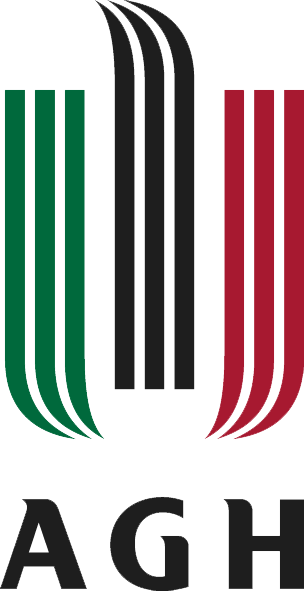
AKADEMIA GÓRNICZO – HUTNICZA

Inżynieria Oprogramowania



**Rozbudowa środowiska do modelowania i optymalizacji ruchu drogowego o różne algorytmy modelowania ruchu.**

Zespół projektowy w składzie:

*Małgorzata Majcherek, Aleksander Lech*

Spis treści

[1. Opis projektu 3](#_Toc359175618)

[1.1 System Kraksim 3](#_Toc359175619)

[1.2 Algorytm Nagla-Schreckenberga 3](#_Toc359175620)

[2. Nowe algorytmy 5](#_Toc359175621)

[2.1 Fazy korków 5](#_Toc359175622)

[2.2 Randomizacja zależna od prędkości (VDR) 5](#_Toc359175623)

[2.3 Model świateł stopu (BL) 6](#_Toc359175624)

[3. Zmiany w konfiguracji 8](#_Toc359175625)

[3.1 Property carMoveModel 8](#_Toc359175626)

[3.2 Szczegóły implementacyjne konfiguracji 9](#_Toc359175627)

[3.3 Przykladowa konfiguracja 11](#_Toc359175628)

[4. Analiza działania algorytmu VDR 12](#_Toc359175629)

[5. Analiza działania algorytmu BL 15](#_Toc359175630)

[6. Testy i porównanie działania zaimplementowanych algorytmów 19](#_Toc359175631)

[6.1 Konfiguracja testowa 19](#_Toc359175632)

[6.2 Czas symulacji w zależności od prawdopodobnieństwa zmniejszenia prędkości 19](#_Toc359175633)

[6.3 Prędkosć pojazdów w zależności od prawdopodobieństwa zmniejszenia prędkości 20](#_Toc359175634)

[6.4 Średnia prędkość i czas symulacji w zależności od progu odległości (model BL) 21](#_Toc359175635)

[6.5 Średnia prędkośc w zależności od parametru movingProb 24](#_Toc359175636)

[7. Wielopasmowośc w obecnym systemie 26](#_Toc359175637)

[8. Wielopasmowy model Nagla-Schreckenberga 27](#_Toc359175638)

[9. Wnioski 29](#_Toc359175639)

[10. Bibliografia 30](#_Toc359175640)

# Opis projektu

Tematem projektu jest rozszerzenie systemu Kraksim, służącego do modelowania i optymalizacji ruchu drogowego, o nowe algorytmy modelowania ruchu.

Stosowany do tej pory model Nagla-Schreckenberga ogranicza możliwości modelowania faz korków oraz natężonego ruchu drogowego.

Wprowadzenie bardziej skomplikowanych algorytmów umożliwi lepszą, bardziej rzeczywistą symulację sytuacji na drodze. Wybór algorytmu powinien być uzależniony od modelowanego środowiska -czy mapa reprezentuje centrum miasta, autostradę czy drogę podmiejską?

## System Kraksim

System Kraksim opiera się o wykorzystanie mikroskopowego modelu ruchu oparty o automat komórkowy– modeluje zachowanie pojedynczych samochodów. Każdy samochód posiada swoją prędkość i porusza się po drodze złożonej z pewnej ilości komórek, które reprezentują jednostkową odległość. Wszystkie parametry systemu są opisywane wartościami dyskretnymi, co upraszcza model, ale narzuca ograniczenia np. na prędkość pojazdów.

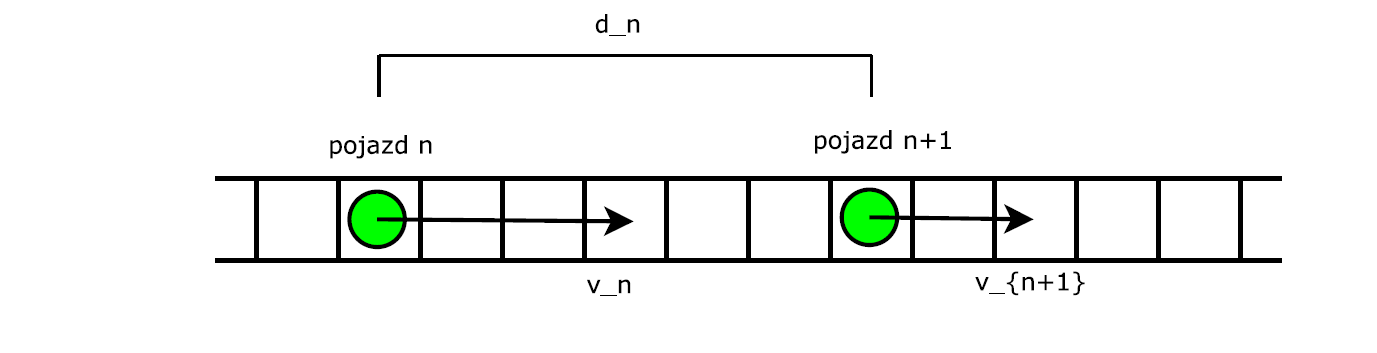
W systemie stosowane są następujące techniki: automaty komórkowe (modelowanie ruchu), system agentowy (optymalizacja sygnalizacji), uczenie ze wzmocnieniem (optymalizacja sygnalizacji), algorytmy wyszukiwania scieżki w grafie (optymalizacja trasy).

Jezykiem implementacji systemu Kraksim jest Java.

## Algorytm Nagla-Schreckenberga

Obecnie w systemie zaimplementowany jest stosunkowo prosty model Nagla-Schreckenberga. Droga reprezentowana jest jako tablica komórek, które moga byc puste lub zajete przez dokładnie jeden pojazd *n*. Pojazd posiada prędkosc z zakresu 0 do Vmax. Wszystkie parametry w tym modelu, takie jak odległość i prędkość, są dyskretne. Prędkosc wyrażana jest w komórkach na jendostkę czasu (turę). Tura składa się z czterech kroków obliczeniowych, które dobrano tak, aby nie dochodzilo do kolizji pojazdów.

Działanie tego algorytmu opiera się o prostą idee – w każdym kroku algorytmu pojazd odpowiednio zmienia swoją prędkość, a następnie przesuwa się o tyle komórek w kierunku jazdy, ile wynosi wartość jego nowej prędkości.



Model ruchu- jednowymiarowy automat komórkowy

Opis algorytmu :

**1. Przyspieszanie**: jeśli prędkość v jest mniejsza od Vmax, a odległość dn do pojazdu (n + 1) znajdującego się bezpośrednio przed analizowanym pojazdem jest większa niz (vn + 1), to prędkość pojazdu n wzrasta o jeden.

**2. Zwalnianie:** jeśli odległość dn do następnego pojazdu jest mniejsza niz vn + 1 to prędkość jest zmniejszana do wartości (dn - 1).

**3. Czynnik losowy:** wartość prędkości vn pojazdu n jest zmniejszana o 1 z prawdopodobieństwem p.

**4. Ruch:** mija tura i każdy pojazd przesuwa się o vn pól do przodu.

Algorytm bardzo dobrze opisuje ruch o małej gęstości. Jeśli chodzi o rejony o dużym zagęszczeniu, niedokładnie odwzorowuje fazy i rodzaje korków, a w szczególności brakuje fazy ruchu synchronicznego, co jednak w modelowaniu ruchu miejskiego nie stanowi aż tak poważnego problemu.

Dodanie nowych algorytmów oraz opcji ich wyboru przed startem symulacji umożliwiłoby lepsze odzwierciedlenie rzeczywistości przez model.

System został rozbudowany o wielopasowość, co wydłuża listę możliwych do dodania algorytmów.

# Nowe algorytmy

Rozbudowaliśmy systemu Kraksim o możliwość wyboru algorytmu modelowania ruchu, co za tym idzie dostosowaliśmy system do tworzenia takich algorytmów, wyboru odpowiednich opcji, zaimplementowaliśmy kilka z algorytmów bazujących na teorii Nagla-Schreckenberga, które uwzględniają nowe właściwości oraz więcej parametrów.

## Fazy korków

Głównym osiągnieciem nowych algorytmów jest poprawny opis korków. Jednak do zrozumienia różnic w zachowaniu modeli, potrzebna jest wiedza o trzech podstawowych fazach ruchu drogowego:

1. Faza ruchu swobodnego- ruch o małej gęstości, bardzo dobrze opisywany przy pomocy modelu Nagla-Schreckenberga.
2. Faza ruchu synchronicznego- faza ruchu, w której pojazdy poruszają się ze średnią prędkością mniejszą niż w fazie ruchu swobodnego, zagęsczenie pojazdów jets o wiele większe, natomiast nie dochodzi do trwałych korków. Pojazdy ze sobą ściśle oddziałowują, co wymusza prawie synchroniczne ruchy. Ta faza ruchu może zostać pominięta w symulacjach gęstej sieci dróg (częste skrzyżowania, krótkie odcinki jezdni między światłami), jednak jest niezbędna do prawidłowego opisu zagęszczonego ruchu na autostradzie.
3. Faza korków

## Randomizacja zależna od prędkości (VDR)

To bardziej realistyczny model ruchu drogowego. W tym modelu wprowadzony jest parametr losowy zależny od prędkości p(v). Parametr p(v) jest podobny do parametru p z poprzedniego modelu i opisuje prawdopodobieństwo zmniejszenia prędkości.

Najprostsza wersja wprowadza rozdział parametru dla pojazdów jadących oraz stojacych:

Obliczanie losowego parametru p(v) następuje przed pierwszym krokiem algorytmu Nagla-Schreckenberga. Po pierwszym i drugim kroku zostanie ustalona możliwa prędkosc jazdy. W kolejnym kroku zastosowany jest czynnik losowy (prędkosc może zostać zmiejszona do 0).

Jeśli p(n) ma większa wartosc dla pojazdów stojacych, to częściej obserwuje się opóźnienia w ruszaniu pojazdu z miejsca. Można w ten sposób przykładowo wprowadzić regułę powolnego startu, tzn. przypisywać parametrowi p wartość w zależności od tego czy dany uczestnik bierze udział w ruchu czy też dopiero startuje, takie działanie umożliwia lepszą symulację korków. Odpływ samochodów z korka jest mniejszy niż maksymalny przepływ, co powoduje, ze taki korek jest stabilny (nie powiększa się „do przodu” oraz nie zmniejsza się nienaturalnie szybko).

Model ten dobrze symuluje fazę ruchu swobodnego i fazę korków, ale nie oddaje charakterystyki ruchu synchronicznego.

## Model świateł stopu (BL)

Łączy regułę powolnego startu z metodą przewidywania prędkości. Zakłada, że kierowcy zmieniają prędkość reagując na światła stopu pojazdów jadących przed nimi. Model ten bardzo dobrze oddaje charakterystykę trzech podstawowych faz ruchu.

Zasada działania tego modelu po raz kolejny opiera się na modyfikacji parametru p oznaczającego prawdopodobieństwo zmniejszenia prędkości , jednak w tym przypadku nie zależy on tylko on prędkości danego uczestnika ruchu ale także od tego czy samochód będący odpowiednio blisko przed nim aktualnie nie hamuje.

Parametr losowy w takim modelu wygląda następująco:

Gdzie:

Model ten pozwala zachować nam regułę powolnego startu przy dodaniu kolejnych parametrów dlatego też dobrze oddaje charakterystykę wielu faz ruchu.

# Zmiany w konfiguracji

Do tej pory implementacja systemu Kraksim nie przewidywała możliwości zmiany modelu ruchu. W celu umożliwienia wyboru algorytmu oraz specyzowania jego parametrów zostaly wprowadzone zmiany w konfiguracji.

W pliku projekt/kraksim/configuration/kraksim.properties dodane zostało property *carMoveModel,* służące do sprecyzowania modelu ruchu i jego parametrów. Zgodnie z konwencją przyjętą przez poprzednich twórców, najpierw należy podać nazwę algorytmu, a po dwukropku nazwy i wartosci parametrów, oddzielone przecinkami.

Przykładowo:

*carMoveModel = vdr:movingProb=0.4,zeroProb=0.2*

## Property carMoveModel

Każdy z zaimplementowanych obecnie w systemie algorytmów może przyjmowac różne parametry. Niektóre z nich są wspólne, jak np prawdopodobieństwo zahamowania, inne- charakterystyczne dla danego modelu (np. próg odległości w BL).

Tabela obrazuje możliwe wartości property carMoveModel:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Nazwa** | **Możliwe parametry** | **Opis parametrów** |
| Nagla-Schreckenberga | nagle | decProb | Czynnik określający prawdopodobieństwo, z jakim prędkość pojazdu zostanie zmniejszona. |
| Randomizacja zależna od prędkości | vdr | movingProb | Czynnik określający prawdopodobieństwo, z jakim prędkość pojazdu zostanie zmniejszona dla pojazdów jadących. jeśli wartość zeroProb > movingProb, model wprowadza tzw. regułę powolnego startu. |
| zeroProb | Czynnik określający prawdopodobieństwo, z jakim prędkość pojazdu zostanie zmniejszona dla pojazdów stojących. |
| Światła stopu | bl | zeroProb | Czynnik określający prawdopodobieństwo, z jakim prędkość pojazdu zostanie zmniejszona dla pojazdów stojących. |
| movingProb | Czynnik określający prawdopodobieństwo, z jakim prędkość pojazdu zostanie zmniejszona dla pojazdów jadących. jeśli wartość zeroProb > movingProb, model wprowadza tzw. regułę powolnego startu. |
| brakeProb | Prawdopodobieństwo zahamowania w przypadku spełnionych warunków. |
| threshold | Parametr określa, w jakiej odległości musi znajdować się poprzedzający pojazd, żeby samochód reagował na jego światła stopu. |

## Szczegóły implementacyjne konfiguracji

Konfiguracja modelu ruchu jest całkowicie opakowana za pomocą klasy *pl.edu.agh.cs.kraksim.main.CarMoveModel.* Klasa ta przechowuje parametry modelu ruchu- jego nazwę oraz mapę zawierającą nazwę property i jego wartość. Są one używane w symulacji, do wybrania odpowiedniego mechanizmu sterowania samochodem.

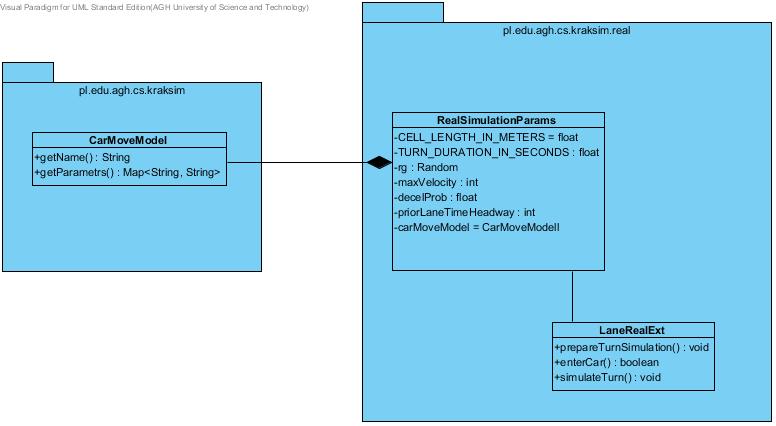
Obiekt tej klasy tworzony jest na podstawie konfiguracji odczytanej z pliku *kraksim.properties* lub na podstawie wyborów dokonanych w gui aplikacji.

Po utworzeniu obiekt ten umieszczany jest w instancji obiektu *pl.edu.agh.cs.kraksim.real.RealSimulationParams*, (która przechowuje wszystkie wartości niezbędne do symulacji).

RealSimulationParams są przekazywane do konstruktora właściwej dla symulacji klasy *pl.edu.agh.cs.kraksim.real.LaneRealExt*. W klasie tej znajduje się kluczowa metoda- *simulateTurn*. To w niej, na postawie parametru CarMoveModel odbywa się symulacja kolejnych tur.

Obiekt CarMoveModel ten jest więc łatwo dostępny dla metod które odpowiadają za symulacje ruchu.

Zależności pomiędzy opisanymi klasami obrazuje diagram klas:



## Przykładowa konfiguracja

Po wprowadzonych modyfikacjach przykładowy plik kraksim.properties wygląda następująco:

|  |  |
| --- | --- |
| *cityMapFile = maps\\6x6map.xml*  *travelSchemeFile = maps\\6x6\_120000\_new3.xml*  *statOutFile = maps\\debug.bl.test3.1*  *lastSessionFile = lastSession.properties*  *workDir = .*  *#visualization = false*  *visualization = true*  *algorithm = sotl:zone=18*  *#algorithm = rl*  *#algorithm = static*  *dynamicRouting=true*  *#dynamicRouting=false*  *#enablePrediction = true*  *enablePrediction = false*  *predictionModule = weka*  *#predictionModule = pattern*  *predictionFile = configuration\\prediction.strong\_big.config.xml*  *yellowTransition = 3*  *#turnStats = true*  *#turnStatsQuietPeriod = 1000*  *minimalSpeedUsingPrediction = false*  *globalUpdateInterval = 100*  *#Parametry dla optymalizacji ruchu opartego o miary SNA*  *#czy algorytm wlaczony*  *snaEnabled=true*  *#liczba klastrów*  *snaClusters=4*  *#co ile kroków ma sie odbywac przeliczenie miar i klastrowanie*  *snaRefreshInterval=100*   |  | | --- | | *#model ruchu*  *carMoveModel = bl\:movingProb\=0.4,zeroProb\=0.4,brakeProb\=0.8,threshold\=1* | |

# Analiza działania algorytmu VDR

Porównaliśmy działanie zaimplementowanego algorytmu oraz istniejącego już modelu Nagla-Schreckenberga.

Algorytmy modelujące ruch zostały uruchomione z następującymi parametrami:

* + algorytm Nagla-Schreckenberga z domyślną wartością prawdopodobieństwa zmniejszenia prędkości która wynosi 0.2
  + algorytm VDR z następującymi wartościami parametrów:
    - 0.9 - prawdopodobieństwo zmniejszenia prędkości przy starcie pojazdu (gdy prędkość jest równa 0 )
    - 0.2 - prawdopodobieństwo zmniejszenia prędkości podczas ruchu (gdy prędkość > 0)

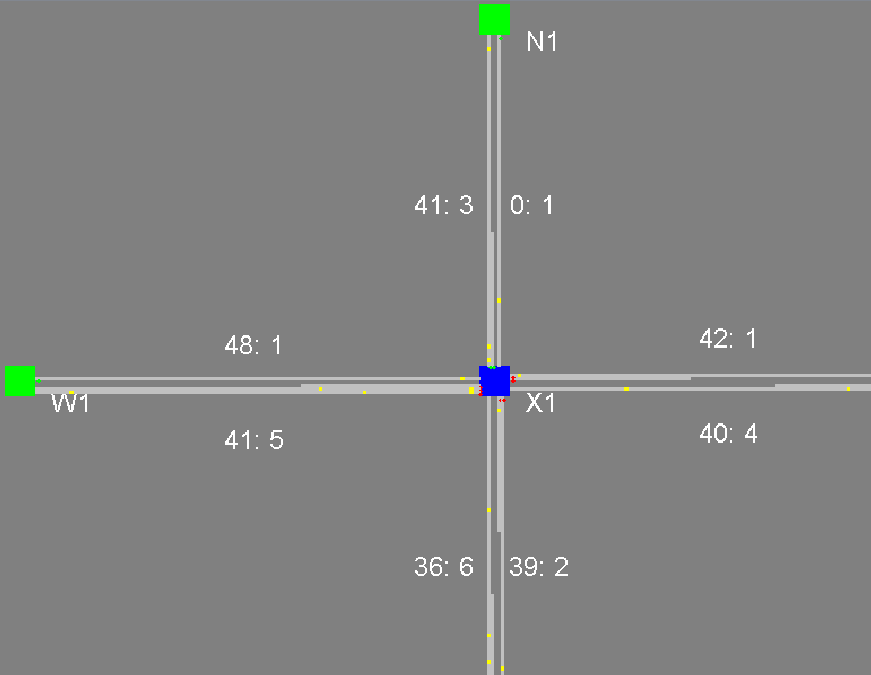
Po analizie działania oraz obserwacji zmian ruchu zauważyć można znaczącą różnicę w zachowaniu pojazdów. Przy tak dobranych parametrach i użyciu algorytmu VDR pojazdy mają małą szansę na ruszenie z miejsca, przez co na skrzyżowaniach szybko tworzą się korki.

Z kolei przy użyciu modelu Nagla-Schreckenberga ruch był bardziej płynny, a na skrzyżowaniach nie tworzyły się korki.

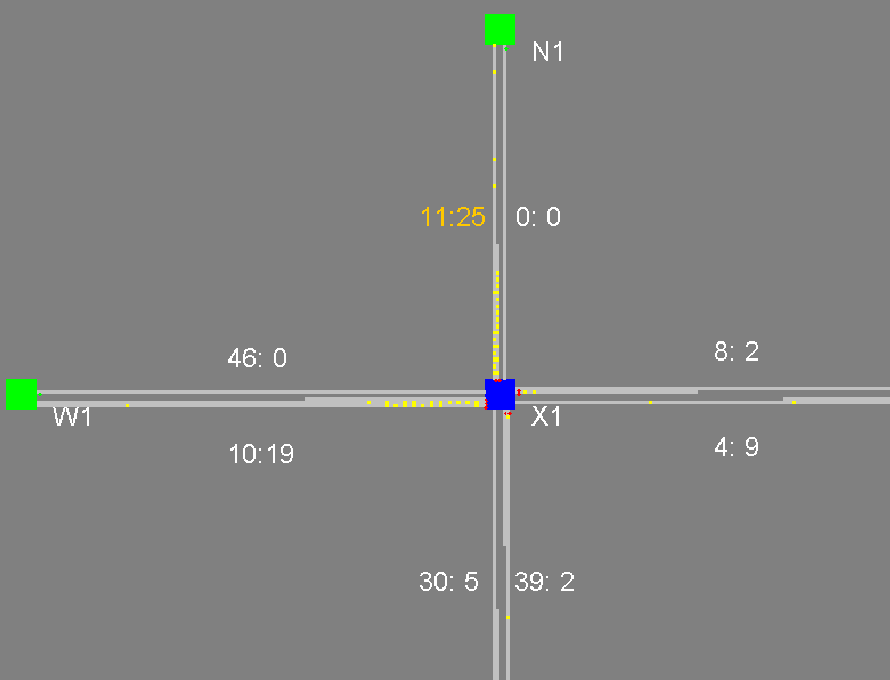
Należy zwrócić uwagę, iż takie zachowanie wynika jedynie z dobranych parametrów. Jeśli parametry obu algorytmów mają podobne wartości to symulacja przebiega podobnie. Gdy prawdopodobieństwo zmniejszenia prędkości przy starcie jest większe niż 0.2, tworzą się mniejsze korki.

Widoczne zmiany w zachowaniu pojazdów po zmianie modelu pozwoliły nam się upewnić, że model randomizacji zależnej od prędkości został zaimplementowany prawidłowo.

Stan mapy podczas symulacji z użyciem modelu Nagla - Schreckenberga



Stan mapy podczas symulacji z użyciem modelu VDR



# Analiza działania algorytmu BL

Porównaliśmy działanie zaimplementowanego algorytmu oraz istniejącego już modelu Nagla-Schreckenberga.

Algorytmy modelujące ruch zostały uruchomione z następującymi parametrami:

* + algorytm Nagla-Schreckenberga z domyślną wartością prawdopodobieństwa zmniejszenia prędkości która wynosi 0.2
  + algorytm BL z następującymi wartościami parametrów:
    - 0.6 - prawdopodobieństwo zmniejszenia prędkości przy starcie pojazdu (gdy prędkość jest równa 0 )
    - 0.2 - prawdopodobieństwo zmniejszenia prędkości podczas ruchu (gdy prędkość > 0)
    - 0.8 – prawdopodobieństwo zmniejszenia prędkości przy spełnionych warunkach
    - 5 – próg odległości

Po analizie działania oraz obserwacji zmian ruchu zauważyć można znaczącą różnicę w zachowaniu pojazdów. Przy tak dobranych parametrach i użyciu algorytmu BL pojazdy mają małą szansę na ruszenie z miejsca, przez co na skrzyżowaniach szybko tworzą się korki.

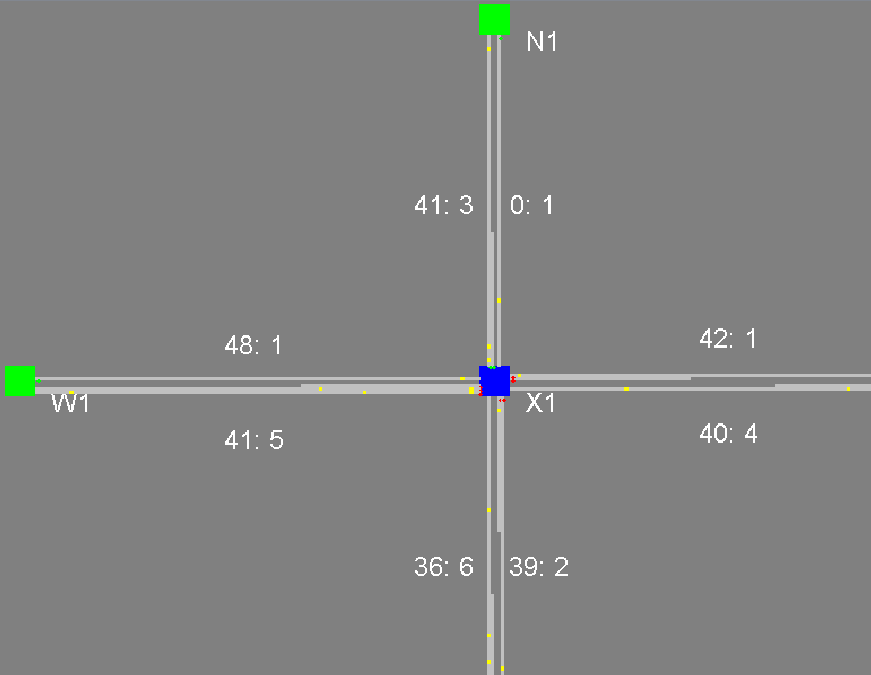
Jednak zanim powstał korek, a ruch był już zagęszczony, można było zaobserwować synchroniczny ruch pojazdów.

Z kolei przy użyciu modelu Nagla-Schreckenberga ruch był bardziej płynny, a na skrzyżowaniach nie tworzyły się korki.

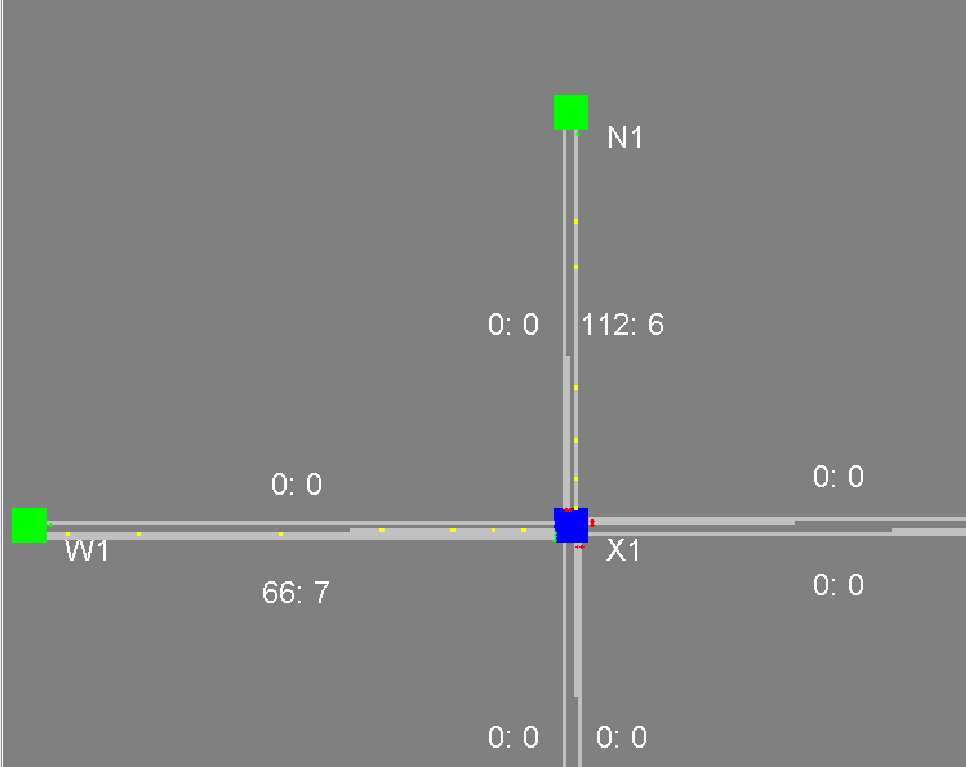
Należy zwrócić uwagę, iż takie zachowanie wynika jedynie z dobranych parametrów. Jeśli parametry obu algorytmów mają podobne wartości to symulacja przebiega podobnie. Gdy prawdopodobieństwo zmniejszenia prędkości przy starcie jest większe niż 0.2, tworzą się mniejsze korki.

Widoczne zmiany w zachowaniu pojazdów po zmianie modelu pozwoliły nam się upewnić, że model randomizacji zależnej od prędkości został zaimplementowany prawidłowo.

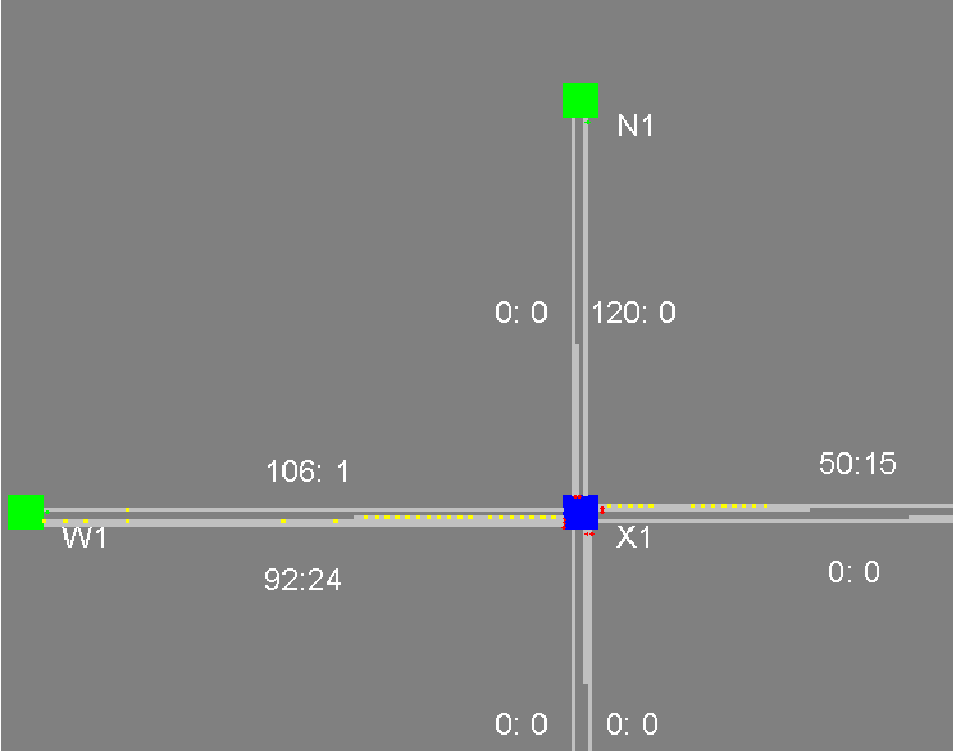
Symulacja z użyciem modelu Nagla-Scgreckenberga:



Symulacja fazy ruchu synchronicznego, model świateł stopu:



Symulacja fazy korku, model świateł stopu:



# Testy i porównanie działania zaimplementowanych algorytmów

Przeprowadziliśmy serię mających na celu stwierdzenie, jak zmieniają parametry symulacji w zależności od zastosowanego algorytmu.

Pod uwagę wzięliśmy 2 wartości:

* Średni czas symulacji
* Średnią prędkość pojazdów

Obie te wartości z łatwością mogliśmy odczytać z pliku ze statystykami uzupełnianego po każdej zakończonej symulacji.

## Konfiguracja testowa

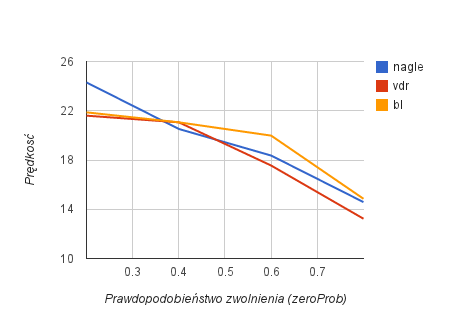
Konfiguracją testowa dla była mapa z prostym schematem ruchu, w którym 200 samochodów próbuje przejechać przez jedno skrzyżowanie w kolidujących kierunkach.

Pozwala to uwzględnić dużo elementów ruchu (swobodny dojazd do skrzyżowania, zatrzymanie się, rozpoczęcie ruchu i kontynuowanie go aż do celu).

## Czas symulacji w zależności od prawdopodobnieństwa zmniejszenia prędkości

Ponieważ każdy z aglorytmów przyjmuje różne parametry, które wpływają znacząco na jego zachowanie, ciężko obiektywnie porównać czas działania. Postanowiliśmy za wspólną wartośc uznać prawdopodobnieństwo zmiejszenia szybkości, który, choć traktowany inaczej w każdym modelu, jest parametrem wspólnym.

Można zauważyć, że najszybciej wykonuje się symulacja z użyciem modelu Nagla-Schreckenberga. Jest to najprostszy model, uwzględnia najmniej czynników. Parametry w pozostałych algorytmach zostały dobrane tak, aby zachowywac regułę powolnego startu, przez co dobrze odwzorowują korki. Symulacja trwa dłużej, jednak jest bliższa rzeczywistości.

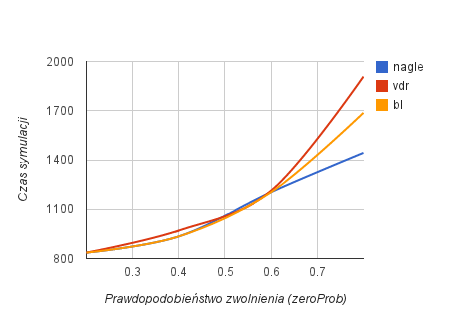
Co oczywiste, razem ze wzrostem prawdopodobieństwa zwolnienia samochodom jest “cieżej” przyspieszyć lub ruszyć z miejsca, więc wydłuża się czas symulacji.

## Prędkosć pojazdów w zależności od prawdopodobieństwa zmniejszenia prędkości

Dla porównania średniej prędkosci pojazdów również jako wartośc wspólną przyjęliśmy prawdopodobieństwo zwolnienia.

Wyniki sa zgodne z przewidywaniami- im większe prawdopodobieństwo zmiejszenia szybkości, tym mniejsza prędkosc pojazdów. Największy spadek prędkosci następuje zdecydowanie dla algorytmu randomizacji zależnej od prędkości. Ma to odbicie w teorii- model Nagla-Schreckenberga w ogóle nie symuluje korków, wiec prędkość jest największa. Z kolei w model swiateł stopu występuje ruch synchroniczny przed korkiem i drastyczną zmianą prędkości.

Wykres przedstawiaja czas działania w symulacji wszystkimi trzema algorytmami, w zależności od prawdopodobieństwa zmniejszenia prędkości.



## Średnia prędkość i czas symulacji w zależności od progu odległości (model BL)

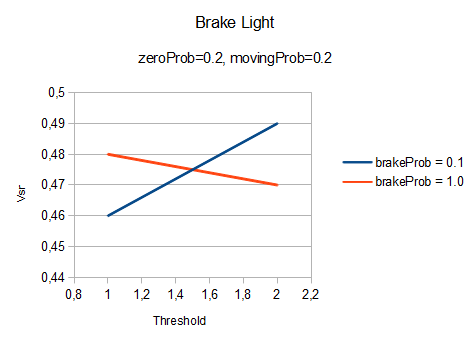
Kolejnym testem jaki przeprowadzono w celu zgromadzenia danych była analiza wyniku działania algorytmu Brake Light zależnie od ustalonej odległości powodującej reakcję kolejnych kierowców.

W systemie domyślną wartością maksymalną prędkosci jest 2. Jest to wartośc sensowna (równa 52 km/h) w warunkach symulacji ruchu miejskiego.

Z racji takiego ograniczenia, inne wyniki dają tylko dwie wartośći : 1 oraz ≥2 (ponieważ samochód nigdy nie przyspieszy do 3, nigdy nie sprawdzi, czy kierowca odległy od niego o 3 komórki zwalnia).

Testy zostały przeprowadzone dla dwóch wartości parametru brakeProb (0.1 oraz 1.0).

Wyniki przedstawione są na wykresie :



Na potrzeby analizy zmieniliśmy domyślną maksymalną wartość prędkości na 5.

Jest to zbyt wysoka wartość do realistycznej symulacji ruchu miejskiego (aż 135 km/h!), jednak tylko w ten sposób mogliśmy sprawdzić jak zmieniają się parametry symulacji.

W celu zwiększenia maksymalnej prędkosci zmiany można dokonać w pliku xml opisującym mapę miasta za pomocją parametru:

*<roads defaultSpeedLimit="5">*

lub w wartosciach domyślnych parametru DEFAULT\_MAX\_VELOCITY w klasie *pl.edu.agh.cs.kraksim.real.RealSimulationParams.*

Testy zostały przeprowadzone z następującymi parametrami algorytmu BL:

* *movingProb=0.4*
* *zeroProb=0.4*
* *brakeProb=0.8*

Przy jednakowych pozostałych parametrach, sprawdziliśmy jak zwiększanie wartości parametru threshold wpływa na symulację.

Można zauważyć, że jeśli brane pod uwagę są tylko samochody będące bezpośrednio przed pojazdem (oddalone o 1), to czas symulacji jest znacząco krótszy. Im więcej danych o zwalniających samochodach model musi przeanalizować, tym dłuższy czas symulacji.

Wartości parametru threshold większe niż maksymalna prędkość nie mają sensu- ich zwiększenie nie ma wpływ na symulację i zachowanie pojazdów.



Wielkość parametru wpływa też na prędkośc pojazdu- reagowanie na większą ilość świateł stopu, zmniejsza średnią prędkość.



## Średnia prędkośc w zależności od parametru movingProb

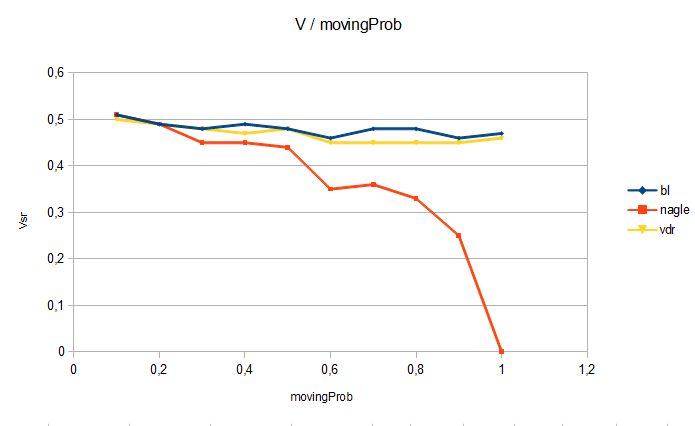
Następnie porównaliśmy wszystkie dostępne w obecnym momencie algorytmy pod względem zachowania w zależności od podobnego we wszystkich parametru jakim jest movingProb (decProb w nagle).

Jak widać wyniki się pogarszają wraz ze wzrostem wartości (należy jednak zwrócić uwagę iż każdy z nich w jakimś stopniu zależy od generowanych liczb losowych, wobec czego wyniki te cechują się pewną dozą prawdopodobieństwa. Najwyraźniej widać to w przypadku algorytmu Nagla-Schreckenberga- nie ma tam wyróżnionego przypadku dla prędkości równej 0.

Widać że dla wartości 1 oraz algorytmu Nagla-Schreckenberga. W tej sytuacji każdy pojazd na mapie ma 100% szansy na zmniejszenie prędkości (do 0). Raz zatrzymany samochód nie powróci już do ruchu przy takich parametrach.

Dla pozostałych algorytmów sytuacja nie jest aż tak drastyczna- samochody, które stoją mają osobny parametr. Pomimo, że kazdy jadący samochód się zatrzyma, ma on szanse ruszyć w kolejnej turze.

Po raz kolejny mogliśmy wiec zaobserwowac różnice w symulacji, jakie wprowadzają różne modele ruchu.



# Wielopasmowośc w obecnym systemie

W obecnej postaci system posiada zaimplementowany dosyć prosty model wielopasowy. Równoważenie obciążenia opiera się na zliczaniu ilości samochodów znajdujących się na poszczególnych pasach

oraz odległości najbliższego samochodu od wjazdu ze skrzyżowania. Samochody wybieraja pas na skrzyżowaniu po czym trzymają się go, ewentualnie zmieniając na ten bliższy strony docelowej (skrętu na skrzyżowaniu).

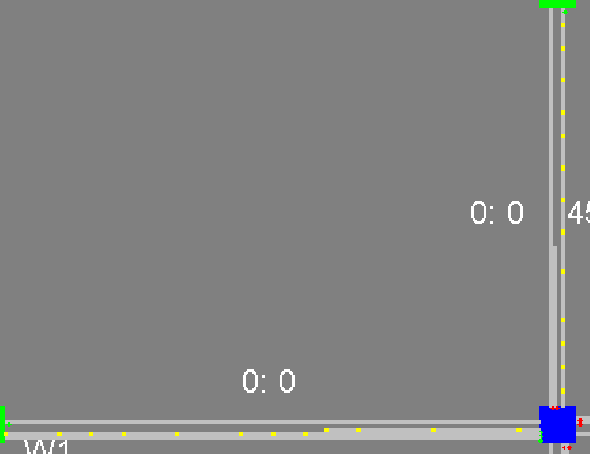
Pierwszeństwo wprowadzania samochodów na pasy jest następujące:

* pas z najmniejszą ilością samochodów
* wśród pasów z równą ilością samochodów – pas, na którym jest największa odległość do kolejnego samochodu
* jeśli powyższe kryteria nie pomogą – pas najbardziej po prawej.

Jest to rozwiązanie zgodne z logiką oraz przepisami, jednak ma też swoje wady.

Powoduje to, że nawet kilka pasów jezdni nie jest wykorzystane jeśli wszystkie samochody chcą skręcić (rysunek). Oczywiście jest to w jakiś sposób optymalna strategia, powodująca że nie blokują się pasy dla innych kierunków. Jednak w miarę możliwości chcielibyśmy spróbować implementacji innych zachowań, tak by system odzwierciedlał także inne sytuacje. Na przykład kierowców, którzy próbują skręcić dopiero później niż moment w którym znajdą się obok pasa dojazdowego. W podobny sposób zachowują się kierowcy jadący prosto , po wyborze pasa już nie próbują go zmieniać niezależnie od panujących na nim warunków czy też jakieś ustalonego prawdopodobieństwa.

Obecne rozwiązania implementuje klasa pl.edu.agh.cs.kraksim.real.MultiLaneRoutingHelper oraz metoda pl.edu.agh.cs.kraksim.real.LinkRealExt.enterCar(Car, int, int).



# Wielopasmowy model Nagla-Schreckenberga

Jezdnia wielopasmowa traktowana jest tu jako kilka równoległych jezdni jednopasmowych, z dodatkowymi regułami określającymi zmianę pasów.

Celem tych reguł jest zbadanie opłacalności zmiany pasa oraz sprawdzenie, czy zmiana ta jest możliwa. Sprawdzenie możliwości zmiany pasa naśladuje zachowanie kierowcy, tj. polega na zmierzeniu dystansów do najbliższych pojazdów z przodu i tyłu na docelowym pasie i odniesieniu ich do aktualnej prędkości.

Pojazd może zmieniać pas ruchu w celu wyprzedzenia. Innym celem zmiany pasa ruchu przez kierującego pojazdem, jest ustawienie sie na odpowiedniej pozycji do przejazdu przez skrzyżowanie.

Po analizie kodu źródłowego okazało się że w obecnej chwili implementacja modelu ruchu (Nagla-Schreckenberga) znajduje się w klasie *pl.edu.agh.cs.kraksim.real.LaneRealExt* a konkretnie odpowiada za niego metoda *simulateTurn*.

# Wnioski

Model ruchu samochodów jest kluczowym elementem realistycznej symulacji ruchu drogowego. Oczywiście na ruch pojazdów mają wpływ też inne czynniki- jak ustalona odgórnie maksymalna prędkość, czy algorytm sterujący światłami. Jednak bez odpowiednio dobranego modelu ruchu oraz jego parametrów, wyniki symulacji mogą znacząco mijac się z rzeczywistością.

Dlatego konieczna jest znajomość kilku modeli ruchu oraz wiedza, co najlepiej odzwierciedlają oraz jaki wpływ na ich zachowanie mają parametry. Nawet poprawnie dobrany algorytm przy nierealnych wartościach parametrów (np, przy założeniu że prawdopodobieństwo ruszenia z miejsca stojącego pojazdu jest bardzo małe, albo że kierowca reaguje na światła stopu pojazdu jadacego daleko przed nim) nie zadziała prawidłowo.

Najbardziej znaczącym parametrem tych algorytmów jest prawdopodobieństwo zwolnienia, wysztępujące w każdym modelu. Bez niego pojazdy poruszają się w sposób deterministyczny, z każdą turą zbliżając się co celu- jednak nie oddaje to ruchu rzeczywistych pojazdów. Dopiero wprowadzenie czynnika losowego w trzeciej turze dodaje symulacji realizmu. Parametr ten nalezy więc dobrać ze szczególną dokładnością.

Z naszych obserwacji zachowań pojazdów w róznych modelach, oraz z literatury wynika, że każdy z zaimplementowanych modeli nadaje się do symulacji innych sytyacji.

**Model Nagla-Schreckenberga**, najprostszy z dostępnych algorytmów, bardzo dobrze odzwierciedla swobodny ruch, a symulacje przy jego użyciu są najszybsze. Warto więc go stosować do wizualizacji ruchu pojazdów na drogach miejskich o małym nateżeniu ruchu. Nie nadaje się jednak do symulacji korków.

**Model randomizacji zależnej od prędkości** dobrze oddaje charakterystykę dwóch faz ruchu- ruchu swobodnego oraz korków. Dlatego powinien być stosowany do symulacji ruchu miejskiego o dużym natęzeniu (krótkie odcinki, dużo skrzyżowań). W tych warunkach zazwyczaj nie występuje faza ruchu synchronicznego, więc jej brak nie wpływa na realizm prezentacji.

Nalezy też pamiętać o odpowiednim dobraniu prawdopodobieństwa zmiejszenia prędkości- jeśli chcemy symulować tzw „szerokie korki”, samochodom stojącym powinno być ciężej ruszyć.

**Model świateł stopu** jako jedyny z tych trzech algorytmów modelu fazę ruchu synchronicznego. Jest ona charakterystyczna dla długich odcinków, gdzie przy zwiększajacym się natężeniu pojazdów, najpierw jadą one wolniej, później dopiero następuje faza stania w korku. Model ten więc świetnie obrazuje natężony ruch na autostradzie lub obwodnicy.

# Bibliografia

1. Bartosz Rybacki, „*Modelowanie i optymalizacja ruchu miejskiego przy uzyciu wybranych technik”*
2. Łukasz Dziewoński, Maciej Zalewski, *„Kraksim- wielopasmowość i predykcja ruchu”*
3. Wikipedia, artukuł o modelu Nagla-Schreckenberga <http://en.wikipedia.org/wiki/Nagel%E2%80%93Schreckenberg_model>