AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA INFORMATYKI

PRACA DYPLOMOWA MAGISTERSKA

Koordynacja ruchu pojazdów na skrzyżowaniu dróg wielopasmowych algorytmem planowania wielowariantowego

Autor: ***Piotr Kala***
Kierunek studiów: ***Informatyka***
Opiekun pracy: ***dr inż. Wojciech Turek***

Kraków, 2016

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

…….

Składam szczególne podziękowania mojemu Promotorowi, dr hab inż. ?????, profesorowi nadzwyczajnemu AGH za życzliwość, cenne uwagi merytoryczne, wszechstronną pomoc oraz poświęcony czas.

Spis treści

Wstęp

1.1 Dziedzina Problemu

W pracy rozwiązywany jest problem planowania ruchu na skrzyżowaniach.

W rozwiązaniu wykorzystany jest algorytm A*, który najczęściej służy do wyszukiwania najkrótszej ścieżki w grafie.

Poruszony jest także problem unikania kolizji wraz z użyciem algorytmu A*.

Omówiona jest trudność w planowaniu ruchu globalnego jakim jest złożoność, która rośnie wraz z rozmiarem środowiska.

Planowanie ruchu na skrzyżowaniach wiaże się z czestymi zmianami środowiska. W związku z tym system planujący dokładnie ruch powinien być wstanie szybko reagować na pomyłki.

1.2 Specyfikacja problemu

Problemem przedstawionym w pracy jest szybkie i bezpieczne zaplanowanie ruchu na skrzyżowaniach.

Mając określony stan początkowy będącym położeniem wszystkich pojazdów na skrzyżowaniu, system zwraca dokładny plan, który prowadzi do szybkiego i bezpiecznego opuszczenia

przez samochody skrzyżowań.

Problem polega na znalezieniu kolejnych stanów prowadzących do optymalnego rozwiązania. Pojedyńczy stan oraz jego zmiany wprowadzają dużą liczbę możliwości.

Dla przykładu pojazd musi mieć określoną pozycję na drodze na której się znajduje oraz prędkość. W kolejnych stanach pojazd może zmienić położenie, przyspieszyć, zwolnić lub prędkości nie zmieniać.

Kolejną trudnością w problemie, jest unikanie kolizji. Dla wszystkich możliwość algorytm planowania powinien usunąć wszystkie możliwości kolizji na jednym pasie oraz przy przekraczaniu przez skrzyżowanie.

1.3 Cel

Celem pracy jest opracowanie metody planującej ruch drogowy na skrzyżowaniach dróg wielopasmowych przy użyciu zmodyfikowanego algorytmu A*.

Wynikiem rozwiązania jest szczegółowy opis, który dokładnie planuje ruch każdego z samochodów w celu najszybszego i bezpiecznego opuszczenia skrzyżowania.

System zapewnia unikanie kolizji oraz optymalny dojazd do celu pojazdów na zadanej sieci skrzyżowań.

1.4 Struktura pracy

TODO

Istniejące rozwiązania w zakresie planowania ruchu i koordynacji ruchu na skrzyżowaniach

2.1 Planowanie ruchu na skrzyżowaniach

Cel - lista wymagań i cech mojego rozwiązania - tego nie będzie w treści

- Zastosowanie algorytmu A* do zaplanowania ruchu na skrzyżowaniach
- Brak świateł powinno przyspieszyć ruch na szkyżowaniach - auta nie czekają na świetle pomarańczowym
- Rozwiązanie uwzględnia unikanie kolizji
- A* powinien szybciej zaplanować drogę niż np. Dijkstra
- W rozwiązaniu zapewnione jest bezpieczeństwo - nie dochodzi do kolizji
- Rozwiązanie koordynuje ruch globalnie

Planowanie ruchu zawsze wiąże się z rozmiarem problemu. Dla koordynacji ruchu na skrzyżowaniach o rozmiarze problemu stanowi rozmiar dróg, ich liczba oraz liczba przecięć między nimi, a także parametry i liczba pojazdów. Z planowaniem ruchu wiąże się trudność jaką jest szybkość zmian w środowisku. W przypadku nie zastosowania się pojazdu do planu, algorytm musi natychmiast wyliczyć poprawiony plan. System planujący ruch drogowy, po-

winien być więc systemem czasu rzeczywistego. Cechą takich systemów, jest równoległość w czasie zmian w środowisku oraz prowadzonych przez system obliczeń opierających się o stan środowiska. Algorytmy heurystyczne są najczęściej stosowane w wielokrotnie zmieniających się środowiskach.

Popularnym modelem reprezentującym sieci dróg jest graf. Wierzchołki grafu to skrzyżowania dróg. Krawędzie natomiast, reprezentują odległości między skrzyżowaniami za pomocą wag. Często stosowanym algorytmem do znajdowania najkrótszej ścieżki w grafie, jest algorytm Dijkstry [18], [1], [12], [11], [4], [13]. O rzędzie złożoności algorytmu decyduje implementacja kolejki. W przypadku implementacji kolejki poprzez kopiec, złożoność algorytmu to:

$$O(E + V \log V)$$

gdzie E to liczba krawędzi grafu, a V liczba wierzchołków grafu.

Jednym z algorytmów heurystycznych, stosowanych do szukania najkrótszej ścieżki, jest Algorytm A* [16], [17], [19], [8], [6]. Złożoność czasowa algorytmu, jest zależna od zastosowanej funkcji heurystyki. Jeśli funkcja heurystyki spełnia następujący warunek:

$$|h(x) - h^*(x)| = O(\log h^*(x))$$

gdzie h^* jest optymalną heurystyką - liczba przeszukanych węzłów rośnie wielomianowo w stosunku do długości rozwiązania.

W artykule [19] autorzy przedstawili algorytm A*, w celu planowania wielowymiarowego. W celu przybliżenia złożoności jako przykładowe środowisko podana została dwuwymiarowa, dyskretna przestrzeń kwadratowa o boku 7. Pojazdy mogą zajmować wówczas 49 pozycji oznaczonych współrzędnymi (x, y). Dla jednego pojazdu, możliwość zajmowanych pozycji wynosi $(7 * 7) = 49$. Dla dwóch pojazdów jest to już $(7 * 7) * (7 * 7 - 1) = 2352$. Aktualny stan jest wówczas opisywany jako $(x1, y1, x2, y2)$. Złożoność rośnie wraz z przetrzymywaniem kolejnych informacji na temat pojazdów, czyli na przykład prędkości czy przyspieszenia.

Autorzy artykułu [15] zwrócili uwagę na podział planowania na globalne i lokalne. W podejściu globalnym, dla algorytmu musi być znane całe środowisko, co wiąże się z dużym obciążeniem pamięciowym - algorytm przechowuje informacje na temat stanów wszystkich pojazdów na skrzyżowaniach. W podejśyiu lokalnym, środowisko dla pojedynczego pojazdu nie jest znane, algorytm poznaje je w czasie rzeczywistym. W przedstawionym rozwiązaniu zastosowane jest podejście globalne.

2.2 Planowanie ruchu przy użyciu świateł drogowych

Najczęściej stosowanym systemem do koordynacji ruchu na skrzyżowaniach są światła drogowe. Powodują one opóźnienia wiążące się z oczekiwaniem przy świetle pomarańczowym, które jednocześnie zapewnia bezpieczeństwo. Powstało wiele optymalizacji świateł drogowych w celu przyspieszenia ruchu na skrzyżowaniach.

W artykule [3] autorzy opisują następujące optymalizacje sygnalizacji świetlnej: *Synchronized Traffic Lights, Green Wave, Random Offset*. Synchroniczne światła drogowe oparte są o obliczane cykle oraz o klastry samochodów poruszających się po skrzyżowaniach. Klaster samochodów rusza, gdy pojawi się światło zielone. Pierwszy samochód rusza z największym przyspieszeniem, aż dojedzie do następnego skrzyżowania. Wówczas natychmiast włącza się światło czerwone i reszta samochodów w klastrze dojeżdża do skrzyżowania. Cykle są obliczane w sposób, aby opóźnienia były najmniejsze. Zgodnie ze zdaniem autorów, jedną z wad jest fakt, że rozwiązanie działa najlepiej dla cykli, które nie występują na rzeczywistych skrzyżowaniach. W celu usprawnienia, autorzy zaproponowali strategię zielonej fali. Głównym celem tego podejścia jest to, aby duże klastry samochodów były w ruchu. W artykule, strategia zielonej fali daje lepsze efekty niż strategia synchronicznych świateł. Ostatnią przedstawioną strategią jest losowa zmiana świateł przed skrzyżowaniami. W przeciwieństwie do poprzednich strategii - światła na skrzyżowaniach nie przełączają się jednocześnie. Użycie tej strategii powoduje powstawanie dużych klastrów samochodów na niektórych drogach i szybki przepływ na innych. Globalnie, strategia zielonej falii przynosiła najlepsze efekty. Przedstawione rozwiązania, muszą jednak stosować światło pomarańczowe w celu utrzymania bezpieczeństwa, co powoduje opóźnienia na każdym skrzyżowaniu.

Autorzy artykułu [2] opisują koordynację ruchu przy użyciu algorytmu 'REAL TIME

QUEUE LENGTH ESTIMATION: THE APTTC ALGORITHM'. Koordynacja ta działa przy użyciu tablicy sensorów fotoelektrycznych rozmieszczonych na drogach przy skrzyżowaniach. Algorytm w zależności od danych i estymacji kolejek, odpowiednio steruje sygnalizacją świetlną w celu uniknięcia opóźnień. Algorytm został uruchomiony w mieście Chennai w Indiach, gdzie wyestymowane zostały długości kolejek aut na drogach poprzez zamontowane sensory na każdy dzień tygodnia. Rozwiązanie jest jednak zaprojektowane do działania na jednym skrzyżowaniu oraz nie eliminuje opóźnień spowodowanych światłem pomarańczowym. Co więcej, w rozwiązaniu konieczne jest montowanie sensorów na drogach.

W artykule [5] opisane zostało kierowanie ruchem drogowym na jednym skrzyżowaniu poprzez optymalizację kolejek. W rozwiązaniu brane pod uwage są takie dane, jak średni czas oczekiwania, średnia długość kolejki i najgorszy czas oczekiwania. Istnieje także możliwość przekazania parametrów odpowiadających za maksymalny czas trwania światła czerwonego oraz zielonego. Wynikiem rozwiązania jest schemat zmiany świateł, który liczony jest za pomocą funkcji obiektowych na podstawie długości kolejek. W rozwiązaniu pominięty został element bezpieczeństwa jakim jest światło pomarańczowe. Rozwiązanie zostało pokazane także na jednym skrzyżowaniu, a nie na całej sieci skrzyżowań.

Autorzy artykulu [14] przedstawili system kontroli świateł i przepływu pojazdów na sieciach drogowych. Rozwiązanie stosowane jest dla całych sieci skrzyżowań. Autorzy opierają się o krótkie prognozu ruchu i na ich podstawie liczą długość trwania światła zielonego. Klasyczne rozwiązania działają na wcześniej obliczonych danych oraz działają we wcześniej określonych miejscach największego ruchu. Takie rozwiązania powodują, że zielona fala trwa dużej niż powinna. W rozwiązaniu zastosowana jest także strategia stabilizacji, która zapewnia fakt, iż długość kolejek nie będzie zbyt długa. Zapewnione jest to przez odpowiednią długość trwania światła zielonego. W rozwiązaniu, na każdym ze skrzyżowań, samochody będą tracic czas na świetle pomarańczowym, które w rozwiązaniu jest konieczne, aby było ono bezpieczne. Ruch jest także optymalizowany dla sytuacji "średnich", które tak na prawdę nigdy nie występują. Wynikiem tego zjawiska jest fakt, że optymalizacja nigdy nie jest stosowana dla prawdziwej, aktualnej sytuacji.

W artykule [9] przedstawiony został system zarządzania światłami za pomocą komunikacji pojazdów między sobą. Działanie systemu opiera się o sensory zamontowane przy skrzyżowaniach oraz urządzenia DSRC zamontowane w każdym z aut. Pojazdy komunikują

się ze sobą za pomocą sieci bezprzewodowej, posiadają te same wersje map oraz posiadają GPS z dokładnością do pasa na którym się znajdują. W zależności od zebranych danych system odpowiednio przełącza światła w celu uniknięcia opóźnień. Przedstawione rozwiązanie jest kosztowne oraz pominięta jest kwestia bezpieczeństwa, która w innych rozwiązaniach zapewniona jest poprzez oczekiwanie na świetle pomarańczowym.

2.3 Planowanie ruchu bez sygnalizacji świetlnej

W celu uniknięcia użycia świateł drogowych, istnieją także rozwiązania z wykorzystaniem algorytmów znajdujących najkrótszą ścieżkę. W artykule [18] autorzy wykorzystali algorytm Dijkstry do zarządzania ruchem przy użyciu automatycznie prowadzonych pojazdów. Autorzy przedstawili zarządzanie ruchem pojazdów w magazynie chcąc ograniczyć czas transportu. Algorytm Dijkstry wskazuje najkrótszą ścieżkę od jednego pojazdu do następnego. W przypadku, gdy na ścieżce znajdzie się przeszkoda - algorytm wylicza ścieżkę od nowa. Zastosowanie algorytmu Dijkstry do koordynacji ruchu na skrzyżowaniach jest zbyt złożone. Ponadto, unikanie kolizji jest wykonane poprzez ponowne wyliczanie najkrótszej ścieżki, zamiast unikania ich na bieżąco.

Autorzy artykułu [12] przedstawiają ulepszoną wersję algorytmu Dijkstry w celu znalezienia najkrótszej ścieżki. Do algorytmu Dijkstry wprowadzona została funkcja, dzięki której ignorowane są nieistotne wierzchołki w grafie. Dzięki tej modyfikacji algorytm jest 43%-76% szybszy. Rozwiązanie jest nadal zbyt złożone, by zastosować je do koordynacji ruchu na skrzyżowaniach. W rozwiązaniu nie ma także uwzględnionego unikania kolizji.

W artykule [1] przedstawione zostało planowanie ruchu przy użyciu automatycznie prowadzonych pojazdów wraz z unikaniem kolizji. W rozwiązaniu, każdy robot indywidualnie liczy najkrótszą dla siebie ścieżkę. Następnie, dane tras są wymieniane pomiędzy robotami, w celu ustalenia finalnego rozwiązania. Szukanie ścieżki jest wykonywane przy użyciu algorytmu Dijkstry. W celu unikania kolizji dla każdego pojazdu, liczona jest funkcja kary. Waga kary jest zwiększana tak długo, aż zostanie znaleziona bezkolizyjna ścieżka. Złożoność rozwiązania jest zbyt duża by zastosować je dla skrzyżowań. Wymienianie danych pomiędzy pojazdami oraz podany w rozwiązaniu sposób rozwiązywania kolizji, nie znajduje zastosowania w koorydynacji ruch na skrzyżowaniach.

W patencie [11] autorzy przedstawili centralnie zaplanowany, generalny system przydzielający trasy, w celu utrzymania optymalnego ruchu. System zawiera dużą ilość komputerów zamontowanych w pojazdach w celu komunikacji między nimi, poprzez sieć bezprzewodową. Do znajdowania najkrótszej ścieżki, autorzy wykorzystali algorytm Dijkstry. Ścieżki liczone są przy użyciu danych dostarczanych przez komputery zamontowane w pojazdach. Celem patentu jest znalezienie drogi do celu w najkrótszym czasie poprzez omijanie zakorkowanych dróg. Nie jest to dokładne zaplanowanie poruszania się pojazdu wraz z unikaniem kolizji.

Autorzy patentu [4] opisali system do kontroli ruchu i unikania kolizji na autostradach. Pod uwagę wzięte zostały zmiany pasów, unikanie kolizji oraz kontrolowanie trasy pojazdu. Unikanie kolizji zostało zapewnione poprzez komunikacje przez transmitery radiowe oraz radioodbiorniki zamontowane w każdym z pojazdów. Rozwiązanie znajduje zastosowanie na autostradach. Unikanie kolizji ma miejsce na równoległych pasach. Tego podejścia nie da się zastosować jeżeli chodzi o koordynacje ruchu na skrzyżowaniach.

W artykule [13] przedstawiony został system dynamicznego planowania ruchu do nawigacji samochodów przy użyciu 'virus genetic algorithms'. Autorzy zaproponowali algorytm genetyczny, który jest bardziej wydajny od algorytmu A*, czy algorytmu Dijkstry. W rozwiązaniu zastosowana jest strategia infekcji wirusowej, dzięki czemu omijane są korki na drogach. Rozwiązanie jest zaprojektowane do wskazania najlepszej trasy, nie jest to dokładne zaplanowanie ruchu samochodów wraz z wymijaniem kolizji.

2.4 Użycie algorytmu A* do planowania ruchu

Algorytm A* jest często stosowany do szukania najkrótszej ścieżki oraz planowania ruchu. W artykule [6] omówiona została optymalność tego algorytmu oraz funkcja heurystyki. A* jest algorytmem heurystycznym, który znajduje najkrótszą ścieżkę w grafie ważonym z dowolnego podanego wierzchołka do wierzchołka celu. Wierzchołkiem celu jest wierzchołek spełniający zadane warunki wygranej. W artykule przedstawione zostały poprzednie próby udowodnienia optymalności algorytmu. Autorzy przeprowadzili także własny dowód na optymalność algorytmu A*. Dowód polega na fakcie, iż kiedy funkcja heurystyki nigdy nie przeszacowuje, wtedy algorytm A* jest optymalny. Dla grafu, algorytm A* od zadanego

wierzchołka startowego tworzy ścieżkę, wybierając niezbadany sąsiedni wierzchołek. Wybiera go w sposób aby minimalizować funkcję:

$$f(x) = g(x) + h(x)$$

gdzie:

$g(x)$ to suma wag krawędzi od wierzchołka startowego do wierzchołka x, a $h(x)$ to przewidywana przez funkcję heurystyki suma wag z wierzchołka x do wierzchołka docelowego.

W artykule [7] autorzy przeprowadzili porównanie algorytmów do planowania drogi. Na początku omówiony został algorytm Dijkstry, aby później pokazać algorytmy od niego szybsze. Następnie, autorzy przedstawiają kolejki priorytetowe jako implementację algorytmu Dijkstry, które są w stanie poprawić jego złożoność z $O(n^2)$ do $O(nlogn)$. Używanie kolejek priorytetowych nie sprawdza się jednak na bardzo dużych zbiorach danych. Następną zaproponowaną modyfikacją algorytmu Dijkstry jest wyszukiwanie dwukierunkowe. Algorytm równolegle uruchamia algorytm Dijkstry, w stronę celu oraz w stronę startu. Kiedy wierzchołek jest odwiedzony z dwóch strony, ścieżka może być znaleziona. Kolejno opisany został Algorytm A* korzystający z funkcji heurystyki, opartej o odległość Euklidesową. Według autorów znajdywanie najkrótszej ścieżko jest niewiele przyspieszone. Mimo to czas wykonania algorytmu A* jest krótszy od czasu wykonania algorytmu Dijkstry.

Wykorzystanie modyfikacji algorytmu A* wraz z algorytmem Wavefront zostało przedstawione w artykule [19]. Autorzy przedstawili wersję algorytmu A*, która operuje na stanach. Autorzy pokazali przykład na dwuwymiarowej, dyskretnej przestrzeni kwadratowej o boku 7. Stanem początkowym jest rozłożenie czterech obiektów na rogach zadanej przestrzeni. Stanem końcowym natomiast, jest znalezienie się tych czterech obiektów po przeciwnych stronach przekątnych. Algortym Wavefront został zastosowany w celu otrzymania planów wielowariantowych dla obiektów. Zastosowanie algorytmu A* operującego na stanach wraz z prędkościami pojazdów wnosi duże narzuty obliczeniowe. Przeprowadzenie koordynacji ruchu dla dużych sieci skrzyżowań używając algorytmu A* wraz z Wavefront jest zbyt złożone.

W artykule [8] autorzy zaproponowali modyfikację algorytmu A*, która skupia się na

bezpieczeństwie w problemie nawigacji robotów. Modyfikacja polega na wzięciu pod uwagę rozmiaru robota oraz unikanie ostrych skrętów jeżeli chodzi o bezpieczeństwo. Do algorytmu A* przekazywany jest stan początkowy oraz cel jaki robot ma osiągnąć. Rozwiązanie przedstawione jest dla pojedynczego pojazdu, który ma za zadanie znaleźć najprostszą i bezpieczną ścieżkę wraz z unikaniem statycznych kolizji. Tego podejścia nie da się zastosować dla wielu pojazdów poruszających się po sieci skrzyżowań.

2.5 Modyfikacje A*

Autorzy artykułu [16] opisali modyfikację algorytmu A* do planowania ruchu. Rozwiązanie zostało zaprezentowanie dla robotów mobilnych. Autorzy stwierdzają fakt, że algorytm A* dla dużych danych potrzebuje dużo pamięci żeby śledzić dane, które są związane z aktualnym wierzchołkiem. Wspomniana została także modyfikacja algorytmu A*, która polega na równoczesnym przeszukiwaniu grafu w dwóch kierunkach. Kolejną modyfikacją jest IDA*, która nie używa pamięci do przechowywania wcześniej odwiedzonych wierzchołków. Minusem tej modyfikacji jest fakt, iż IDA* odwiedza niektóre wierzchołki wiele razy. Na końcu, autorzy zaproponowali swoją modyfikację - Fast A*. W modyfikacje zamiast list, użyte zostały drzewa binarne w celu przechowywania danych. W wyniku pracy zmodyfikowany algorytm Fast A* dawał lepsze wyniki od wersji klasycznej. Przedstawione modyfikacje są oparte o działanie na grafie, polegające na znalezieniu najkrótszej ścieżki z punktu startowego do punktu końcowego. Modyfikacje nie są wstanie wskazać najkrótszej ścieżki dla stanu początkowego i końcowego, którym może być na przykład położenie wszystkich samochodów na skrzyżowaniach.

W artykule [17] autorzy zaproponowali genetyczny algorytm oparty na zmodyfikowanym algorytmie A* w celu optymalizacji znajdowania ścieżek dla wielu obiektów. Modyfikacja algorytmu A* polega na znalezieniu ścieżki suboptymalnej poprzez dodanie do algorytmu funkcji prawdopodobieństwa. Uniknięto więc szukania najkrótszej ścieżki, aby zwiększyć optymalność algorytmu. To rozwiązanie następnie jest przekazywane do algorytmu genetycznego. W rozwiązaniu uwzględnione zostało także unikanie kolizji. Autorzy opisali podejście optymalizacji szukania najkrótszej ścieżki dla pojedynczego pojazdu z unikaniem kolizji ze statycznymi elementami. Tego rozwiązania nie da się zastosować w sieci skrzyżowań, na któ-

rej znajduje się wiele aut.

W celu planowania ruchu na skrzyżowaniach zostało przedstawione sporo rozwiązań. Większość z nich opiera się o stosowanie świateł. Powyżej zostały opisane optymalizacje świateł w celu przyspieszania ruchu drogowego. Rozwiązania te gwarantują bezpieczeństwo poprzez czekanie na świetle pomarańczowym, co powoduje kilkusekundowe straty na każdym skrzyżowaniu. Takich opóźnień nie ma stosując planowanie algorytmem. Przedstawione rozwiązania przy użyciu algorytmów planowania często nie brały pod uwagę unikania kolizji lub planowały jedynie ruch dla pojedynczego pojazdu. Tych podejść nie da się zastosować w celu zaplanowania ruchu na sieci skrzyżowań dla kilkunastu pojazdów.

Teza

3.1 Wprowadzenie

W pracy rozwiązywany jest problem planowania ruchu na skrzyżowaniach dróg wielopasmowych. Rozwiązanie w porównaniu do wcześniej opisanych podejść, wyróżnia się użyciem algorytmu A* do dokładnego zaplanowania ruchu dla każdego z pojazdów wraz z unikaniem kolizji. Większość przedstawionych rozwiązań opiera się o światła drogowe, co wiąże się z opóźnieniami spowodowanymi oczekiwaniem na świetle pomarańczowym, a także opóźnieniami spowodowanymi na oczekiwaniu na świetle czerwonym, kiedy na drodze poprzecznej nie ma żadnych pojazdów. Implementacja algorytmu A* jest generyczna i opiera się na stanach. Algorytm przyjmuje obiekt klasy implementującej metodę zwracającą stany sąsiednie dla stanu aktualnego. Do algorytmu przekazywane są także warunki końcowe. Algorytm przeszukuje stany korzystając z funkcji heurystyki, aż osiągnięte zostaną warunki końcowe - nie przeszukiwane są wszystkie możliwe stany, algorytm kończy obliczenia, kiedy znajdzie stany spełniające określone warunki.

3.2 Planowanie ruchu przy użyciu modyfikacji A*

Można postawić tezę: zastosowanie metody planującej ruch drogowy przy użyciu algorytmu A* pozwala na zaplanowanie bezkolizyjnego ruchu drogowego dla pojazdów na skrzyżowaniu, co pozwala na osiągnięcie lepszej przepustowości niż w przypadku świateł sekwencyjnych wraz z optymalizacjami. Przepustowość rozumiemy jako ilość samochodów opuszczających skrzyżowanie w jednostce czasu. Rozwiązanie jest uruchamiane na jednym

rdzeniu procesora. Korzystając z większej ilości rdzeniów, metoda może policzyć plany dla wielu wariantów. Plany wielowariantowe znajdują zastosowanie w przypadku pomyłki pojazdów, polegającej na niezastosowanie się do wyliczonego planu.

Rozwiązanie przyjmuje położenie samochodów na sieci skrzyżowań oraz informacje na temat dróg jako dane wejściowe. Wynikiem jest dokładny plan poruszania się dla każdego z pojazdów. W planie znajdują się informacje o tym z jaką prędkością pojazd powinien się poruszyć, kiedy zwolnić lub przyspieszyć, aby omijać kolizje.

W metodzie, do planowania wykorzystany został algorytm A*. Jest on oparty o działanie na stanach, gdzie pojedynczym stanem jest rozmieszczenie pojazdów na sieci skrzyżowań wraz z ich prędkościami. Stanem sąsiednim dla aktualnego stanu, jest zmiana pozycji pojazdów na sieci skrzyżowań wraz z ich prędkościami. Implementacja algorytmu A* jest generyczna - algorytm przyjmuje dowolny obiekt klasy impementujący metodę 'neighbours', która zwraca stany sąsiednie dla aktualnego stanu. Algorytm działa w sposób dynamiczny. W przypadku działania algorytmu A* na grafie, z góry znane są krawędzie oraz wierzchołki. W opracowanym rozwiązaniu algorytm dynamicznie tworzy sąsiednie stany.

Do algorytmu przekazywana jest funkcja heurystyki, a także warunek końcowy. Dla koordynacji ruchu, warunkiem końcowym jest opuszczenie przez wszystkie pojazdy skrzyżowania. Funkcja heurystyki operuje na danych stanu - każdemu stanowi przypisuje wartość. W przedstawionym rozwiązaniu funkcja heurystyki zwraca najmniejsze wartości dla stanów, w których samochody maksymalnie przyspieszają.

Do systemu wprowadza się także następujące dane:

- maksymalną prędkość jaką pojazd może osiągnąć
- wartośći przyspieszeń pojazdu
- parametr bezpieczeństwa, czyli odległość utrzymywana pomiędzy pojazdami

System posiadając stan początkowy i powyższe dane, tworzy wszystkie możliwe stany pochodne. Następnie korzystając z funkcji heurystyki algorytm wybiera stany, które prowadzą do najszybszego opuszczenia przez samochody skrzyżowania.

Rozwiązanie bierze pod uwagę unikanie kolizji pojazdów na skrzyżowaniu oraz pojazdów poruszających się tym samym pasem. Unikanie kolizji polega na usuwaniu stanów sąsiednich, które powodują kolizję - metoda 'neighbours' zwraca jedynie stany bezkolizyjne.

Do systemu można przekazać każdy rodzaj skrzyżowania jako listę dróg wraz z ich rozmiarami oraz informację o przecięciach z innymi drogami. Rozwiązanie oparte jest o przestrzeń dyskretną. Drogi mają konkretny rozmiar. Do systemu przekazuje się także listę samochodów, która zawiera informacje o położeniu samochodów na drogach oraz informację o ich prędkościach. System opiera się o działanie na krokach czasowych. Prędkość samochodów wyrażana jest w liczbie odcinków drogi na jeden krok czasowy.

W celu wizualizcji stworzny został także moduł graficzny, który jest w stanie zwizualizować każde podane skrzyżowanie. Przedstawia on także w formie zdjęć, zaplanowany ruch samochodów. Każde zdjęcie odpowiada następnemu krokowi czasowemu. Na ostatnim zdjęciu skrzyżowanie jest opuszczone przez wszystkie samochody.

Wadą metody jest złożoność obliczeniowa, która jest zależna od ilosci samochodów na skrzyżowaniach. Czas trwania programu zależy wykładniczo od ilości pojazdów na skrzyżowaniu. Dla 14 pojazdów znajdujący się na skrzyżowaniu 8 dróg obliczenia trwają do kilku godzin, co jest znaczącym limitem dla rozwiązania.

Implementacja

4.1 Opis modelu danych do reprezentacji skrzyżowań

W opracowanym rozwiązaniu drogi reprezentowane są w sposób dyskretny. Każda droga zaczyna się w pozycji 1 i rośnie wraz z jej rozmiarem. Droga jest opisywana przez:

- unikatowy numer drogi
- rozmiar drogi
- informacja o przecięciach z innymi drogami
- kierunek w którym samochód się porusza na drodze (z północy na południe, czy z południa na północ lub z zachodu na wschód czy ze wschodu na zachód)

Samochody na drogach są opisane przez:

- unikatowy numer samochodu
- unikatowy numer drogi, na której pojazd się znajduje
- numer pozycji na drodze, na której samochód się znajduje
- prędkość początkowa
- numer pozycji docelowej na drodze - punkt za ostatnim skrzyżowaniem

Samochody poruszają się po drogach w krokach czasowych. Prędkość samochodu wyrażana jest w liczbie odcinków drogi na jeden krok czasowy.

W Systemie można wybrać maksymalne przyspieszenia ujemne oraz dodatnie samochodów z następujących możliwości {-2, -1, 0, 1, 2}. Oznacza to, że w następnym stanie pojazd może przyspieszyć o 1 lub 2 odcinki drogi na jeden krok czasowy. Dla wartości 0 pojazd utrzymuje swoją prędkość. Dla wartości ujemnych pojazd zwalnia o 1 lub 2 odcinki drogi na jeden kroku czasowym. Ilość możliwych dla samochodu przyspieszeń znacznie wpływa na złożoność czasową algorytmu, z tego względu, że wraz ze wzrostem ilośći możliwych przyspieszeń rośnie ilość możliwość stanów sąsiednich dla samochodu, przez które algorytm musi przejść.

Do modelu przekazać należy także maksymalną prędkość, którą wszystkie samochody mogą osiągnąć oraz parametr prędkości, czyli liczba pozyji będąca dystansem trzymanym między pojazdami.

4.2 Zmodyfikowany algorytm A*

Zaprezentowany w tej pracy zmodyfikowanym algorytmu A* jest oparty na stanach, gdzie pojedyńczym stanem jest rozłożenie samochodów na drogach wraz z ich prędkościami. Algorytm zaczynając od zadanego stanu początkowego analizuje możliwe stany sąsiednie i wybiera najszybszą ścieżkę kierując się funkcją heurystyki. Ścieżka prowadzi do określonego celu. W przypadku opracowanego rozwiązania, celem jest przekroczenie przez wszystkie samochody ostatniego skrzyżowania na drogach, na których się znajdują.

Zrealizowany Algorytm A* jest generyczny. Do algorytmu przekazujemy klasę reprezentującą dowolny stan. Klasa musi implementować metodę 'neighbours', która zwraca stany sąsiednie dla danej instancji stanu. Do algorytmu przekazywana jest także funkcja heurystyki, zależna od danych danego stanu.

Modyfikacja róźni się od innych tym, że stany sąsiednie są liczone dynamicznie. Algorytm nie ma z góry podnaego celu. Dla przykładu w przeszukiwaniu ścieżki w grafie celem jest podany wierzchołek końcowy. Algorytm przyjmuje funkcje wygranej, która określa warunki końca algorytmu. W przedstawionej pracy warunkiem końca jest przekroczenie przez wszystkie pojazdy skrzyżowań.

4.3 Stan w opracowanym rozwiązaniu dla algorytmu A*

Stan w modyfikacji algorytmu A* przedstawionej w tej pracy opisuje stan wszystkich samochodów na skrzyżowaniach wraz z ich prędkościami. Zgodnie z wcześniej opisaną generycznością wierzchołek jest obiektem klasy, który implementuje metodę 'neighbours'. Metoda generuje wszystkie stany sąsiednie dla aktualnego stanu. Dla przykładu, dla ustawień przyspieszeń {-2, -1, 0, 1, 2}, dla każdego z aut na skrzyżowaniu generowane są stany z jego prędkością dodając wartości przyspieszeń. Pomijane są stany, w których prędkość samochodu byłaby ujemna. Przykładowo jest pojazd z prędkością 1 pozycji na krok czasowy znajdujący się na 1 pozycji na drodze numer 1. Dla pojazdu rozpatrywane są następujące stany:

1. Dodając przyspieszenie o wartości 0: 2 pozycja na drodze numer jeden z prędkością 1
2. Dodając przyspieszenie o wartości 1: 3 pozycja na drodze numer jeden z prędkością 2
3. Dodając przyspieszenie o wartości 2: 4 pozycja na drodze numer jeden z prędkością 3
4. Dodając przyspieszenie o wartości -1: 1 pozycja na drodze numer jeden z prędkością 0
5. Dodając przyspieszenie o wartości -2: Stan jest eliminowany, ponieważ prędkość byłaby ujemna

Dodatkowo w przypadku ustawienia maksymalnej prędkości równej 2 pozycje na jeden krok czasowy - możliwość nr 3 także zostałaby wyeliminowana.

W metodzie usuwane są także stany powodujące kolizje, co będzie opisane w następnym rozdziale.

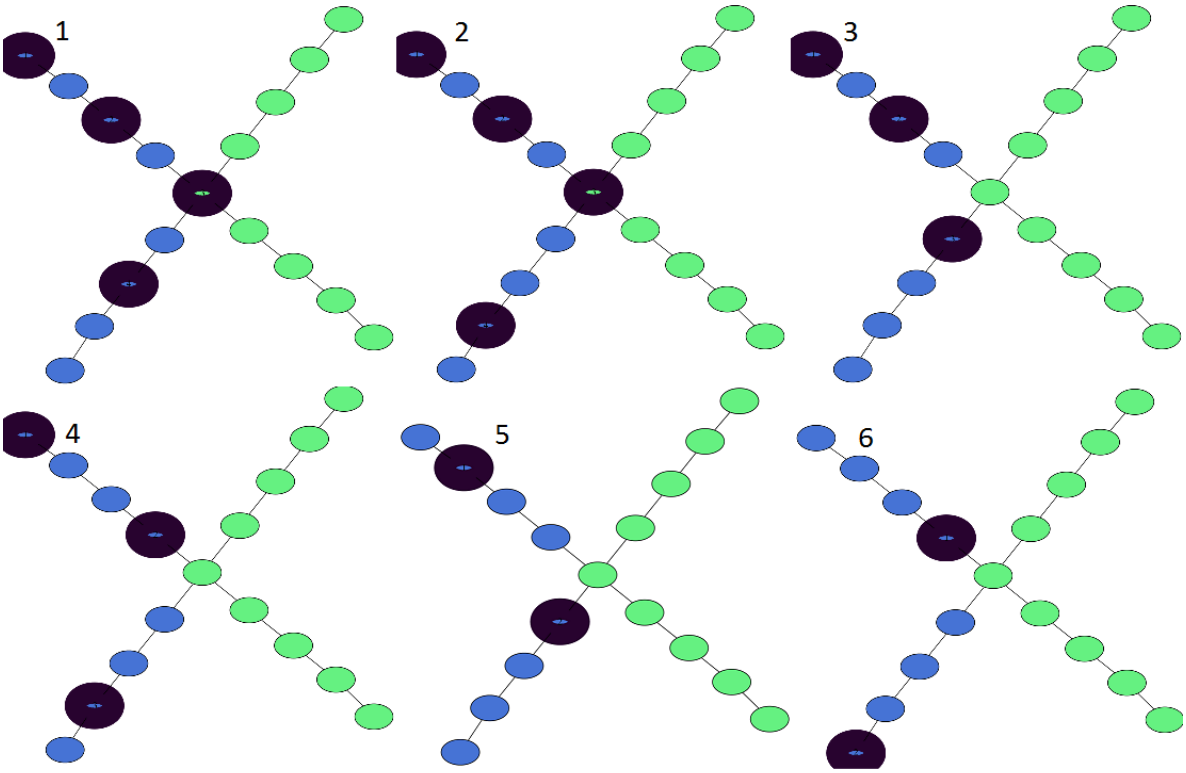
4.4 Unikanie kolizji

Koniecznym elementem w koordynacji ruchu na skrzyżowaniach wielu pojazdów jest unikanie kolizji. W opracowanym rozwiązaniu nikanie kolizji zostało podzielone na dwa etapy:

1. Unikanie kolizji aut poruszających się po tym samym pasie

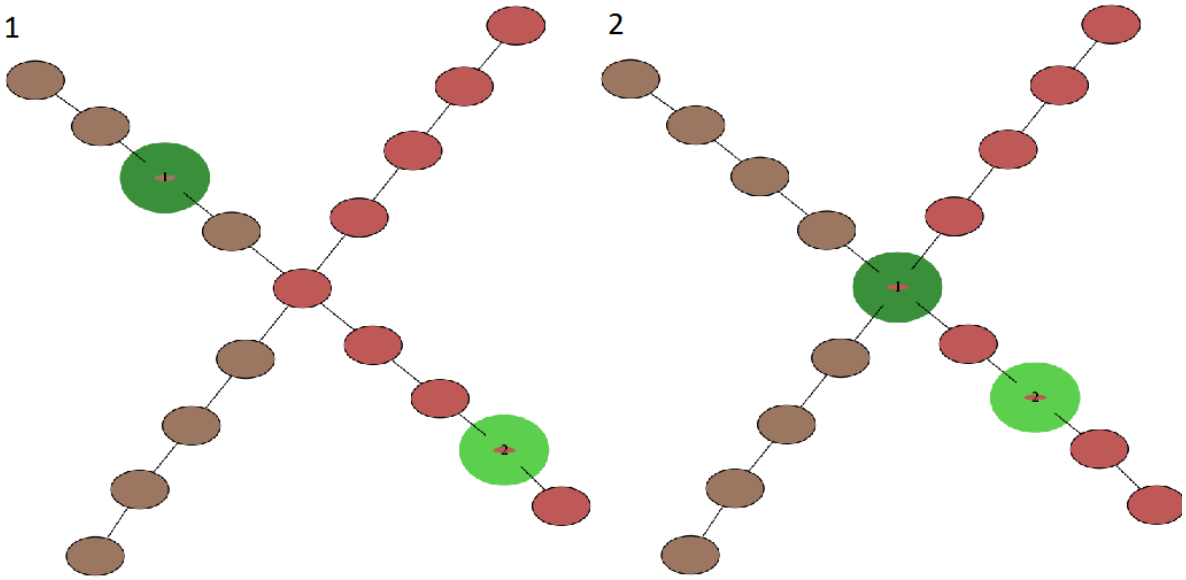
2. Unikanie kolizji na skrzyżowaniach

Implementacja unikania kolizji pojazdów na tym samym pasie polega na policzeniu obszarów dla samochodów znajdujących się na tym samym pasie, który będą one obejmowały w jednym kroku czasowym wraz z uwzględnieniem parametru bezpieczeństwa. Jeżeli obszary dwóch samochdów na jednym pasie pokrywają się - taki stan jest usuwany i wówczas metoda 'neighbours' nie zwróci go jako stanu sąsiedniego. Na rysunku 4.1 przedstawiony jest przykład unikania kolizji dla samochodów poruszających się jednym pasem. Przykład przedstawiony jest z parametrem bezpieczeństwa równym jeden. Samochody, mając przed sobą inny pojazd, czekają aż będzie dla nich miejsce. Stan pierwszy jest stanem startowy. Kolejno w stanie drugim, pojazd ostatni znajdujący się najdalej od skrzyżowania rusza i zmienia swoją pozycję o jeden do przodu. Reszta pojazdów stoi i czeka na miejsce. Następnie w stanie trzecim kolejny pojazd widząc, że ma miejsce porusza się o jedną pozycję do przodu. W stanie następnym na miejsce czeka jedynie samochód pierwszy - reszta zmienia pozycję. W kroku piątym i szóstym pojazdy mając miejsce, swobodnie opuszczają skrzyżowanie.



Rysunek 4.1: Unikani kolizji na pasie

Unikanie kolizji na skrzyżowaniach polega na eliminacji stanów, w których conajmniej dwa auta przekroczyły w jednym kroku czasowym to samo skrzyżowanie. Metoda 'neighbours' także nie zwróci takich stanów. Na rysunku 4.2 przedstawiony jest przykład unikania kolizji dla dwóch samochodów na prostym skrzyżowaniu. Samochód z kolorem ciemnozielonym przyspieszył o dwie jedostki i zmienił pozycję o dwie jednostki do przodu. Samochód z kolorem jasnozielonym przyspieszył natomiast o jedną jednostkę i zmienił pozycję o jedną jednostkę do przodu w celu uniknięcia kolizji. W algorytmie usunięty został stan, w którym oba pojazdy przyspieszyły o dwie jednostki. Został natomiast wybrany stan gdzie jeden z samochodów przyspieszył o dwie jednostki, a drugi o jedną jednostką, ponieważ prowadzi to, do najszybszego opuszczenia skrzyżowania przez pojazdy.



Rysunek 4.2: Unikanie kolizji na skrzyżowaniu

4.5 Format danych wejściowych

Zaprezentowane rozwiązanie potrzebuje dane na temat skrzyżowań, samochodów oraz ograniczeń parametrów samochodów. Skrzyżowania reprezentowane są w pliku CSV posiadającym atrybuty: numer drogi, rozmiar, przecięcia, kierunek. Przecięcie reprezentowane są przez dwie liczby. Na przykład 5 7 - oznacza, że droga przecina drogę o numerze 5 w pozycji 7. Kolejne przecięcia oddzielane są średnikami. Stany samochodów zapisane są w pliku CSV posiadającym atrybuty: numer samochodu, numer drogi samochodu, pozycję na drodze, pozycję końcową. Ograniczenia przechowywane są w osobnym pliku CSV posiadającym atrybuty: maksymalna prędkość, parametr bezpieczeństwa, wartości przyspieszeń.

4.6 Funkcja heurystyki

Funkcja heurystyki dla algorytmu A* jest to suma kroków czasowych, po których wszystkie auta przekroczą ostatnie skrzyżowanie - czyli osiągną cel. Funkcja liczona jest, przy założeniu, że wszystkie samochody maksymalnie przyspieszają.

W opracowanym rozwiązaniu każdy samochód ma wyznaczoną pozycję, na której musi się znaleźć lub ją przekroczyć (jest to pozycja za ostatnim skrzyżowaniem). Poniżej przedstawiony jest przykład liczonej funkcji heurystyki dla dwóch stanów opisujących położenia samochodów na skrzyżowaniu.

Numer pojazdu	Numer drogi	Pozycja na drodze	Prędkość	Pozycja końcowa
1	1	1	1	6
2	2	1	1	6
3	3	1	1	6
4	4	1	1	6

Tabela 4.1: Stan 1

Numer pojazdu	Numer drogi	Pozycja na drodze	Prędkość	Pozycja końcowa
1	1	1	0	6
2	2	1	1	6
3	3	1	0	6
4	4	1	1	6

Tabela 4.2: Stan 2

Dla stanu opisanego w tabeli 4.1 funkcja heurystyki jest liczona następująco:

Dla pojazdów 1-4:

1 krok czasowy:

prędkość = (prędkość) 1 + (maksymalne przyspieszenie) 1 = 2
pozycja = (pozycja) 1 + (prędkość) 2 = 3

Pojazd znajduje się na pozycji 3, która jest przed pozycją końcową 6 - liczony jest kolejny krok czasowy.

2 krok czasowy:

prędkość = (prędkość) 2 + (maksymalne przyspieszenie) 1 = 3
pozycja = (pozycja) 3 + (prędkość) 3 = 6

Pojazd znajduje się na pozycji 6, która jest równa pozycji końcowej.

Razem 2 kroki czasowe dla jednego pojazdu. Dla wszystkich czterech pojazdów - 8 kroków czasowych.

Dla stanu opisanego w tabeli 4.2 funkcja heurystyki jest liczona następująco:

Dla pojazdów 1 i 3

1 krok czasowy:

prędkość = (prędkość) 0 + (maksymalne przyspieszenie) 1 = 1
pozycja = (pozycja) 1 + (prędkość) 1 = 2

Pojazd znajduje się na pozycji 2, która jest przed pozycją końcową 6 - liczony jest kolejny krok czasowy.

2 krok czasowy:

prędkość = (prędkość) 1 + (maksymalne przyspieszenie) 1 = 2
pozycja = (pozycja) 2 + (prędkość) 2 = 4

Pojazd znajduje się na pozycji 4, która jest przed pozycją końcową 6 - liczony jest kolejny krok czasowy.

3 krok czasowy:

prędkość = (prędkość) 2 + (maksymalne przyspieszenie) 1 = 3
pozycja = (pozycja) 4 + (prędkość) 3 = 7

Pojazd znajduje się na pozycji 7, która jest za pozycją końcową 6.

Razem 3 kroki czasowe dla jednego pojazdu. Dla pojazdów 1 i 3 - 6 kroków czasowych.

Dla pojazdów 2 i 4:

1 krok czasowy:

prędkość = (prędkość) 1 + (maksymalne przyspieszenie) 1 = 2
pozycja = (pozycja) 1 + (prędkość) 2 = 3

Pojazd znajduje się na pozycji 3, która jest przed pozycją końcową 6 - liczony jest kolejny krok czasowy.

2 krok czasowy:

prędkość = (prędkość) 2 + (maksymalne przyspieszenie) 1 = 3
pozycja = (pozycja) 3 + (prędkość) 3 = 6

Pojazd znajduje się na pozycji 6, która jest równa pozycji końcowej.

Razem 2 kroki czasowe dla jednego pojazdu. Dla pojazdów 2 i 4 - 4 kroki czasowe.

Razem dla wszystkich pojazdów - 10 kroków czasowych.

Podsumowując, dla stanu opisanego w tabeli 4.1 funkcja heurystyki zwróci wartość 8 (osiem kroków czasowych), a dla stanu opisanego w tabeli 4.2 funkcja heurystyki zwróci wartość 10 (dziesięć kroków czasowych). Algorytm wybierze wartość minimalną - czyli wybierze stan opisany w tabeli 4.1.

4.7 Złożoność rozwiązania

W przedstawionym modelu stan jednego samochodu to (p, v), gdzie p to pozycyja na drodzę oraz v to prędkość pojazdu w odcinkach drogi na krok czasowy. Pozostałe parametry, czyli numer drogi na której znajduje się samochód i pozycja końcowa czy przyspieszenie nie zmieniają się. Dla wartości przyspieszeń ze zbioru {-1, 0, 1} dla samochodu powstają następujące stany:

1. (p + v, v) - dla przyspieszenia 0
2. (p + v + 1, v + 1) - dla przyspieszenia 1
3. (p + v - 1, v - 1) - dla przyspieszenia 1

Dla jednego pojazdu z każdym krokiem czasowym powstają 3 stany. Mając na skrzyżowaniu 12 pojazdów otrzymujemy daje 531441 możliwości w jednym kroku czasowym. Algorytm nie musi przechodzić przez wszystie możliwości, jeżeli wcześniej znajdzie rozwiązanie.

Największym obciążeniem algorytmu jest unikanie kolizji. Zajmuje ono około 60 % czasu znajdowania rozwiązania. Jest to spowodowane tym, że dla każdego stanu, którym jest położenie wszystkich pojazdów na skrzyżowaniu, algorytm przeprowadza dwie weryfikacje w każdym kroku czasowym:

1. Czy istnieją dwa pojazdy, które w tym samym czasie przekroczyły skrzyżowanie
2. Czy odcinki zajęte przez dwa pojazdy na jednym pasie się przecinają

Powyższe stany eliminowane są z rozwiązania co zapewnia unikanie kolizji.

Dla zbioru przyspieszeń {-2, -1, 0, 1, 2} dla każdego pojazdu powstaje 5 stanów. Mając na skrzyżowaniu 12 pojazdów mamy 244140625 możliwości w jednym kroku czasowym. Możliwości przypadające na krok czasowy są silnie związane z czasem wykonania algorytmu. Wraz ze wzrostem pojazdów na skrzyżowaniach, a co za tym idzie dodatkowych możliwośći, czas wykonania rośnie ekspotencjalnie.

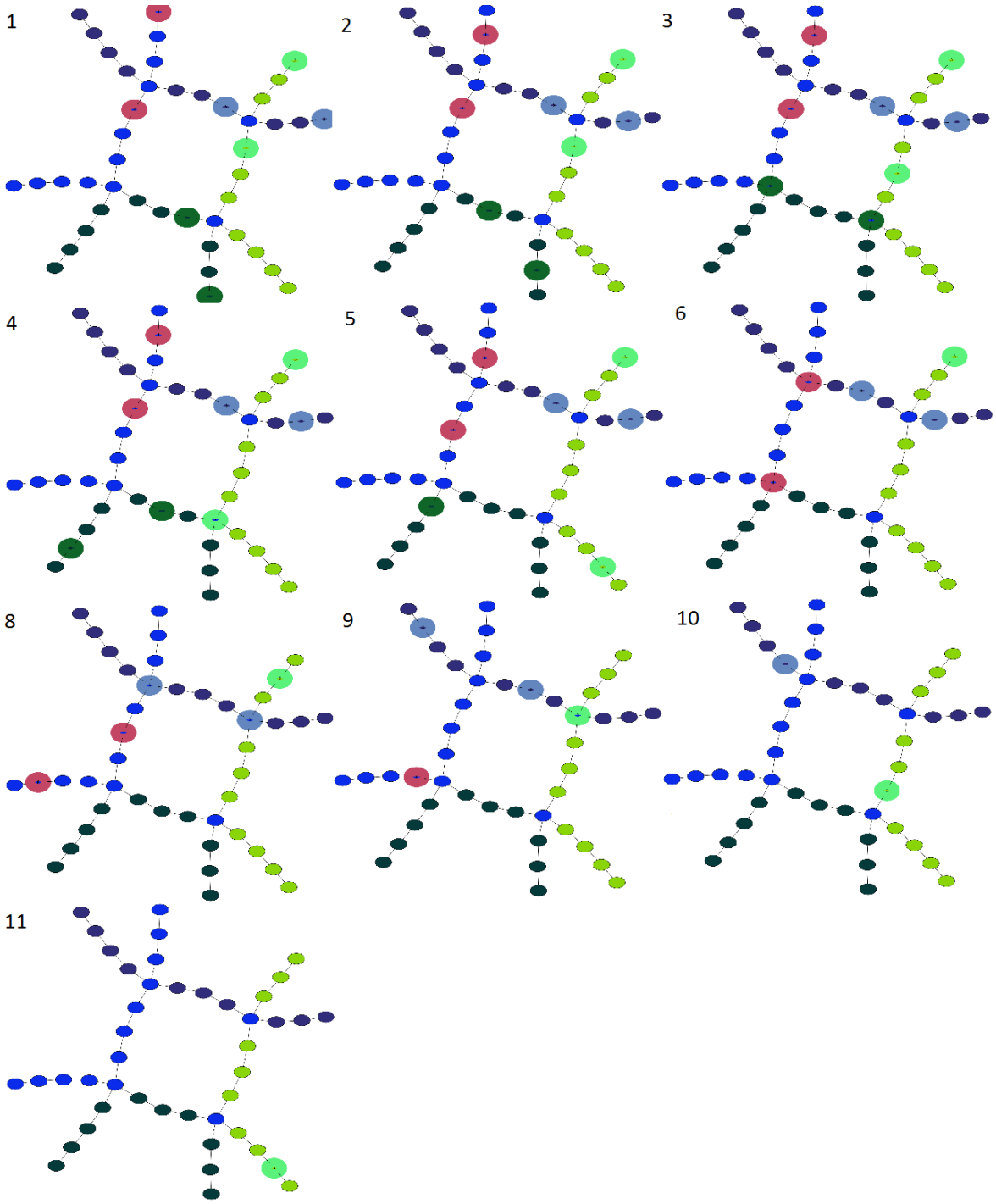
4.8 Reprezentacja graficzna wyników

W celu graficznej reprezentacji wyników zaimplementowany został moduł graficzny. Działanie modułu graficznego oparte jest na rysowaniu grafów z wykorzystaniem biblioteki Graphviz oraz jej implementacji w języku Ruby. Skrzyżowanie jest reprezentowane za pomocą grafu. Wierzchołkami grafu są poszczególne odcinki dróg. Krawędziami grafu są połączenia pomiędzy odcinkami. Samochody poruszające się po tej samej drodze są zaznaczone grubszym obwodem oraz mają ten sam kolor. Samochody można rozróżnić poprzez unikatowy numer na nich napisany. Kolejne kroki czasowe reprezentowane są poprzez następne grafy z kolejnymi ułożeniami samochodów. Efektem końcowym jest stworzony GIF z następujących po sobie grafów przy użyciu biblioteki 'rmagick' dla języka Ruby.

Moduł graficzny wczytuje wszystkie dane wejściowe: pozycję samochodów, dane skrzyżowania oraz ograniczenia. Kolejno na wejście otrzymuje plik zawierający JSON, który reprezentuje wszystkie stany samochodów na skrzyżowaniach od momentu startu, aż do opuszczenia skrzyżowań przez wszystkie samochody. Plik powstaje w wyniku wykonania zaprezentowanego algorytmu A*. Moduł wczytując dane początkowe tworzy graf bazowy. Graf bazowy przedstawia skrzyżowanie wraz z pojazdami. Kolory dla dróg oraz samochodów są losowane oraz przetrzymywane w zbiorze tak, aby żaden z kolorów się nie powtórzył. Następnie moduł, posiadający kolejne stany w kolejnych krokach czasowych, korzysta z grafu bazowego i zmienia pozycję samochodów. Po każdym kroku czasowym zapisywane jest zdjęcie grafu z numerem kroku czasowego.

W module znajduje się także dodatkowe sprawdzanie kolizji. Jeżeli kolizja wystąpi, w danym kroku czasowym na zdjęciu dodany zostanie napis "Collision". Dzięki temu można zobaczyć prawdopodobieństwo kolizji w przypadku, gdy jeden z pojazdów nie zastosuje się do planu.

Na rysunku 4.3 zaprezentowane jest działanie modułu graficznego dla skrzyżowania czterech dróg i ośmiu pojazdów. Na 11 elemencie rysunku widać, że wszystkie samochody opuściły wszystkie skrzyżowania po 11 krokach czasowych.



Rysunek 4.3: Wynik działania modułu graficznego

Wyniki

5.1 Wstęp

W celu zbadania działania systemu przeprowadzone zostały następujące pomiary:

- pomiar czasu wykonania programu w zależności od liczby samochodów na skrzyżowaniu
- pomiar ilości korków czasowych po których wszystkie samochody opuszczą skrzyżowania w zależnośći od liczby samochodów na skrzżowaniu

Pomiary zostały wykonane na dwóch sieciach dróg:

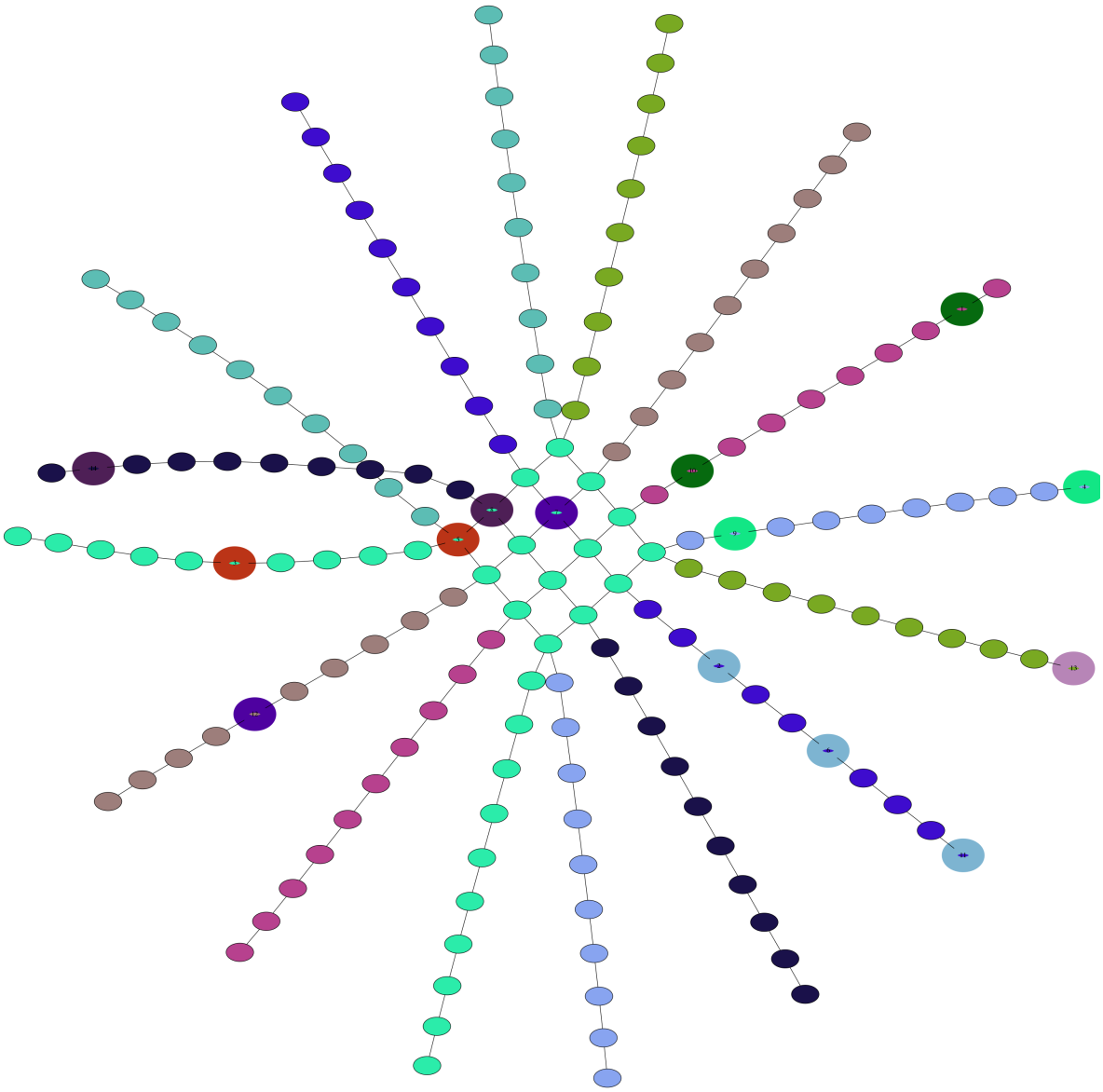
- skrzyżowanie ze sobą ośmiu dróg
- skrzyżowanie ze sobą czterech dróg

Drogi na obydwu sieciach mają rozmiar 24.

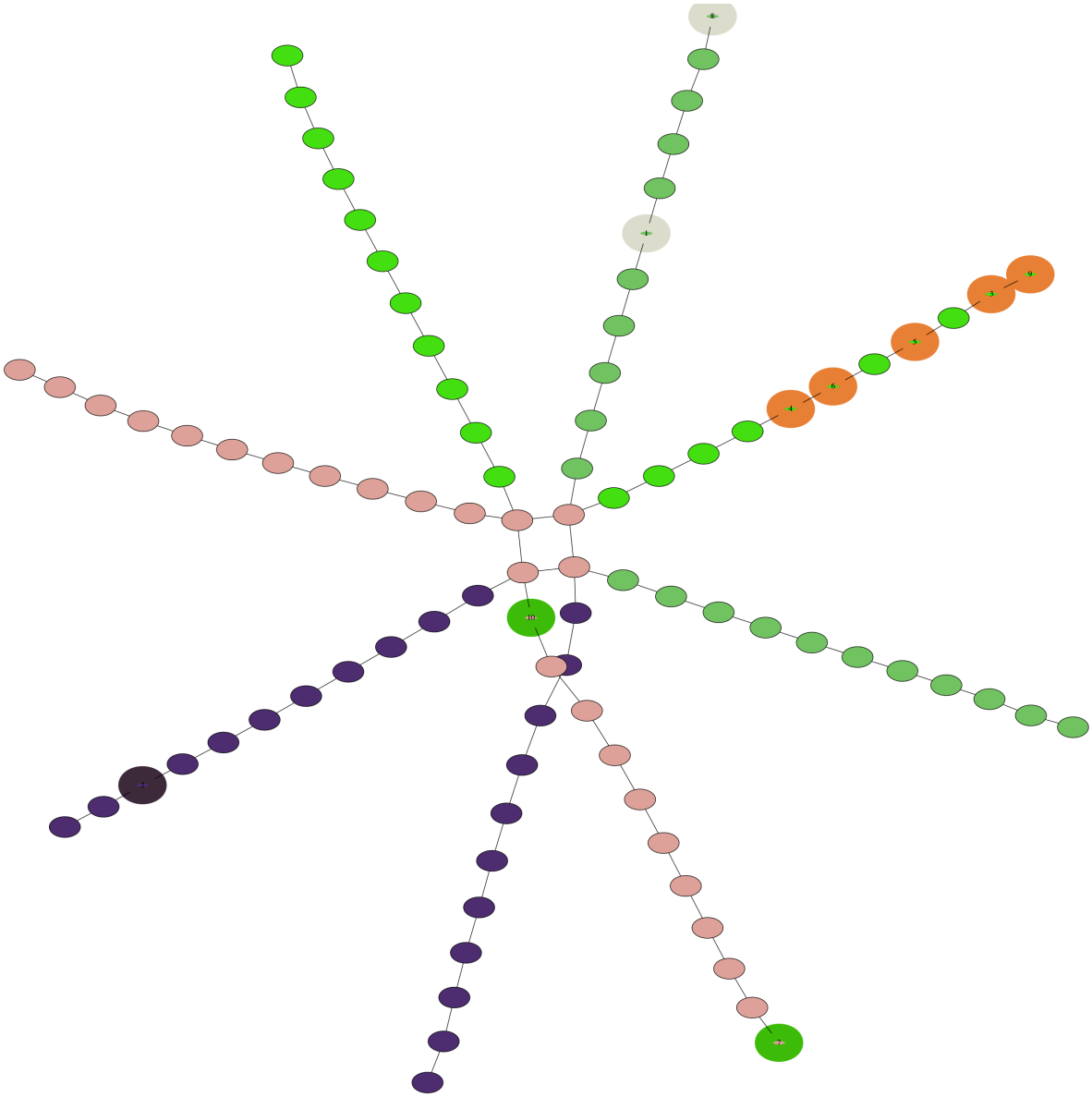
Reprezentacja graficzna skrzyżowań jest przedstawiona na rysunkach 5.4 oraz 5.5 z przykładowymi rozmieszczeniami pojazdów.

Pomiary zostały przeprowadzone dla następującej liczby samochodów $\{2, 4, 6, 8, 10, 12\}$.

Dla każdej liczby samochodów wygenerowane losowo zostało 30 położeń pojazdów przy założeniu, że żaden z pojazdów nie startuje za ostatnim skrzyżowaniem.



Rysunek 5.4: Skrzyżowanie ośmiu dróg

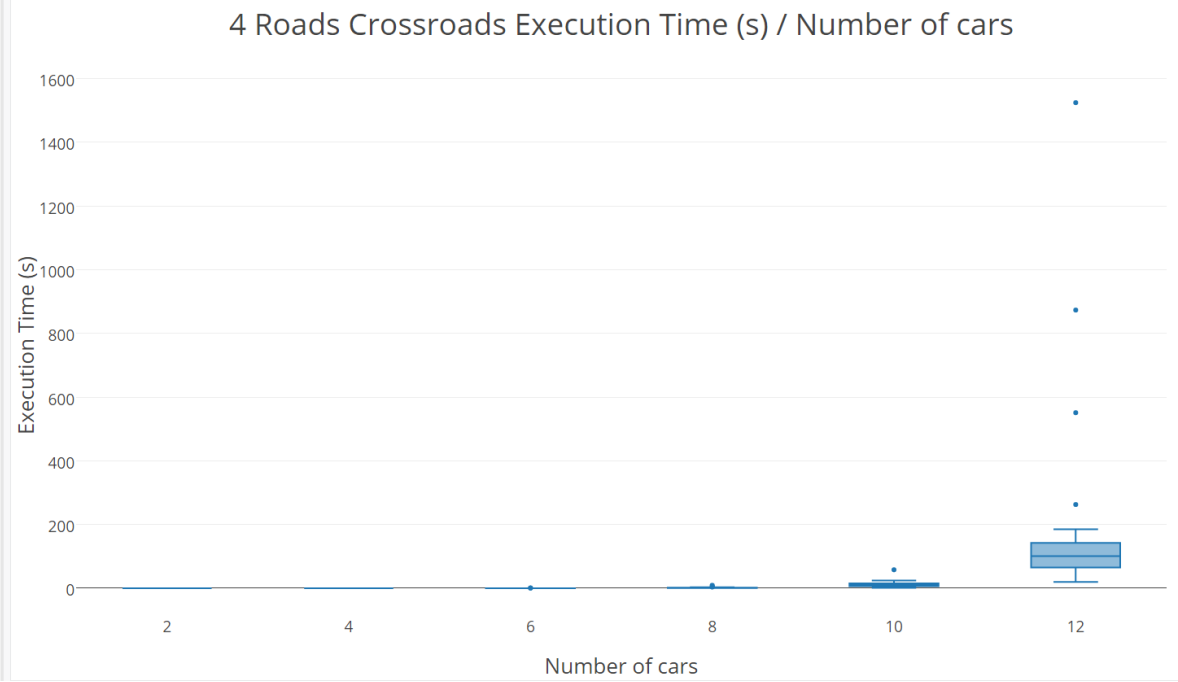


Rysunek 5.5: Skrzyżowanie czterech dróg

5.2 Wyniki pomiarów dla skrzyżowania czterech dróg

Zgodnie z wcześniej opisanymi założeniami dla skrzyżowania czterech dróg program został uruchomiony 30 razy dla samochodów ze zbioru {2, 4, 6, 8, 10, 12} co razem daje 180 uruchomień.

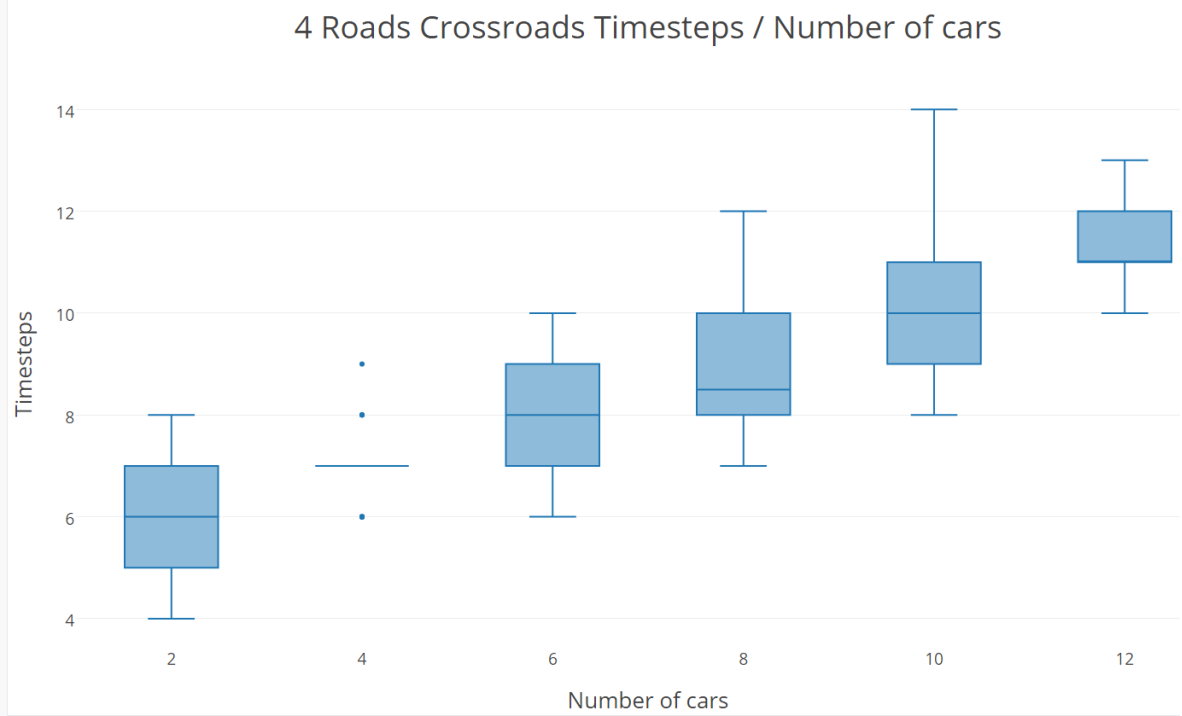
Wykresy przedstawiające wyniki na skrzyżowaniu czterech dróg zaprezentowane są na rysunkach 5.6 oraz 5.7



Rysunek 5.6: Wykres zależności czasu wykonania programu od liczby pojazdów

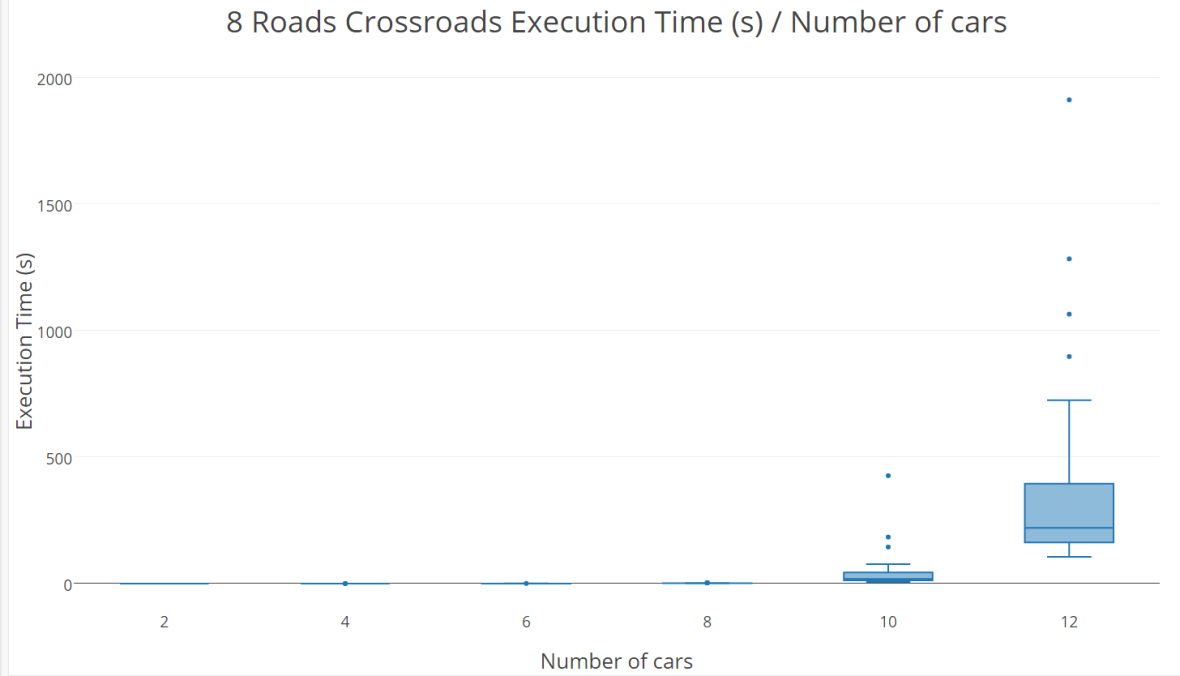
5.3 Wyniki pomiarów dla skrzyżowania ośmiu dróg

Dla skrzyżowania ośmiu dróg też zostało przeprowadzone 180 uruchomień.

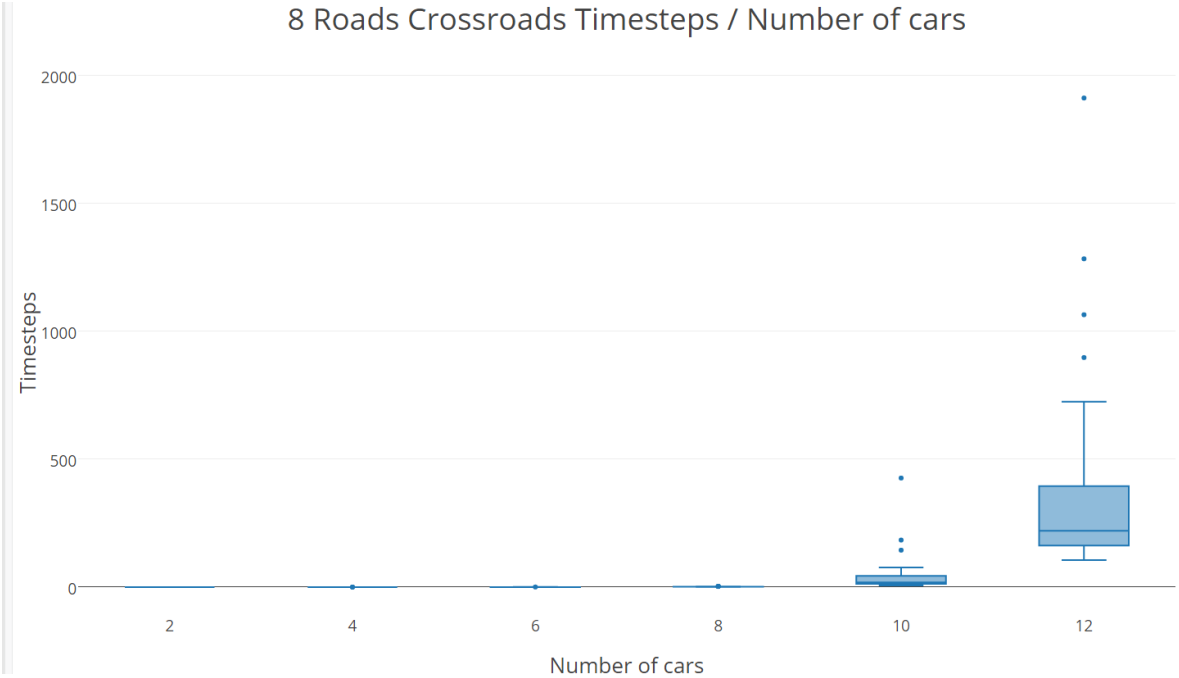


Rysunek 5.7: Wykres zależności kroków czasowych od liczby pojazdów

Wykresy przedstawiające wyniki na skrzyżowaniu czterech dróg zaprezentowane są na rysunkach 5.6 oraz 5.7



Rysunek 5.8: Wykres zależności czasu wykonania programu od liczby pojazdów



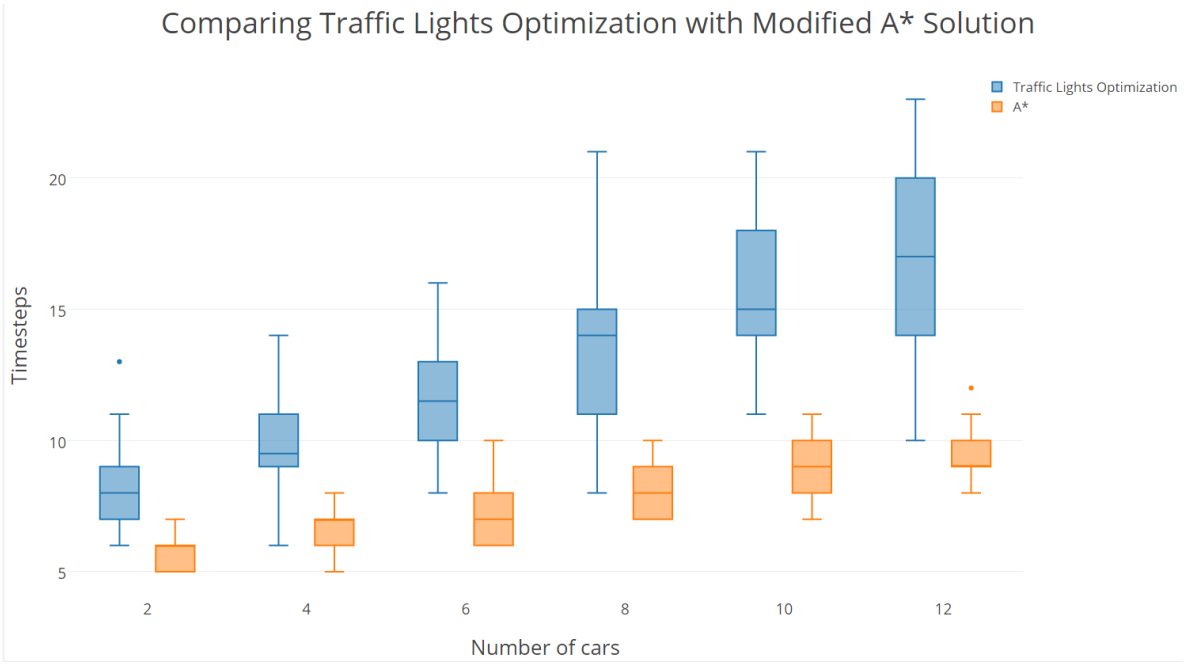
Rysunek 5.9: Wykres zależności kroków czasowych od liczby pojazdów

5.4 Porównanie wyników z rozwiązaniem ze światłami drogowymi

W celu zweryfikowania szybkości podejścia z wykorzystaniem zmodyfikowanej wersji A* pomiary zostały porównane z rozwiązaniem polegającym na optymalizacji świateł drogowych przedstawionym w pracy [TUTAJ CYTAT PRACY SEBASTIANA].

Wykonując pomiary uzgodnione zostały wspólne warunki w obu rozwiązanich tak, aby można było w krokach czasowych porównać oba rozwiązania.

Wykres z wynikami pokazany jest na rysunku 5.10



Rysunek 5.10: Porównanie rozwiązania ze zmodyfikowanym A* z zoptymalizowanymi światłami drogowymi na skrzyżowaniu ośmiu dróg

Wnioski

6.1 Podrozdział

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

6.2 Podrozdział

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis

urna dictum turpis accumsan semper.

6.3 Podrozdział

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Spis rysunków

Bibliografia

[1] Ando M., Nishi T., Konishi M., Imai J.: An Autonomous Distributed Route Planning Method for Multiple Mobile Robots. In: *Transactions of the Society of Instrument and Control Engineers*, vol. 39, pp. 759–766, 2003.

[2] Athmaraman N., Soundararajan S.: Adaptive predictive traffic timer control algorithm. In: *Proceedings of the 2005 Mid-Continent Transportation Research Symposium*. 2005.

[3] Brockfeld E., Barlovic R., Schadschneider A., Schreckenberg M.: Optimizing traffic lights in a cellular automaton model for city traffic. In: *Physical Review E*, vol. 64(5), p. 056132, 2001.

[4] Broxmeyer C.: Vehicle longitudinal control and collision avoidance system for an automated highway system, 1994. URL `https://www.google.com/patents/US5369591`. US Patent 5,369,591.

[5] De Schutter B., De Moor B.: Optimal traffic light control for a single intersection. In: *European Journal of Control*, vol. 4(3), pp. 260–276, 1998.

[6] Dechter R., Pearl J.: Generalized best-first search strategies and the optimality of A. In: *Journal of the ACM (JACM)*, vol. 32(3), pp. 505–536, 1985.

[7] Delling D., Sanders P., Schultes D., Wagner D.: Engineering route planning algorithms. In: *Algorithmics of large and complex networks*, pp. 117–139. Springer, 2009.

[8] ElHalawany B.M., Abdel-Kader H.M., TagEldeen A., Elsayed A.E., Nossair Z.B.: Modified A* algorithm for safer mobile robot navigation. In: *Modelling, Identification &*

Control (ICMIC), 2013 Proceedings of International Conference on, pp. 74–78. IEEE, 2013.

[9] Ferreira M., Fernandes R., Conceição H., Viriyasitavat W., Tonguz O.K.: Self-organized traffic control. In: *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, pp. 85–90. ACM, 2010.

[10] Fleischmann B., Gnutzmann S., Sandvoß E.: Dynamic vehicle routing based on online traffic information. In: *Transportation science*, vol. 38(4), pp. 420–433, 2004.

[11] Gazis D., Jaffe R., Pope W.: Optimal and stable route planning system, 1997. URL `https://www.google.com/patents/US5610821`. US Patent 5,610,821.

[12] Huang Y., Yi Q., Shi M.: An improved Dijkstra shortest path algorithm. In: *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE)*, pp. 0226–0229. 2013.

[13] Kanoh H.: Dynamic route planning for car navigation systems using virus genetic algorithms. In: *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 11(1), pp. 65–78, 2007.

[14] Lämmer S., Helbing D.: Self-control of traffic lights and vehicle flows in urban road networks. In: *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008(04), p. P04019, 2008.

[15] Leena N., Saju K.: A survey on path planning techniques for autonomous mobile robots. In: *IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)*, vol. 8, pp. 76–79, 2014.

[16] Muntean P.: Mobile Robot Navigation on Partially Known Maps using a Fast A Algorithm Version. In: .

[17] Oleiwi B.K., Hubert R., Kazem B.: Modified Genetic Algorithm based on A* algorithm of Multi objective optimization for Path Planning. In: *6th International Conference on Computer and Automation Engineering*, vol. 2, pp. 357–362. 2014.

[18] Shaikh E.A., Dhale A.: AGV Path Planning and Obstacle Avoidance Using Dijkstra's Algorithm. In: *International Journal of Application in Engineering and Mangement (IJAIEM)*, 2013.

[19] Wojnicki I., Ernst S., Turek W.: A robust planning algorithm for groups of entities in discrete spaces. In: *Entropy*, vol. 17(8), pp. 5422–5436, 2015.