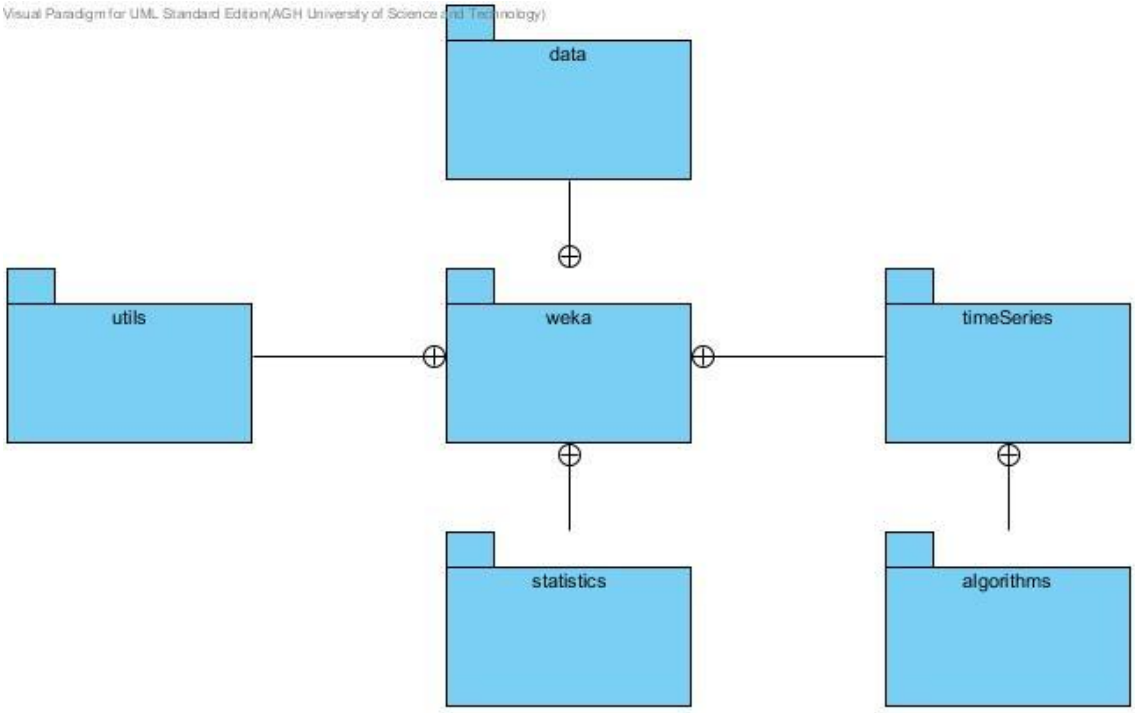
Czas mierzenia: Wartość mierzona po zakończeniu badanego okresu czasu

Wartość: liczba tur jaka minęła od ostatniej zmiany faz sygnalizacji świetlnej

6.2.2 ORGANIZACJA PAKIETÓW

Rysunek 6.3. prezentuje strukturę pakietów nowego modułu. Jego kod znajduje się w pakiecie *pl.edu.agh.cs.kraksim.weka*. W nim znajduje się główna klasa reprezentująca moduł: PredictionModule i kilka klas pomocniczych, a także cztery podpakiety:

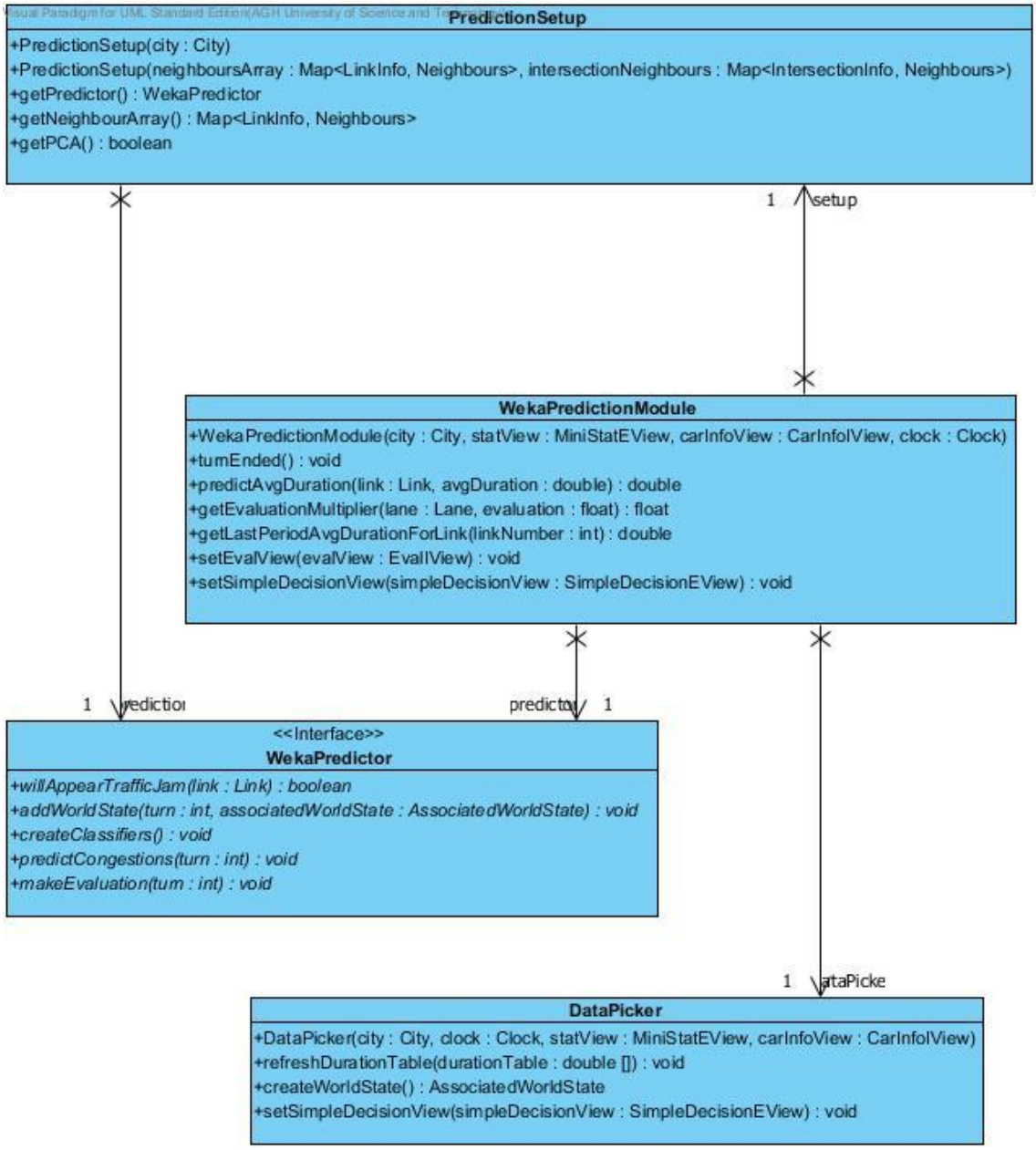
- *data* – zawiera klasy implementujące struktury danych na jakich operuje moduł przewidywania. Zawiera m.in. klasę *History* implementującą tablicę historii w module, klasę *Transaction* i *TransactionTable* implementujące struktury danych przechowujące informacje do budowy modeli regresji przewidujących dany parametr drogi lub skrzyżowania;
- *timeSeries* – agreguje klasy implementujące funkcjonalność uczenia modułu przewidywania. Posiada podpakiet *algorithms* w którym znajdują się klasy fabryk tworzące obiekty modeli regresji;
- *statistics* – umieszczone są w nim klasy służące do przewidywania wartości parametrów dróg, sprawdzenia czy przewidywana wartość oznacza wystąpienie zagęszczenia i czy przewidywanie jest poprawne. Znajdują się tutaj też klasy służące do zbierania statystyk o tym procesie;
- *utils* – tutaj znajdują się wszystkie pozostałe klasy takie jak np. *NeighbourArrayCreator* służącą do generowania list sąsiednich dróg i skrzyżowań, które mogą mieć wpływ na ruch na konkretnych drogach i parametry działania sygnalizacji świetlnej na skrzyżowaniach.



Rysunek 6.3 Diagram pakietów modułu predykcji

6.2.3 GŁÓWNE ELEMENTY MODUŁU

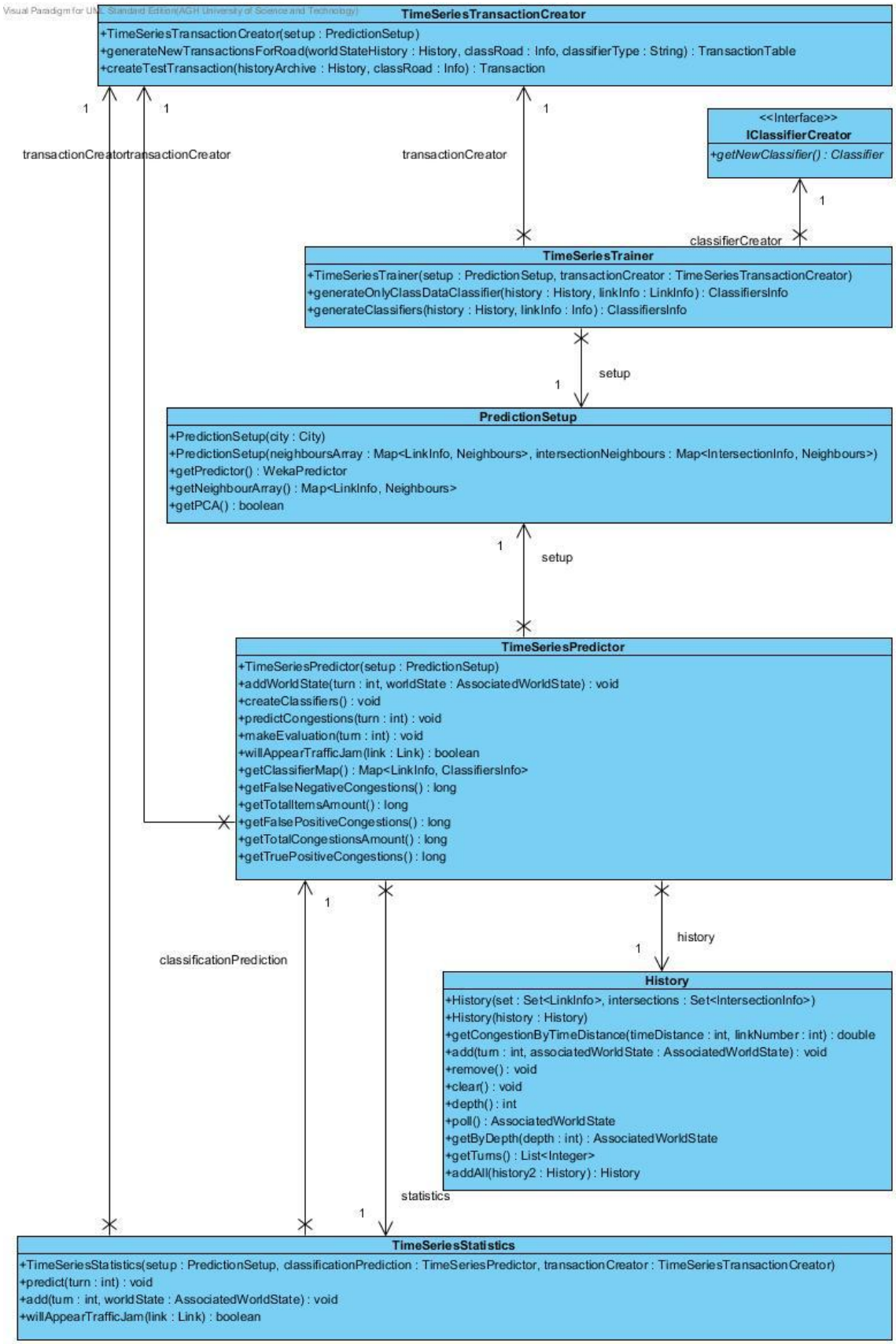
Moduł przewidywania reprezentowany jest w systemie przez klasę *WekaPredictionModule* [Rysunek 6.4.]. Jej główna metoda *turnEnded()* wywoływana w pętli symulacji, na podstawie numeru tury, decyduje czy należy zebrać dane o stanie sieci drogowej lub ponownie nauczyć moduł predykcji. W tym celu wywołuje metody interfejsu *WekaPredictor,* którego implementacja *TimeSeriesPredictor* [Rysunek 6.5] reprezentuje właściwy algorytm przewidywania. Klasa *WekaPredictionModule* korzysta z klasy *DataPicker,* służącej do pobierania informacji o stanie sieci drogowej z modułów systemu Kraksim i z klasy *PredictionSetup,* zwracającej wartości parametrów konfiguracyjnych modułu. Klasa *PredictionSetup* odczytuje konfigurację modułu z podanego przez użytkownika pliku typu properties.



Rysunek 6.4 Diagram klas, które odpowiadają za konfigurację i zarządzanie modułem przewidywania

Rysunek 6.5. prezentuje diagram klas zawierający główne klasy modułu przewidywania:

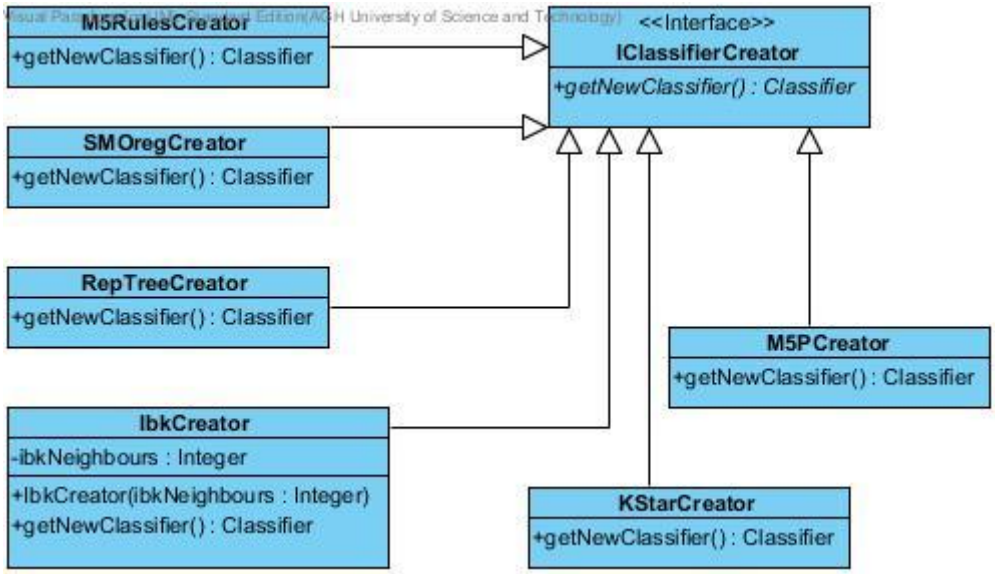
- *PredictionSetup* – wspomniana wyżej klasa zawierająca konfigurację modułu;
- *TimeSeriesPredictor* – wspomniana wcześniej klasa implementująca interfejs *WekaPredictor* reprezentująca algorytm przewidywania zagęszczeń;
- *History* – klasa w której zbierane i przechowywane są informacje o kolejnych stanach sieci drogowej;
- *TimeSeriesTrainer* – klasa odpowiedzialna za uczenie modeli regresyjnych przewidujących parametry dla jednego elementu sieci drogowej: drogi lub skrzyżowania;
- *IClassifierCreator* – interfejs zwracający instancję algorytmu budującego model regresyjny przewidujący jeden z parametrów elementu sieci drogowej;
- *TimeSeriesTransactionCreator* – na podstawie tabeli historii, dla danego elementu sieci drogowej i parametru, który go opisuje, buduje tabelę transakcji uczących dla algorytmu regresji, który będzie przewidywać wartości tego parametru. Klasa ta zwraca też transakcje testowe wykorzystywane bezpośrednio w procesie przewidywania;
- *TimeSeriesStatistics* – klasa odpowiedzialna bezpośrednio za przewidywanie zagęszczeń i zbieranie statystyk na temat poprawności tego procesu;



Rysunek 6.5 Diagram klas implementujących główną funkcjonalność modułu

Rysunek 6.6. prezentuje klasy służące do tworzenia obiektów modeli regresji uczonych do przewidywania wartości parametrów w systemie. Każda z klas zwraca inny możliwy do wykorzystania w procesie przewidywania algorytm regresji zaimplementowany w bibliotece Weka. Ogólny opis tych algorytmów znajduje się w rozdziale trzecim. Klasy te zwracają instancje następujących algorytmów:

- *M5RulesCreator* zwraca instancję klasy *M5Rules* implementującej algorytm tworzenia reguł z drzew regresji;
- *M5PCreator* zwraca instancję klasy *M5P* implementującej algorytm drzew regresji;
- *SMOregCreator* zwraca instancję klasy *SMOreg* implementującej algorytm regresji opartej o maszynę wektorów nośnych.
- *REPTreeCreator* zwraca instancję klasy *REPTree* implementującej algorytm regresji na podstawie drzew decyzyjnych;
- *IbkCreator* zwraca instancję klasy *IBk* implementującej algorytm k-najbliższych sąsiadów;
- *KStarCreator* zwraca instancję klasy *KStar* implementującej algorytm typu Instance Based Learner na podstawie pewnej funkcji podobieństwa.



Rysunek 6.6 Diagram klas dla fabryki tworzącej nowe algorytmy regresji

7 EKSPERYMENTY

W rozszerzonym o nowy moduł przewidywania i zapobiegania korków ulicznych systemie Kraksim, uruchomiono szereg symulacji. Wyniki tych testów mają dać w pierwszej kolejności odpowiedź na pytanie, przy jakich parametrach nowy moduł systemu działa najlepiej. Konieczne jest określenie czy niezależnie od rodzaju danych wejściowych i konfiguracji symulacji, optymalne wartości parametrów algorytmu przewidywania pozostają stałe.

Część eksperymentów ma pomóc określić jak różne rodzaje informacji o ruchu drogowym przydatne są do przewidywania zagęszczeń.

Jednym z celów jest również sprawdzenie jak efektywnie metody zapobiegania powstawania zagęszczeń współpracują z systemem znaków o zmiennym tekście i algorytmami sterowania sygnalizacją świetlną.

Następnie po wyznaczeniu optymalnych parametrów, na podstawie wyników, jakie dzięki nim udało się otrzymać, nastąpi próba oceny realizacji celów pracy.

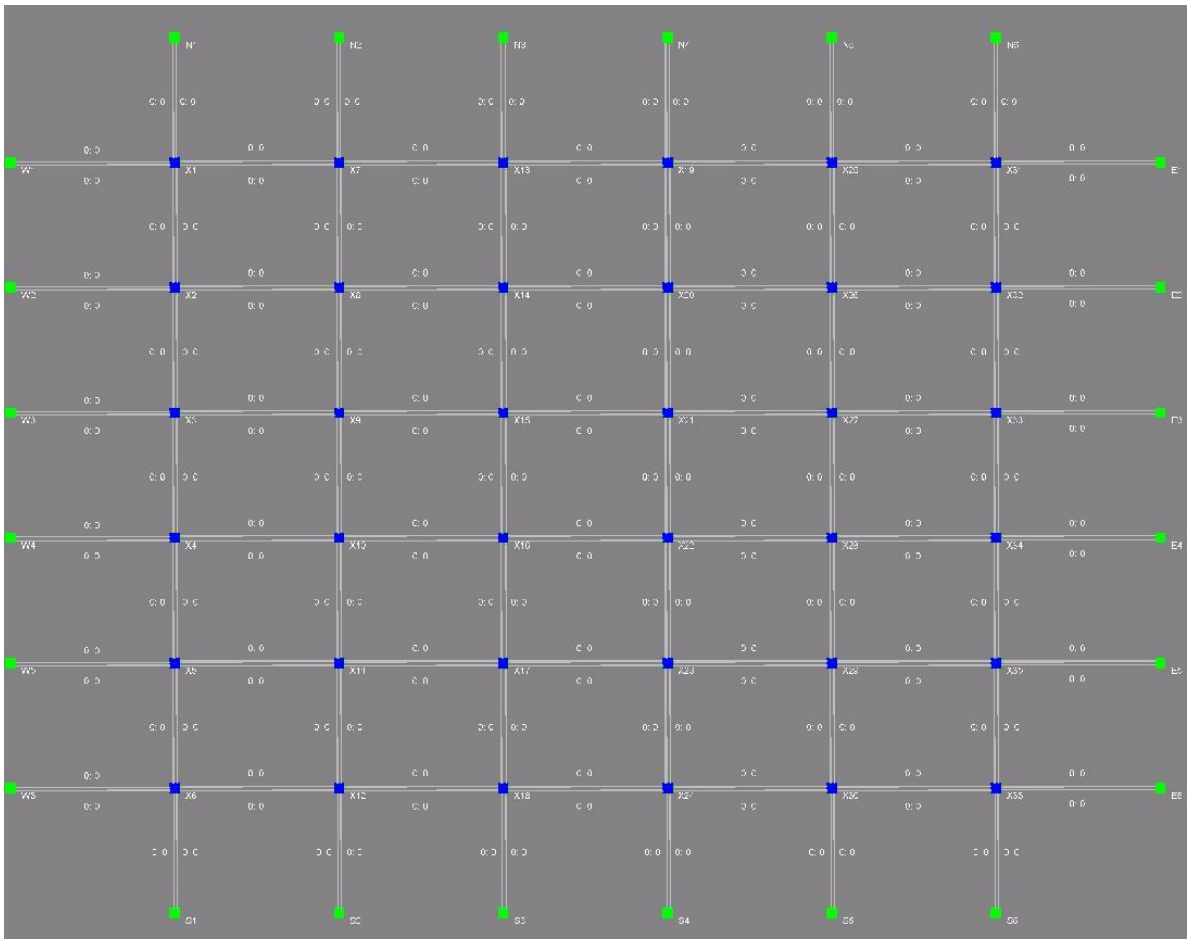
7.1 DANE

Zestaw danych potrzebnych do przeprowadzenia symulacji składa się z dwóch plików. Jednym jest plik mapy zawierający w sobie model sieci drogowej w której będzie odbywać się symulacja. Drugi z nich zawiera definicję ruchu drogowego opartą o zestaw bram z pliku pierwszego. Na potrzeby przeprowadzonych eksperymentów przygotowano odpowiednie zestawy map i schematów ruchu dla nich.

7.1.1 MAPY

7.1.1.1 Mapa Manhattan

Mapę tworzy prosta sieć drogowa ułożona w formę siatki przecinających się ulic. Jest ona na tyle duża, że samochody, które pod wpływem działania algorytmu zdecydują się zmienić trasę, rzeczywiście będą miały możliwość wyboru alternatywnej drogi.



Rysunek 7.1 Model sieci drogowej typu „Manhattan”

Charakterystyka mapy:

Liczba bram generujących ruch: 24

Liczba skrzyżowań: 36

Liczba dróg: 168

Liczba dróg z dwoma pasami ruchu: 0

Liczba dróg z jednym pasem ruchu: 168

Średnia długość drogi: 375 metrów

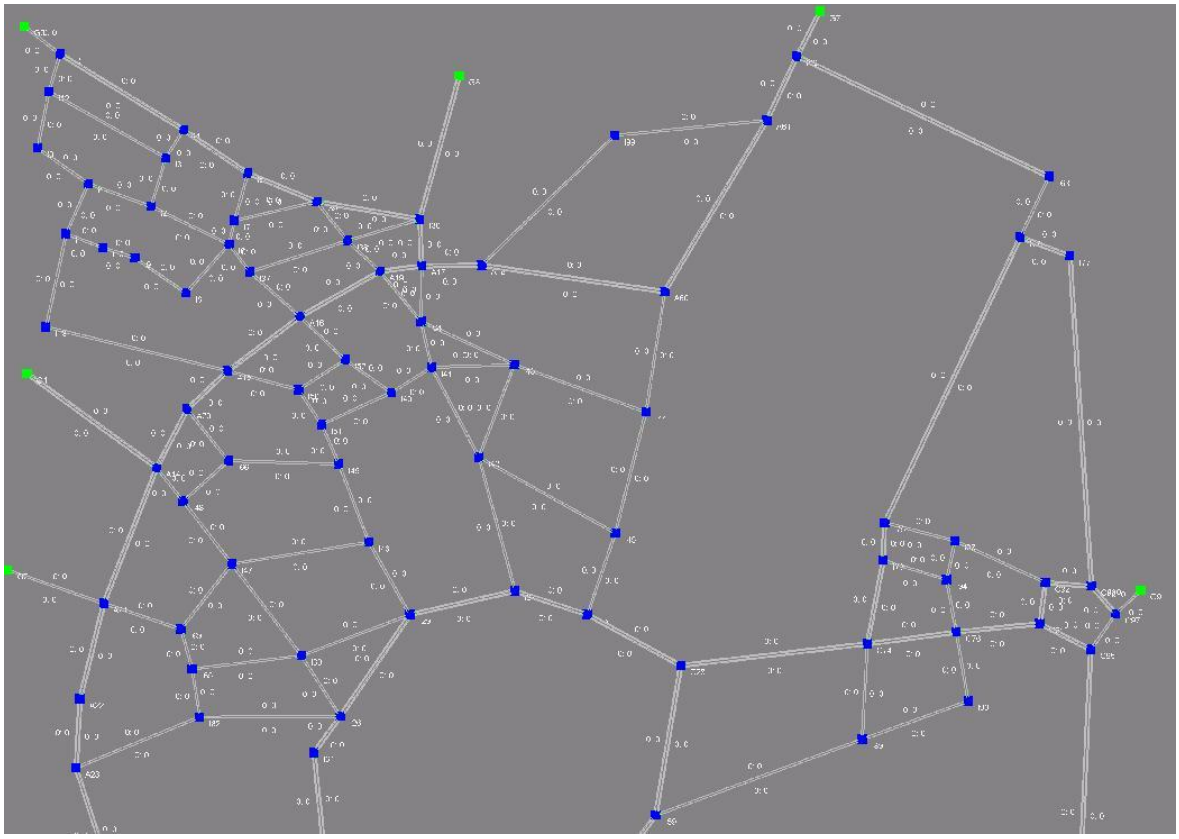
Całkowita długość sieci drogowej: 63 kilometry

7.1.1.2 Mapa Krakowa

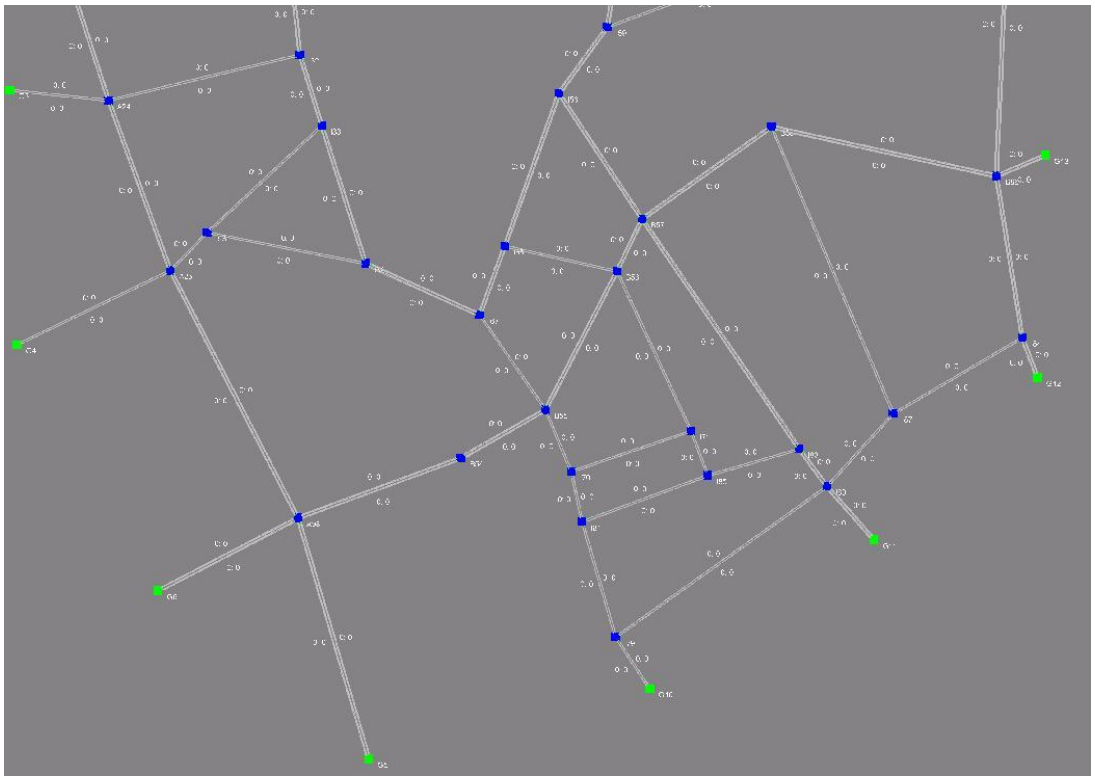
Mapa przedstawiająca fragment sieci drogowej Krakowa. Od północy ograniczona jest ulicą Wrocławską, Biskupa Jana Brandoty i Aleją 29 Listopada. Na południu jest to ulica Podgórska i mosty przerzucone przez Wisłę. Wschodnią granicę mapy stanowi Rondo Mogilskie i Aleje Powstania Warszawskiego oraz

Pułkownika Władysława Beliny-Prażnowskiego. Wzdłuż zachodniej krawędzi mapy biegną aleje Adama Mickiewicza i Juliusza Słowackiego.

Symulacje przeprowadzone na tej mapie mają dowieść możliwości prognozowania zagęszczeń i zapobiegania im w prawdziwym ruchu miejskim.



Rysunek 7.2 Model sieci drogowej Krakowa (część północna)



Rysunek 7.3 Model sieci drogowej Krakowa (część południowa)

Charakterystyka mapy:

Liczba bram generujących ruch: 14

Liczba skrzyżowań: 100

Liczba dróg: 330

Liczba dróg z dwoma pasami ruchu: 130

Liczba dróg z jednym pasem ruchu: 200

Średnia długość drogi: 274 metry

Całkowita długość sieci drogowej: 90 kilometrów

7.1.2 RUCH DROGOWY

Dla **mapy Manhattan** stworzono kilka schematów ruchu drogowego, generujących ruch o różnym natężeniu. Mapa jest symetryczna, więc generowany ruch jest rozkładany równomiernie po całej mapie. Miedzy dowolną parą bram w sposób jednostajny w ciągu 240000 tur symulacji zostaje wygenerowana następująca ilość samochodów: 700, 800, 900, 1000.

W przypadku **mapy Krakowa**, która nie jest symetryczna, w celu zapewnienia w miarę równomiernego rozłożenia ruchu w sieci drogowej i nie tworzenia niemożliwych do rozwiązania zatorów w ruchu stworzono schemat,

który generuje między bramami mapy ruch o następujących trzech wysokościach natężeń: 1100, 1800 i 3000. Ruch ten też jest generowany w ciągu 240000 tur symulacji. Większa ilość generowanych samochodów wynika z większej ilości dróg i mniejszej ilości bram, które będą ten strumień samochodów wygenerować w systemie symulacyjnym.

7.2 RODZAJE PRZEPROWADZONYCH EKSPERYMENTÓW

Opisane w poniższych podrozdziałach testy będą wykorzystywać opisany tutaj zestaw parametrów konfiguracyjnych.

Czas trwania każdej z uruchomionych symulacji to 240000 tur. Ponieważ tura symulacji odpowiada jednej sekundzie, symulacja trwa około 66 godzin czasu rzeczywistego. Okres działania modułu predykcji ustalony jest na połowę tego okresu czasu. Oznacza to, że przez pierwsze 33 godziny czasu symulacji, algorytm będzie zbierał dane o ruchu w systemie. Następnie przez drugą cześć symulacji będzie następował proces przewidywania korków ulicznych jak również ich zapobieganie, jeżeli proces ten zostanie odpowiednio skonfigurowany.

Parametrami, które pozostają stałe dla wszystkich symulacji, jest wartość wysokości historii ustawiona na 6 i wartość sąsiedztwa ustawiona na 3. Oznacza to, że przewidywanie wartości parametru w następnym kroku czasowym będzie odbywać się w oparciu o dane z 6 poprzednich. Przewidywanie będzie wykorzystywać parametry dróg i skrzyżowań odległych o 3 skrzyżowania od drogi z przewidywanym parametrem.

7.2.1 PARAMETRY SYSTEMU PREDYKCJI

Celem poniższych testów jest określenie wpływu parametrów konfiguracyjnych modułu na wyniki jego działania i określenie najbardziej przydatnych wartości tych parametrów.

7.2.1.1 Badany parametr: wielkość kroku czasowego

Badane wartości: 45 sekund, 60 sekund, 90 sekund, 120 sekund

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: M5Rules

Definicja korka: gęstość samochodów na ulicy większa niż 20%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Kraków

Algorytm sterowania sygnalizacją świetlną: RL

Analiza wyników i wnioski

Wyniki zaprezentowane w tabelach 7.1. i 7.2. można uznać za spodziewane. Przewidywana wartość parametru gęstości samochodów mierzona jest pod koniec okresu czasowego. Zagęszczenia są więc przewidywane na coraz większy okres czasu. Zwiększa się więc błąd przewidywania samego parametru z 14 do prawie 21 procent i spada ilość przewidzianych zagęszczeń opartych o ten parametr z 92 do 86 procent.

Tabela 7.1 Wyniki przewidywania przy zmianie wielkości okresu, w którym następuje pomiar stanu sieci drogowej. Zapobieganie wyłączone

Wielkość kroku czasowego	Ilość korków	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
45 sekund	9430	0,923	0,999	14.15
60 sekund	6511	0,913	0,999	16.30
90 sekund	3977	0,907	0,999	16.20
120 sekund	3102	0,863	0,998	20.97

Podobne wyniki otrzymywane są gdy włączony jest mechanizm zapobiegania zatorów w ruchu miejskim. Rośnie błąd przewidywania wartości gęstości samochodów i spada ilość przewidzianych zagęszczeń opartych o ten parametr.

Nie jest jednak zauważalna zależność między długością kroku czasowego a skutecznością przeciwdziałania

Tabela 7.2 Wyniki przewidywania przy zmianie wielkości okresu, w którym następuje pomiar stanu sieci drogowej. Zapobieganie włączone.

Wielkość kroku czasowego	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
45 sekund	4667	49.49%	0,848	0,999	15.45
60 sekund	3689	56.65%	0,837	0,998	16.13
90 sekund	1931	48.55%	0,8	0,998	16.05
120 sekund	1776	57.25%	0,73	0,998	21.15

7.2.1.2 Badany parametr: wielkość średniej kroczącej

Badane wartości: średnia z 1, 5, 10, 20 ostatnich wartości

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: M5Rules

Definicja korka: gęstość samochodów na ulicy większa niż 20%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Kraków

Algorytm sterowania sygnalizacją świetlną: SOTL

Analiza wyników i wnioski

Zastosowanie średniej kroczącej, która wygładza dane o ruchu wydaje się konieczne. Bez jej zastosowania udało się przewidzieć tylko 29 procent zagęszczeń. Wraz z użyciem średniej ilość przewidzianych zagęszczeń rośnie znacznie. Jest to spodziewany wynik, gdyż prognozowana jest wartość średnia z jednej nieznanej wartości nieznanej i n – 1 wartości znanych (gdzie n jest rozmiarem średniej).

Widoczny jest również spadek ilości przewidywanych zagęszczeń. Wartość graniczna definiująca zagęszczenie pozostaje stała, natomiast średnia krocząca

zmniejsza ilość wartości skrajnych w zbiorze danych. Znacznie mniej z nich znajduje się powyżej wartości granicznej.

Tabela 7.3 Wyniki przewidywania przy zmianie wielkości średniej wygładzającej. Zapobieganie wyłączone.

Wielkość średniej kroczącej	Ilość korków	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
1 wartość	9314	0,29	0,997	55.11
5 wartości	7497	0,841	0,998	24.63
10 wartości	6511	0,913	0,999	16.30
20 wartości	6162	0,957	0,999	8.72

7.2.1.3 Badany parametr: wykorzystany w module algorytm regresji

Badane wartości:

iBk - algorytm k-najbliższych sąsiadów, parametr k został ustalony na 5

REPTree – algorytm drzew regresji

M5Rules – algorytm generujący zbiór reguł na podstawie drzewa modeli

SmoReg – algorytm regresji oparty o maszynę wektorów nośnych

Konfiguracja

Parametry modułu predykcji:

Definicja korka: gęstość samochodów na ulicy większa niż 20%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Kraków

Algorytm sterowania sygnalizacją świetlną: SOTL

Analiza wyników i wnioski

Widać wyraźnie, że algorytm k-najbliższych sąsiadów nie radzi sobie z przewidywaniem. Prawdopodobnie wynika to z faktu, że dla wielu punktów wartość regresji została określona na podstawie danych z odległych dróg. Dróg odległych jest znacznie więcej niż tych bliskich. Metryka odległości równo traktuje

wszystkie z nich, zostaje więc zdominowana przez wartości z dróg odległych, które niekoniecznie wpływają zauważalnie na ruch na badanej ulicy.

Zwiększanie liczby sąsiadów nie poprawi tutaj wyniku. Musiałaby ona znacznie przekraczać ilość atrybutów. Tylko ograniczenie wielkości parametru sąsiedztwa do wartości 1 może poprawić wynik przewidywania. Jednak ilość atrybutów nadal może pozostać zbyt duża (przynajmniej 8). Lepiej skorzystać z jednego z pozostałych algorytmów.

Ogólnie opisana wyżej wada nie dotyczy algorytmu k-najbliższych sąsiadów, ale wykorzystania metod opartych na podejściu Instance Based. Metody, które budują model regresji wydają się być lepszym rozwiązaniem, jeżeli tylko kilka atrybutów ma decydujący wpływ na zmienną objaśnianą.

Tabela 7.4 Skuteczność przewidywania zagęszczeń z wykorzystaniem różnych algorytmów regresji. Zapobieganie wyłączone.

Algorytm	Ilość korków	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
iBk	5035	0,553	0,998	53.51
REPTree	4881	0,908	0,999	13.90
m5Rules	4596	0,893	0,999	22.33
smoreg	4200	0,886	0,999	12.26

Tabela 7.5 Skuteczność przewidywania zagęszczeń z wykorzystaniem różnych algorytmów regresji. Zapobieganie włączone.

Algorytm	Ilość korków	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
iBk	1555	0,327	0,997	37.93
REPTree	1809	0,629	0,999	16.75
m5Rules	1372	0,692	0,998	18.27
smoreg	1248	0,721	0,999	12.05

7.2.2 PARAMETRY SYSTEMU ZAPOBIEGANIA ZAGĘSZCZENIOM W RUCHU DROGOWYM

Celem poniższych eksperymentów jest zbadanie skuteczności przeciwdziałania powstawaniu zagęszczeń w ruchu drogowym. Badane będą dwie metody zapobiegania. Pierwsza oparta jest o przekazywanie kierowcom informacji o czasie przejazdu poprzez podsystem znaków o zmiennym tekście. Druga

wykorzystuje mechanizm wpływu na wybór fazy sygnalizacji świetnej. W testach badane są różne wartości parametrów tych metod.

7.2.2.1 Badany parametr: wielkość parametru wpływu na znaki o zmiennym tekście

Badane wartości: 1,2,3,5,10,15

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: M5Rules

Definicja korka: gęstość samochodów na ulicy większa niż 12%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Kraków

Algorytm sterowania sygnalizacją świetlną: SOTL

Analiza wyników i wnioski

Wraz ze wzrostem wartości parametru widać spadek ilości zatorów drogowych powstających w symulacjach z działającym modułem przeciwdziałania powstawaniu zagęszczeń. Dla małych wartości parametru, równych 2 i 3, widoczne jest znaczne zmniejszenie się ilości zagęszczeń. Spadek ten można obserwować, gdy sygnalizację świetlną kontrolują zarówno algorytm SOTL (Tabela 7.6) jak i RL (Tabela 7.7). Dla wyższych wartości parametru ruch nie poprawia się bardziej. Kierowcy, którzy otrzymali informację o tym, że długość przejazdu daną drogą zwiększył się daną ilość razy, unikają jej, wybierając objazd. Drogę z przewidywanym korkiem i długim czasem przejazdu wybierają tylko ci kierowcy, którzy muszą tamtędy przejechać, ale na nich informacja o zwiększonym czasie przejazdu nie będzie miała wpływu.

Tabela 7.6 Zależność skuteczności zapobiegania zagęszczeniom od wartości parametru wpływającego na działanie znaków o zmiennym tekście. Sygnalizacja świetlna – algorytm SOTL

Wielkość parametru	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
1 – brak wpływu na ruch	14865		0,896	0,998	12.98
2	10048	67%	0,848	0,998	15.93
3	7817	53%	0,813	0,998	15.46
4	6533	44%	0,787	0,997	15.42
5	6405	43%	0,773	0,997	15.20
10	5867	39%	0,741	0,996	17.46
15	5918	40%	0,739	0,996	15.94

Tabela 7.7 Zależność skuteczności zapobiegania zagęszczeniom od wartości parametru wpływającego na działanie znaków o zmiennym tekście. Sygnalizacja świetlna – algorytm RL

Wielkość parametru	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
1 – brak wpływu na ruch	23557		0,89	0,997	15.38
2	17809	76%	0,862	0,996	16.23
3	15992	68%	0,841	0,996	18.00
5	14927	63%	0,818	0,995	19.18
10	14145	60%	0,812	0,994	21.11
15	14088	60%	0,802	0,994	20.35

7.2.2.2 Badany parametr: wielkość parametru wpływu na wybór fazy sygnalizacji świetlnej

Badane wartości: 1,2,3,5,10,20,30,50

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: M5Rules

Definicja korka: gęstość samochodów na ulicy większa niż 12% i 20%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Kraków

Algorytm sterowania sygnalizacją świetlną: RL

Analiza wyników i wnioski

Metoda polegająca na zapobieganiu zagęszczeniom poprzez wpływ na wybór fazy świetlnej, osiąga znacznie gorsze rezultaty niż sterowanie ruchem poprzez znaki o zmiennym tekście. Czasami, tak jak w tabeli 7.9, jest widoczny niewielki efekt jej działania, ale równie często nie ma go wcale (tabela 7.8). Należy zauważyć, że wyniki z obu tabel zostały wykonane dla symulacji, które różniły się jedynie wartością definiującą korek.

Tabela 7.8 Zależność skuteczności zapobiegania zagęszczeniom od wartości parametru wpływającego na wybór fazy sygnalizacji świetlnej. Definicja zagęszczenia 12%

Wielkość parametru	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
1 – brak wpływu na ruch	24215		0,892	0,997	15.14
2.5	24299	100%	0,885	0,997	15.13
5	24285	100%	0,883	0,996	15.16
10	23805	98%	0,879	0,996	15.35
15	24987	98%	0,877	0,996	15.40
30	25839	107%	0,869	0,996	15.61
50	25092	104%	0,869	0,996	15.35

Tabela 7.9 Zależność skuteczności zapobiegania zagęszczeniom od wartości parametru wpływającego na wybór fazy sygnalizacji świetlnej. Definicja zagęszczenia 20%

Wielkość parametru	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
1 – brak wpływu na ruch	6981		0,913	0,999	15.03
2.5	6600	94%	0,893	0,999	15.11
5	6163	88%	0,876	0,998	15.05
10	6089	87%	0,874	0,998	15.33
15	5885	84%	0,869	0,998	14.98
30	6094	87%	0,869	0,998	15.36
50	5830	83%	0,863	0,998	15.19

7.2.3 TYPY DANYCH UCZĄCYCH

Ten test stawia sobie za cel zbadanie, czy użycie dodatkowych typów danych w procesie przewidywania może podnieść jego jakość. Bazowa symulacja wykorzystuje tylko informacje o gęstości samochodów na ulicy. Porównane są z nią wyniki innych symulacji, w których do przewidywania użyto także innych informacji jakie można otrzymać z systemu symulacyjnego, takich jak średni czas przejazdu czy ilość samochodów opuszczających ulicę.

Badany parametr: zbiór danych uczących

Badane wartości:

a) gęstość samochodów na ulicy,
b) gęstość samochodów na ulicy, ilość samochodów wjeżdżających na ulicę, ilość samochodów opuszczających ulicę,
c) gęstość samochodów na ulicy, średni czas przejazdu,
d) gęstość samochodów na ulicy, informacje z modułu oceny (SOTL): wartość oceny pasów ruchu na drodze i przewidywany czas palenia się zielonego światła dla tych pasów ruchu,
e) gęstość samochodów na ulicy, faza sygnalizacji świetlnej, długość trwania fazy sygnalizacji świetnej do końca tego okresu, planowany czas trwania aktualnej fazy sygnalizacji świetnej.

Konfiguracja

Parametry modułu predykcji:
Algorytm regresji: M5Rules
Definicja korka: gęstość samochodów na ulicy większa niż 20%
Dane o ruchu drogowym: gęstość samochodów na ulicy
Średnia krocząca: 10 wartości

Parametry symulacji:
Mapa: Kraków
Algorytm sterowania sygnalizacją świetlną: SOTL

Analiza wyników i wnioski

Użycie dodatkowych typów informacji nie wpływa bardzo na polepszenie jakości przewidywania zagęszczeń. Widać lekką poprawę w przypadku użycia informacji o samochodach wjeżdżających i opuszczających ulicę, ale nie jest to bardzo znaczny wzrost i może mieścić się w błędzie statystycznym.

Możliwe jest też, że ilość instancji uczących była za mała, aby nauczyć model tak dużą ilością atrybutów lub też atrybuty te są bardzo skorelowane i zysk informacyjny z ich użycia jest niewielki.

Tabela 7.10 Wpływ użycia różnych rodzajów danych na proces przewidywania

Rodzaj danych	Ilość korków	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
a)	3689	0,837	0,998	16.13
b)	3983	0,907	0,999	15.34
c)	4933	0,883	0,999	16.44
d)	4394	0,795	0,999	20,94
e)	4216	0,868	0,999	15.73

7.2.4 RUCH DROGOWY O RÓŻNEJ WIELKOŚCI

Celem uruchomionych testów jest zbadanie wpływu na wyniki prognozowania zagęszczeń różnej wielkości ruchu drogowego.

Badany parametr: poziom ruchu drogowego

Badane wartości: Schematy ruchu drogowego generujące między każdą parą bram odpowiednio 700, 800, 900, 1000 samochodów.

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: REPTree

Definicja korka: ilość samochodów większa niż 7

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Manhattan

Algorytm sterowania sygnalizacją świetlną: SOTL

Analiza wyników i wnioski

Wraz ze wzrostem poziomu ruchu zwiększa się też ilość korków ulicznych w symulacji. Sprawia to też, że rośnie ich wykrywalność, mimo że błąd prognozowania samego parametru gęstości samochodów jest właściwie stały. Związane jest to z właściwością algorytmu REPTree, który przewidując wartość regresji, oblicza wartość średnią zmiennej objaśnianej dla instancji uczących. Model regresji przewiduje więc wartości średnie, a nie skrajne. Poziom definiujący zagęszczenie pozostaje stały, ale dla symulacji z coraz mniejszą gęstością ruchu ta wartość jest coraz bardziej odległa od średniej gęstości ruchu. Algorytm REPTree przewiduje więc częściej wartości bliższe średniej, niż wartości powyżej parametru określającego zagęszczenie. Myli się więc coraz częściej, osiągając tylko 48,5 procent przewidzianych zatorów dla ruchu o wielkości 700.

Tabela 7.11 Wpływ wielkości ruchu na wyniki przewidywania

Wielkość ruchu	Ilość korków	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (ilość samochodów)
700	1906	0,485	0,998	13.69
800	8023	0,712	0,995	12.55
900	20500	0,83	0,989	12.82
1000	42198	0,867	0,981	12.58

7.2.5 RÓŻNE DEFINICJE ZAGĘSZCZENIA RUCHU

Celem tego eksperymentu jest określenie wpływu wartości definicji zagęszczenia ruchu na poziom ruchu w systemie. Jest to test bardzo podobny do przeprowadzonego w punkcie 7.3.4. Zmienną, która ma wyraźny wpływ na uzyskane wyniki jest zmiana używanego algorytmu regresji. W testach w punkcie 7.3.4. wykorzystywany był algorytm REPTree. Tutaj uruchomiono symulacje z użyciem algorytmu M5Rules.

Badany parametr: definicja korka

Badane wartości: gęstość samochodów na ulicy większa niż

a) 10%
b) 20%
c) 30%

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: M5Rules

Definicja korka: gęstość samochodów na ulicy większa niż 20%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Kraków

Algorytm sterowania sygnalizacją świetlną: SOTL

Analiza wyników i wnioski:

W punkcie 7.3.4 przewidywanie oparte o algorytm regresji REPTree nie radziło sobie z przewidywaniem zagęszczeń, gdy wartość je określająca była bardzo wysoka w stosunku do poziomu ruchu. Oznacza to małą ilość zagęszczeń możliwą do prognozowania. Algorytm M5Rules lepiej reprezentuje zbiór danych uczących niż algorytm REPTree. Przy gęstości zapełnienia drogi na poziomie 30% co oznacza obecność 1321 zagęszczeń udało przewidzieć się 70,6% z nich.

Tabela 7.12 Wpływ wartości definiującej zagęszczenie na wyniki prognozowania

Gęstość ruchu definiująca zagęszczenie	Współczynnik zapobiegania	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
10%	1– brak wpływu na ruch	23498		0,793	0,996	27.27
20%	1– brak wpływu na ruch	4596		0,893	0,999	22.33
30%	1– brak wpływu na ruch	1321		0,706	1	16.87
10%	10	8868	38%	0,535	0,984	29.69
20%	10	1372	30%	0,692	0,998	18.27
30%	10	379	29%	0,435	1	18.51

7.2.6 *WSPÓŁPRACA ALGORYTMÓW STEROWANIA SYGNALIZACJĄ ŚWIETNĄ I METODY ZAPOBIEGANIA OPARTEJ O ZNAKI O ZMIENNYM TEKŚCIE*

Celem poniższych testów jest pokazanie różnicy we współpracy stosowanych algorytmów sterowania sygnalizacją świetlną i metody zapobiegania opartej o znaki o zmiennym tekście.

Badany parametr: algorytm sterowania sygnalizacją świetlną

Badane wartości: SOTL, RL

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: REPTree

Definicja korka: gęstość samochodów na ulicy większa niż 20%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Manhattan

Analiza wyników i wnioski

Głównym wnioskiem z tej grupy testów jest stwierdzenie, że algorytm RL znacznie gorzej nadaje się do użycia w sytuacjach, kiedy chcemy ograniczać liczbę zagęszczeń w sieci drogowej. Po pierwsze znacznie mniej sprawnie współpracuje z algorytmem zapobiegającym ich powstawaniu. Ilość zagęszczeń spadła tylko do 60% podczas gdy dla metody SOTL spadł do 39 procent. Dodatkowo należy zauważyć, że liczba korków z uruchomionym algorytmem zapobiegania i algorytmem RL jest praktycznie taka sama jak dla algorytmu SOTL bez włączonego algorytmu zapobiegania.

Tabela 7.13 Porównanie wyników przewidywania przy współpracy z algorytmami SOTL i RL

Algorytm	Współczynnik zapobiegania	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samoch odów)
SOTL	1– brak wpływu na ruch	14865		0,896	0,998	12.98
RL	1– brak wpływu na ruch	23557		0,89	0,997	15.38
SOTL	10	5867	39%	0,741	0,996	17.46
RL	10	14145	60%	0,812	0,994	21.11

7.2.7 PRZEWIDYWANIE NA KILKA KROKÓW W PRZÓD

Celem tego eksperymentu jest sprawdzenie czy możliwe jest przewidywanie i zapobieganie zagęszczeniom na większych okresach czasowych niż jeden okres zbierania danych.

Badany parametr: ilość kroków czasowych

Badane wartości:

a) 0 – przewidywanie na następny okres
b) 1 – przewidywanie drugie okresu w przyszłości
c) 2 – przewidywanie trzeciego okresu w przyszłości

Konfiguracja

Parametry modułu predykcji:

Algorytm regresji: M5Rules

Definicja korka: gęstość samochodów na ulicy większa niż 10%

Dane o ruchu drogowym: gęstość samochodów na ulicy

Średnia krocząca: 10 wartości

Parametry symulacji:

Mapa: Kraków

Algorytm sterowania sygnalizacją świetlną: SOTL

Analiza wyników i wnioski:

Widoczny jest spadek dokładności przewidywania zagęszczeń, szczególnie gdy włączony jest mechanizm zapobiegania powstawaniu zagęszczeń. Przy prognozowaniu 3 okresu czasowego (gęstość samochodów na ulicy za 3 minuty) skuteczność przewidywania korków spada do 33 procent.

Tabela 7.14 Wyniki przewidywania na kilka okresów w przód

Ilość kroków przewidywania	Współczynnik zapobiegania	Ilość korków	Procent korków względem symulacji z wyłączonym zapobieganiem	Procent przewidzianych korków	Procent przewidzianych okresów bez zagęszczeń	MAPE (gęstość samochodów)
0 – przewidywanie na następny okres	1– brak wpływu na ruch	23498		0,793	0,996	27.27
1 – przewidywanie drugiego okresu w przyszłości	1– brak wpływu na ruch	24937		0,772	0,993	29.11
2 – przewidywanie trzeciego okresu w przyszłości	1– brak wpływu na ruch	26105		0,753	0,995	27.15
0 – przewidywanie na następny okres	10	8868	38%	0,535	0,984	29.69
1 – przewidywanie drugiego okresu w przyszłości	10	10220	41%	0,427	0,984	41.58
2 – przewidywanie trzeciego okresu w przyszłości	10	10205	39%	0,331	0,983	47.50

7.3 PODSUMOWANIE WYNIKÓW

W rozdziale tym zostały przedstawione przykłady wykorzystania zaimplementowanego w ramach pracy dyplomowej modułu do przewidywania i zapobiegania zagęszczeniom, dla dwóch przykładowych sieci drogowych, z których jedna reprezentuje uproszczony układ ulic miasta Kraków. Wnioski z testów są następujące:

W celu uzyskania dobrych wyników przewidywania zagęszczeń należy skorzystać z metody wygładzania kolejnych wartości parametrów ruchu drogowego.

Z wyników testów metod zapobiegania wynika, że pierwsza metoda oparta o znaki ze zmiennym tekstem działa bardzo dobrze, natomiast druga oparta o wywieranie wpływu na system sygnalizacji świetlnej gorzej. Najprawdopodobniej jest ona za prosta i należy ją zmodyfikować, albo bezpośrednio oprzeć sterowanie światłami na skrzyżowaniach o zaimplementowany algorytm predykcji.

Widać wyraźnie, że algorytm sterowania sygnalizacją świetną SOTL lepiej radzi sobie z optymalizacją ruchu, pod względem korków, niż algorytm RL. RL wsparty dopiero przez metodę zapobiegania wykorzystującą znaki o zmiennym tekście jest wstanie ograniczyć ilość korków ulicznych do poziomu metody SOTL, ale bez wsparcia metody zapobiegania. Jest tak pomimo tego, że algorytm RL jest znacznie bardziej skomplikowany od metody SOTL.

8 PODSUMOWANIE

Zmniejszająca się płynność ruchu drogowego jest wyzwaniem dla administracji publicznej lokalnej i państwowej. Spadek płynności generuje dodatkowe koszty tak dla firm jak i dla budżetów publicznych. Dlatego też obecnie miasta mają obowiązek wprowadzania inteligentnych systemów zarządzania ruchem. Wynika to między innymi z obowiązku implementacji [22] do prawa polskiego dyrektywy Parlamentu Europejskiego i Rady 2010/40/UE z 7 lipca 2010 roku w sprawie ram wdrażania inteligentnych systemów transportowych w obszarze transportu drogowego. Powyższa Dyrektywa jest wprowadzana do polskiego prawa Ustawą z dnia 27 lipca 2012.

W Polsce inteligentne systemy transportowe są ciągle nowością, ale w dużej ilości miast europejskich z powodzeniem one funkcjonują. Nie tylko zarządzają komunikacją publiczną, sterują światłami, informują kierowców o korkach ulicznych i wypadkach, ale także pozwalają racjonalnie planować remonty dróg i ulic. Rozwiązywanie problemów ruchu drogowego jest więc ważnym problemem, któremu w przyszłości poświęconych będzie wiele badań.

W ramach prac nad rozwojem systemu Kraksim, udało się stworzyć elastyczny moduł przewidywania zagęszczeń. Koncepcja modułu przedstawiona jest w rozdziale piątym pracy magisterskiej, a w rozdziale szóstym przedstawiono jego implementację. Pozwala on na użycie różnych definicji zagęszczeń i przewidywanie ich za pomocą wielu różnych typów danych charakteryzujących ruch drogowy. Może on wykorzystywać wiele rodzajów algorytmów regresji statystycznej. Przy przewidywaniu zagęszczeń moduł osiąga bardzo dobre wyniki. Przewidywane jest nawet 90% z nich. Dodatkowo możliwa jest też optymalizacja ruchu poprzez eliminację zagęszczeń powstających w trakcie trwania symulacji. Możliwe jest ograniczenie zagęszczeń nawet do 30 procent ich ilości, gdy moduł zapobiegania nie jest włączony. **Należy więc uznać, że cel pracy magisterskiej został zrealizowany.**

Nie można jednak uznać rozwijanej aplikacji za kompletną, ani tym bardziej problemu optymalizacji ruchu drogowego za rozwiązany. Ciągle istnieje problem opisu zagęszczeń w gęstej sieci drogowej z dużą liczbą skrzyżowań

kontrolowanych sygnalizacją świetlną. Zaburza ona w znacznym stopniu wartości pomiarów parametrów opisujących ruch. Można spróbować zmodyfikować system tak, aby przewidywał zagęszczenia jednocześnie na podstawie parametrów wygładzonych średnią i nie wygładzonych. Można też spróbować znaleźć inny sposób opisu ruchu drogowego, który będzie go dobrze reprezentował, pomimo włączonej sygnalizacji świetlnej.

Można również spróbować implementacji bardziej skomplikowanych metod zapobiegania zagęszczeń. Na początek można zacząć od integracji modułu przewidywania z modułem implementującym sterowanie sygnalizacją świetlną za pomocą algorytmu OptAPO. Algorytm ten swoją funkcję oceny wylicza na podstawie aktualnej ilości samochodów dojeżdżających do skrzyżowania. W tym miejscu można podłączyć moduł przewidywania, który będzie w krótkim czasie przewidywał tę wartość.

BIBLIOGRAFIA

[1] B. Rybacki, "Modelowanie i optymalizacja ruchu miejskiego przy uzyciu wybranych technik," EAIiE AGH, Kraków, Praca magisterska 2008.

[2] Ł. Dziewoński and M. Zalewski, "Kraksim – wielopasowość i predykcja," Katedra Informatyki EAIiE AGH, Kraków, Dokumentacja projektowa 2008.

[3] Praca zbiorowa. Wykład na temat telematyki transportu Zakładu Telekomunikacji w Transporcie Politechniki Warszawskiej. [Online]. http://www.it.pw.edu.pl/twt/loader.php?page=telematyka

[4] P.T., Feng, Y., Wang, X. Martin, "Detector Technology Evaluation," Department of Civil and Environmental Engineering University of Utah Traffic Lab, Raport techniczny 2003.

[5] G. Leduc, "Road Traffic Data: Collection Methods and Applications," Institute for Prospective Technological Studies, Raport techniczny JRC 47967, 2008.

[6] S. Bajwa, E. Chung, and M. Kuwahara, "An adaptive travel time prediction model based on pattern matching," , Tokyo, 2004.

[7] E. Chung, "Classification of traffic pattern," in *Proceedings of 10th ITS World Congress*, Madrid, 2003.

[8] O. Masutani et al., "Pheromone Model: Application to Traffic Congestion Prediction," in *Proceeding AAMAS '06 Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, New York, 2006, pp. 73 - 80.

[9] C. Mark, A. Sadek, and D. Rizzo, "Predicting Experienced Travel Time with Neural Networks: A PARAMICS Simulation Study," in *2004 IEEE Intelligent Transpotation Systems Conference*, Washington, 2004, pp. 906-911.

[10] C. Cui, J. Shin, M. Miyazaki, G. Gakuin, and H. Lee, "Traffic Signal Control using Predicted Distribution of Traffic Jam," in *The International Conference on Electrical Engineering*, 2009.

[11] StatSoft. (2006) Elektroniczny Podręcznik Statystyki PL. [Online]. http://www.statsoft.pl/textbook/stathome.html

[12] The University of Waikato Praca zbiorowa. (2012) Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [Online]. http://www.cs.waikato.ac.nz/ml/weka

[13] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining. Practical Machine Learning Tools and Techniques*, Third Edition ed. United States: Elsevier, 2011.

[14] N. T. Ratrout and S. M. Rahman, "A comparative analysis of currently used

microscopic and macroscopic traffic simulation software," *The Arabian Journal for Science and Engineering*, vol. 34, no. 1B, pp. 121- 133, kwiecień 2009.

[15] S. A. Boxill and L. Yu, "An Evaluation of Traffic Simulation Models for Supporting ITS Development," Center for Transportation Training and Research Texas Southern University, Houston, Raport techniczny SWUTC/00/167602-1, 2000.

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

[17] K. Nagel, P. Stretz, M. Pieck, R. Donnelly, and C. L. Barrett, "TRANSIMS traffic flow characteristics," *ArXiv Condensed Matter e-prints*, p. 10003, październik 1997.

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

[19] C. Gershenson, "Self-Organizing Traffic Lights," Centrum Leo Apostel, Vrije Universiteit, Brussel, Raport techniczny 2008.

[20] M. Wiering, J. van Veenen, J. Vreeken, and A Koopman, "Intelligent Traffic Light Control," Institute of Information and Computing Sciences, Utrecht University, Raport techniczny UU-CS-2004-029, 2004.

[21] D. de Oliveira, Ana Bazzan, and V. Lesser, "Using Cooperative Mediation to Coordinate Traffic Lights: a Case Study," *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 463-469, lipiec 2005.

[22] M. Cyrankiewicz. (2012, lipiec) Ruch drogowy: Inteligentne systemy transportowe obowiązkowe w miastach | rp.pl. [Online]. http://prawo.rp.pl/artykul/757925,920335-Ruch-drogowy--Inteligentne-systemy-transportowe-obowiazkowe-w-miastach.html

ZAŁĄCZNIK A – ZAWARTOŚĆ DYSKU DVD

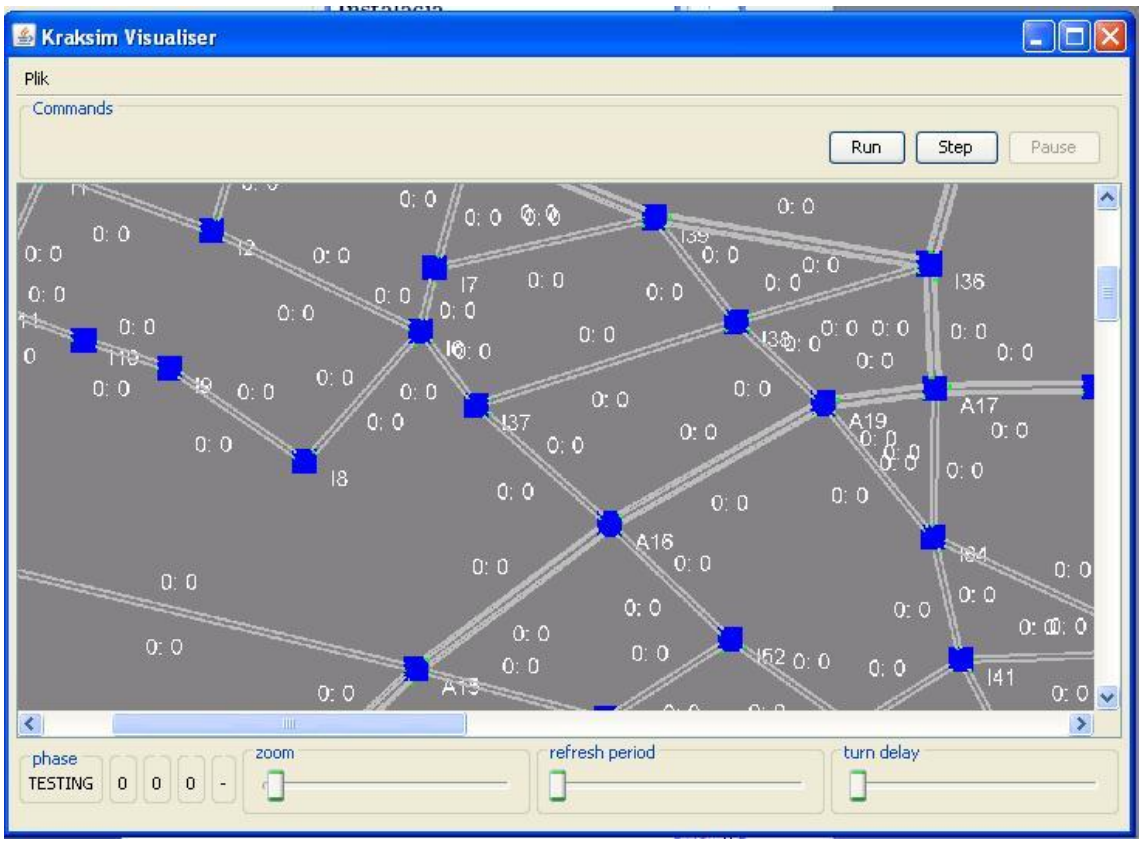
Dołączony do pracy dyplomowej dysk DVD zawiera:

- plik *README.txt*
  - spis zawartości dysku
- katalog *aplikacja*
  - wersja uruchomieniowa aplikacja zawierająca przykładową konfiguracje symulacji
- katalog *kody źródłowe*
  - kody źródłowe systemu Kraksim zawierające modyfikacje wprowadzone w ramach pracy dyplomowej
- katalog *wyniki*
  - wyniki przeprowadzonych eksperymentów
- katalog *dokumentacja*
  - *dokumenty stworzone w ramach prac nad systemem Kraksim, które stanowią dokumentację projektu*

ZAŁĄCZNIK B – INSTRUKCJA OBSŁUGI

W katalogu *aplikacja* na dostarczonym wraz z pracą dysku DVD znajduje się gotowa do działania wersja systemu Kraksim, wraz z przykładowymi plikami konfiguracyjnymi. Kraksim jest aplikacją napisaną w języku programowania Java i wymaga, aby na komputerze była zainstalowana wirtualna maszyna Javy w wersji 1.6. Aby uruchomić graficzny interfejs aplikacji należy dwukrotnie kliknąć na ikonę pliku kraksim.jar. Uruchomioną aplikację prezentuje rysunek poniżej.

Uruchomienie symulacji polega na wybraniu Opcji *Załaduj* z listy rozwijanej *Plik*. W tym momencie zostanie przygotowana symulacja, z wszystkimi opcjami wyspecyfikowanymi w plikach konfiguracyjnych. Wygląd załadowanej symulacji prezentowany jest na rysunku poniżej. Aby rozpocząć proces przeprowadzania symulacji należy wybrać myszką przycisk *Run*.



Informacje na temat plików stanowiących konfigurację, dane wejściowe i wyniki działania symulacji znajdują się w plikach pdf w katalogu *dokumentacja*.

Plik konfiguracyjny modułu przewidywania

Plik konfiguracyjny modułu przewidywania powinien mieć ustawione wszystkie opcje opisane w podrozdziale 5.5. pracy magisterskiej. Przykładowy plik konfiguracyjny może mieć postać:

```
outputMainFolder = outputFolder
writeDataSetToFile = false

worldStateUpdatePeriod = 60
timeSeriesUpdatePeriod = 120000
statisticsDumpTime = 240000
averageSize=10
```

```
maxNumberOfInfluencedLinks = 3
maxNumberOfInfluencedTimesteps = 6
minNumberOfInfluencedTimesteps = 1
predictionSize = 0
timeTableMultiplier = 10
evaluationMultiplier = 1

regressionDataType = carsDensity
congestionValue = 0.3

carsDensity = true
carsOut = false
carsIn = false
carsOn = false
durationLevel = true
evaluation = false
greenDuration = false
carsDensityMovingAvg = false
durationLevelMovingAvg = false
phase = false
phaseWillLast = false
phaseLast = false

regressionAlgorithm = ibk
ibkNeighbours = 1
```

Plik wyjściowy modułu predykcji

Plik z wynikami przewidywania zapisywany jest w katalogu podanym w parametrze *outputMainFolder* pliku konfiguracyjnego. Nosi on nazwę *result.txt.* Jego kolejne linie zawierają następujące informacje:

- czas systemowy pobrany zaraz przed zapisaniem pliku, np.:
  Total Items: 660000
- całkowita ilość okresów czasu dla wszystkich dróg, które mogą lub nie być korkami ulicznymi, np.:
  Total Congestion: 4933
- ilość okresów z poprawnie przewidzianymi zagęszczeniami, np.:
  True Positive: 4357
- ilość okresów z poprawnie przewidzianym brakiem zagęszczeń, np.:
  True Negative: 654430
- ilość okresów w trakcie których nie było zagęszczenia, a moduł je przewidywał, np.:
  False Negative 576

- ilość okresów w trakcie których pojawiło się zagęszczenie w ruchu, a moduł go nie przewidział, np.:
  False Positive 637
- w kolejnych liniach poniżej prezentowane są wyniki przewidywania dla każdej z ulicy osobna. Każda linia zawiera:
  - nazwę drogi
  - ilość korków ulicznych na danej drodze
  - ilość poprawnie przewidzianych zagęszczeń[ułamek poprawnie przewidzianych zagęszczeń]
  - ilość błędnie przewidzianych zagęszczeń[ułamek błędnie przewidzianych zagęszczeń]
  - ilość poprawnie przewidzianych okresów bez zagęszczeń[ułamek poprawnie przewidzianych okresów bez zagęszczeń]
  - ilość błędnie przewidzianych okresów bez zagęszczeń[ułamek błędnie przewidzianych okresów bez zagęszczeń]

Przykładowa linia tekstu z wynikiem wygląda następująco:

C96C97: 1277 TP: 1162[0,910] FN: 115[0,090] TN: 596[0,824] FP: 127[0,176]