Uniwersytet Jagielloński
Wydział Matematyki i Informatyki

KAMIL GÓRSKI

SKUTECZNE UCZENIE ZE WZMOCNIENIEM DLA POJAZDÓW AUTONOMICZNYCH W RUCHU MIESZANYM

Praca magisterska napisana pod kierunkiem
dra inż. Rafała Kucharskiego

Kraków 2022

Spis treści

1 Wstęp **3**
- 1.1 Tło pracy badawczej . . . 3
- 1.2 Eksperymenty z użyciem uczenia ze wzmocnieniem . . . 4
- 1.3 Symulatory używane w badaniach . . . 5
- 1.4 Eksperyment Sugiyamy . . . 5

2 Problem **6**

3 Metoda **7**
- 3.1 Podstawowe definicje i własności . . . 7
- 3.2 Matematyczne podstawy uczenia ze wzmocnieniem . . . 7
- 3.3 Skończony proces decyzyjny Markowa . . . 7
- 3.4 Nagrody . . . 8
- 3.5 Strategie . . . 9
- 3.6 Optymalność strategii . . . 10

4 Metody optymalizacji strategii **11**
- 4.1 Inne formy gradientu strategii . . . 12
- 4.2 Proximal Policy Optimization (PPO) . . . 12
- 4.3 Matematyczny model kierowcy . . . 13

5 Opis narzędzi potrzebnych do symulacji **13**
- 5.1 OpenAI Gym . . . 14
- 5.2 SUMO . . . 14
- 5.3 Ray . . . 14
- 5.4 Flow . . . 15

6 Opis eksperymentów **15**
- 6.1 Symulacja eksperymentu Sugiyamy . . . 16
- 6.2 Ósemka . . . 17
- 6.3 Cukierek . . . 19

7 Wyniki **20**
- 7.1 Pierścień . . . 20
- 7.2 Ósemka . . . 22
- 7.3 Cukierek . . . 26

8 Podsumowanie **28**

Streszczenie

Większość prac, które opisują algorytmy dla autonomicznych pojazdów, skupia się na samolubnym wybieraniu najszybszej trasy, bez brania pod uwagę konsekwencji innych uczestników ruchu. Takie wybory mogą wpływać negatywnie na ruch dla całej sieci. Zadaniem niniejszej pracy jest zaprezentowanie wpływu autonomicznych pojazdów w ruchu mieszanym na różne parametry sieci drogowych, takie jak średnia prędkość wszystkich pojazdów czy zużycie paliwa. W szczególności zostanie zbadane czy wyuczony pojazd lub pojazdy są w stanie je poprawić, przy jednoczesnym zachowaniu swojej optymalnej trasy. Całość będzie dopełniał wstęp teoretyczny algorytmów użytych do nauki oraz opis narzędzi wykorzystanych do symulacji.

1 Wstęp

Liczba ludności w miastach gwałtownie wzrasta [22], co przekłada się proporcjonalnie na zwiększoną liczbę pojazdów poruszających się po ich ulicach. Z tego względu, potrzeba sprawnego transportu miejskiego stale rośnie. Okoliczności tego zjawiska sprawiają, że zatory drogowe stają się jednym z głównych problemów w rozwoju miast. Autonomiczne auta (z ang. *Autonomous Vehicles - AVs*) są jednym z nowoczesnych sposobów rozwiązywania tego problemu. Mają one na celu ekologiczne, zrównoważone sterowanie ruchem oraz optymalizację trasy każdego pojazdu w sieci transportowej poprzez skrócenie czasu podróży do konkretnego celu.

Co więcej, *AVs* pozwalają na znaczną redukcję wypadków. Wyniki badań przedstawione w [15, 8] jasno mówią, że wprowadzenie autonomicznych pojazdów na drogi mogłoby ocalić nawet 10 milionów istnień ludzkich w ciągu dekady, tworząc jeden z najważniejszych postępów w dziedzinie zdrowia publicznego w XXI wieku. Mogą one również zapewnić możliwość zmniejszenia zużycia energii, paliwa oraz emisji spalin do środowiska poprzez poprawę płynności ruchu [30].

Przewiduje się, że pojazdy autonomiczne zostaną wprowadzone do życia codziennego w niedalekiej przyszłości, a ich pełne przyjęcie w wybranych obszarach spodziewane jest już w 2050 roku [42]. Jednakże, to wszystko zależy od intensywności przeprowadzanych badań w temacie. W niniejszej pracy, zostanie opisana propozycja wdrożenia altruistycznych pojazdów autonomicznych, które poprawiają płynność ruchu w sieci drogowej poprzez redukcję zatorów drogowych.

1.1 Tło pracy badawczej

Zjawisko powstawania korków jest dość powszechne, jednak jego przyczyna nie jest do końca zrozumiała. Charakteryzują się one wolniejszymi prędkościami pojazdów, co przekłada się na dłuższy czas podróży, powodując opóźnienia w transporcie oraz zdenerwowanie wśród kierowców. Doprowadza to również do powstawania dużej ilości spalin trafiających do środowiska, co pośrednio wpływa na nas wszystkich. Najprostszym wyjaśnieniem tego zjawiska jest nagromadzenie się zbyt dużej liczby różnorodnych pojazdów na tym samym odcinku drogi, co powoduje jego przeciążenie. Wiele z nich tworzy się nawet przy braku tzw. *wąskiego gardła*, czyli braku sygnalizacji świetlnej, zamkniętych dróg na czas budowy czy nagłych wypadków blokujących pas ruchu.

Okazuje się, że ludzie bardzo często tworzą nieświadomie tzw. korki widma (ang. *phantom jams*) [39], które powstają przy dużym zagęszczeniu samochodów. Wtedy nawet najmniejsze zakłócenia (zbyt mocne hamowanie lub zbytnie zbliżanie się do innego samochodu) mogą szybko ulec wzmocnieniu i przekształcić się w samopodtrzymujący się korek. Zespół z *MIT* opracował model matematyczny [11], który prezentuje powstawanie takich korków, porównując je do równań opisujących fale detonacyjne wytwarzane przez eksplozję. Lepsze zrozumienie struktury korków drogowych może być korzystne dla przyszłych symulacji oraz predykcji rzeczywistego ruchu na autostradach.

Wszystko to jest spowodowane niedoskonałością natury ludzkiej. Czas reakcji podczas zwalniania jest zdecydowanie szybszy niż podczas przyspieszania, a również pojazdy w sensie me-

chanicznym przyspieszają mniej gwałtownie niż zwalniają. Idealny scenariusz zakłada, że ludzie zbliżający się do korka utrzymują stałą prędkość, która pozwala im dotrzeć do niego w momencie odjechania ostatniego samochodu. Dzięki temu, pojazd dojeżdżający do tego co kiedyś było korkiem, nigdy się nie zatrzymuje i tworzy za sobą własną falę ruchu. Kierowcy z przodu ruszają na tyle szybko, na ile pozwala im odległość pomiędzy autem przed nimi, aby jak najszybciej rozluźnić trasę. Niestety, ludzie nie są w stanie tak dobrze się ze sobą zsynchronizować, a dodając do tego zachowania niecierpliwych kierowców, tworzone są warunki drogowe, utrudniające otaczającemu ruchowi płynność.

Takie zachowania starają się odwzorowywać modele kierowcy (ang. *car-following models - CFMs*). Wykorzystywane do opisywania wzorców zachowań pojazdów kierowanych przez człowieka (*human vehicles - HVs*) uważane są one za najważniejszych przedstawicieli mikroskopowych modeli przepływu ruchu w badaniu zachowań komunikacyjnych, w celu rozwiązywania problemów zatorów drogowych [38]. Do pierwszych badaczy, którzy wprowadzili elementy dynamiki ruchu pojazdów należą: Reuschel [29] w 1950 roku oraz Pipes [27] w 1953 roku. W swoich pracach udało im się uwzględnić ważny element współczesnego modelowania ruchu, a mianowicie bezpieczną odległość pomiędzy pojazdami. Ich modele są w stanie obliczyć prędkość oraz przyspieszenia poszczególnych pojazdów w dynamicznym układzie symulującym ruch drogowy, co czyni je podstawą wśród najnowszych usprawnieniach *CFMs*.

1.2 Eksperymenty z użyciem uczenia ze wzmocnieniem

Ostatnio liczne badania skupiły się na zadaniach prowadzenia pojazdów w warunkach dynamicznych opartych na uczeniu ze wzmocnieniem (z ang. *Reinforcement Learning - RL*), które jest podzbiorem uczenia maszynowego. Celem *RL* jest maksymalizacja nagrody ze stanu i obserwacji, zamiast znalezienia ukrytej struktury w danych wejściowych. Ostatnie badania wykorzystują *RL* do zagadnień transportowych - mianowicie, adaptacyjnego sterowania sygnalizacją świetlną [1, 19] oraz agentów pojazdów autonomicznych na rondach [12]. Ponadto, rozwój głębokich sieci neuronowych (z ang. *deep neural network*, *DNN*) może wzmocnić reprezentacje ekstrakcji cech dla złożonych zadań, opartych na wielu ukrytych warstwach. Integracja *RL* i *DNN*, która została nazwana głębokim uczeniem wzmacniającym (z ang. *deep reinforcement learning - DRL*), pozwoliła na rozwój wielu nowych algorytmów uczenia, m.in *Proximal Policy Optimization* (*PPO*) [31], który jest jedną z odmian algorytmu *aktor-krytyk*. Został on użyty do nauki strategii w grze *Dota 2* [6], w której sztuczna inteligencja po raz pierwszy w historii pokonała profesjonalną drużynę w rzeczywistym świecie [24]. Dzięki tym sukcesom, *PPO* stał się jednym z dominujących algorytmów używanych do sterowania połączonych autonomicznych pojazdów (z ang. *connected autonomous vehicles*, *CAVs*).

Największa korzyść podejścia z którego korzystają algorytmy uczenia ze wzmocnieniem polega na ogólności. Cała idea może być powiązana z wieloma środowiskami, które znacznie się od siebie różnią. Co więcej - pozwalają szybko osiągnięć ciekawe rezultaty, w dziedzinach w których nawet nie było intensywnych badań nad sztuczną inteligencją [36]. *RL* wykazał się sukcesem w złożonych, ale bogatych w dane środowiska, takich jak gry Atari [21], czy symulatory fizyki [32]. Najlepszym przykładem jest program *AlphaZero*, który osiągnął poziom mistrzowski w kilka godzin w grach, takich jak szachy [16], *Shogi*, czy *Go* [33].

Berridge i Kringelbach [7] dokonali przeglądu neuronalnych podstaw nagrody i przyjemności, wskazując że przetwarzanie nagrody ma wiele wymiarów i angażuje wiele systemów neuronalnych. Przypuszcza się, że uczenie ze wzmocnieniem w bardzo prymitywny sposób stara się odtworzyć działanie naszego mózgu oraz stara się odtworzyć mechanizm odroczonej gratyfikacji. Pierwsze wzmianki o formalizacji tego procesu pojawiły się w pracy badacza Johna Andreae, który stworzył system *STeLLA*, który uczył się metodą prób i błędów z interakcji ze swoim środowiskiem [4]. Jednak przełomowa praca, która zintegrowała procesy decyzyjne Markowa (z ang. *Markov decision process*, *MDP*) oraz *RL* została opublikowana w 1989 roku przez Watkinsa, który zaproponował także jeden z najbardziej rozpoznawalnych algorytmów w dziedzinie - *Q-learning*

[43].

1.3 Symulatory używane w badaniach

Szkolenie i walidacja jazdy autonomicznej, stały się bardzo złożonym problemem poprzez brak jednolitych środowisk do testowania oraz weryfikowania swoich wyników [41]. Obiecującym podejściem jest symulacja jazdy autonomicznej w środowisku fizycznym. Takie symulacje umożliwiają programy, które reprezentują *HVs* oraz *CAVs* w świecie rzeczywistym opisane powyższymi metodami. Jednym z nich jest symulator samochodów wyścigowych (*The Open Racing Car Simulator - TORCS*), który jest wieloagentowym symulatorem samochodu, opartym na sztucznej inteligencji [46]. Nie obsługuje on jendak symulacji jazdy miejskiej. Brakuje w nim takich czynników jak zasady ruchu drogowego, sygnalizacji świetlej, czy skrzyżowań. Otwarty miejski symulator jazdy *CARLA* (z ang. *car learning to act*) [10] stał się ostatnio popularny przez m.in Ubera. Jest jednak trójwymarowym symulatorem do testowania pojedynczych pojazdów autonomicznych, co uniemożliwiałoby stworzenie wieloagentowych scenariuszy. *Aimsum* (z ang. *Simulation and AI for future mobility*) [3] jest kompletnym symulatorem ruchu mobilności miejskiej. Jego zaletą jest prostota tworzenia sieci, ustawiania parametrów oraz tworzenia animacji. Nie jest to jednak otwarto źródłowy program, co znacznie utrudnia odtwarzanie wyników, czy tworzenie własnych symulacji. *SUMO* (z ang. *Simulation of Urban MObility*), jest pakietem *open source* zaprojektowanym do obsługi dużych sieci [18]. Pozwala on na złożone symulacje z wykorzystaniem wielu autonomicznych agentów, dlatego jest on wykorzystywany do różnych celów badawczych. W odróżnieniu od *Aimsum*, *SUMO* jest trudniejszym w wykorzystaniu symulatorem, przez co nakłada na użytkownika dodatkową presję przy tworzeniu scenariuszy.

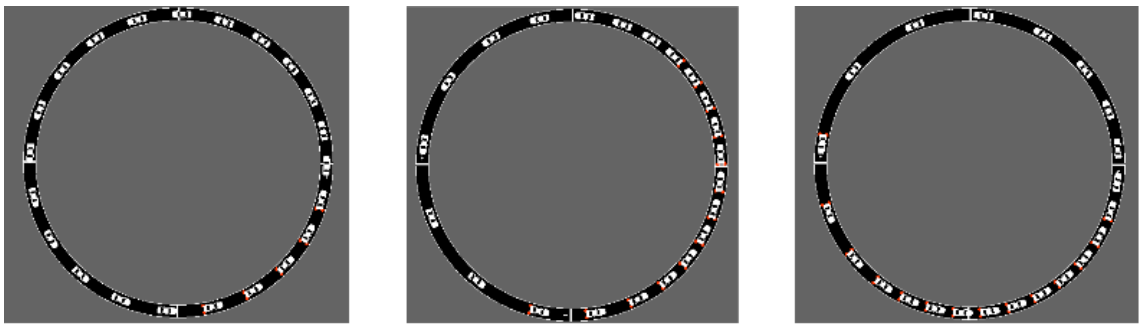
Ponadto Flow, który jest narzędziem opartym na Pythonie, integruje symulator (tj. *SUMO*, *Aimsum*) oraz bibliotekę *RL* (tj. *RLlib*). Tym samym integracja SUMO i Flow stała się nadzieją na podejście do sterowania *CAVs* w środowiskach o mieszanym ruchu. *FLOW* z połączeniem algorytmu *PPO* został wykorzystany do wielu badań, m.in zbadano efektywność wielu agentów w sieci miejskiej [37], jak dobrze radzą sobie na prostych skrzyżowaniach [28] oraz w nagłych sytuacjach [44]. Zaczynają się także próby tworzenia generycznego szablonu do porównywania wydajności agentów [41] . Średnia prędkość stała się obiecującą metryką do weryfikacji polityki szkoleniowej w rzeczywistym środowisku. Wpływa ona bezpośrednio na redukcję czasu opóźnienia, czy zużycia paliwa i emisji szkodliwych substancji.

1.4 Eksperyment Sugiyamy

Jednym z flagowych eksperymentów, jest ustawienie kolejki pojazdów poruszających się po kołowej drodze w warunkach jednorodnego pasa ruchu na płaskim terenie [35]. Wprowadzana jest odpowiednia liczba pojazdów, tak aby gęstość spełniała warunek początku niestabilności swobodnego przepływu, który szacowany jest poprzez model matematyczny [5]. Symulacja tego eksperymentu zaprezentowana została na Rysunku 1.

Autorzy próbują przedstawić w nim, że zachowanie kierowców poruszających się po pierścieniu, nie różni się od tego co widzimy codziennie na drogach. Kolumna pojazdów porusza się z prędkością około $20km/h$, co jest powszechną prędkością korka mierzoną na rzeczywistych autostradach [14]. Mechanizm powstawania korka w praktycznej sytuacji na autostradach, tłumaczy się w zasadzie przez podobny scenariusz do eksperymentu na drodze kołowej. Gdy duża liczba pojazdów, przekraczająca pojemność drogi, jest sukcesywnie wprowadzana na drogę, gęstość przekracza wartość krytyczną i stan swobodnego przepływu staje się niestabilny [35].

Można spotkać się ze stwierdzeniami, że długoterminowe korzyści z *AV*, takie jak bezpieczniejsze drogi, czy lepszy przepływ sieci mogą być odczuwalne dopiero wtedy, gdy zrobotyzowane pojazdy będą stanowiły większość ruchu na drogach. Dopóki tak się nie stanie, ludzie będą nadal wywierać negatywny wpływ na ruch drogowy - na przykład nadal powodować wypadki.



Rysunek 1. Eksperyment Sugiyamy

Symulacja stworzona w programie *SUMO*. Na drodze w kształcie pierścienia o długości 230m, 22 pojazdy starają się poruszać ruchem jednostajnym. Przed rozpoczęciem eksperymentu, pojazdy zostały dostosowane do prędkości $30km/h$. Kierowcy zostali poinstruowani, tak aby bezpiecznie podążać za pojazdem jadącym z przodu, a także starać się utrzymać prędkość. W początkowej fazie, swobodny przepływ jest utrzymywany przez pewien czas, jednak po czasie zaczynają pojawiać się niewielkie fluktuacje w odstępach drogi i rozwijały się one wraz z upływem czasu. Swobodny przepływ zostaje zakłócony, przez co pojazdy nie mogą już poruszać się jednorodnie. W końcu kilka pojazdów zmuszonych jest do zatrzymania się na chwile. Zaobserwowano, że pojazdy wewnątrz skupiska korka zatrzymują się całkowicie, a pojazdy na zewnątrz poruszają się swobodnie. Pojazd znajdujący się z przodu skupiska zaczyna przyspieszać i ucieka z niego, podczas gdy inny pojazd dociera do tyłu, aby następnie się zatrzymać.

Badania [37, 28, 44, 41] sugerują, że dodanie niewielkiej liczby autonomicznych pojazdów może zmniejszyć zatłoczenie na naszych drogach. Na podstawie *korka widmo* pojawiającego się na okręgu, gdzie jeden samochód bez powodu hamuje prowadząc do zniszczenia całego przepływu ukazane zostało, że umieszczenie tylko jednego autonomicznego samochodu może stłumić skutki nieporządanych akcji u ludzi. Eksperyment pokazany na Rysunku 1, został odtworzony w warunkach rzeczywistych [34] z zamianą jednego pojazdu kierowanego przez człowieka na pojazd autonomiczny. Liczby są imponujące: obecność tylko jednego *AV* zmniejsza standardowe odchylenie prędkości wszystkich samochodów w korku o około 50 procent, a liczba ostrych hamowań jest zmniejszona z około 9 na pojazd na km do co najwyżej 2.5, a czasami praktycznie do zera. Obecność pojazdu autonomicznego również zmniejsza zużycie paliwa. Samochody nie muszą już zwalniać i od nowa odzyskiwać swojej prędkości. Według obliczeń zespołu, w rzeczywistości oszczędność wynosi aż 40 procent, gdy jest ona uśredniona dla wszystkich samochodów.

2 Problem

Dotychczas w symulacjach były sprawdzane bardzo proste scenariusze, w których brany był pod uwagę jeden wymiar decyzji - jakie przyspieszenie należy nadać pojazdowi. Zostaje ona rozszerzona o wybór trasy na prostych skrzyżowaniach bez sygnalizacji. Dodatkowo, wprowadzany jest element symulowania zachowań ludzkich, a mianowicie *HV* wybierają różne trasy z pewnym prawdopodobieństwem, a więc całe środowisko nie jest deterministyczne. Takie pojazdy mają egoistyczne cele, które powodują niską wydajność ruchu w sieci. Pojazdy autonomiczne mogą przezwyciężyć tę nieefektywność poprzez doskonałą koordynację. Wykorzystanie niewielkiej liczby *AVs*, może poprawić wydajność systemu transportowego. *CAVs* są w stanie komunikować się oraz koordynować swoje zachowanie, które jest kontrolowane przez agenta *DRL*.

W kolejnym rozdziale zaprezentowany będzie model matematyczny, którego podstawy ukazał Watkins [43]. Następnie opisany zostanie algorytm *PPO*. Bazowa symulacja, która posłuży do porównania wyników, będzie symulować zachowania ludzkie za pomocą modelu kierowcy. Proponowana metoda łączy agentów *DRL* i symulator ruchu poprzez narzędzie *Flow*. Ponadto, zostanie przeglądnięty zestaw hiper parametrów, aby poprawić wydajność agentów. Co więcej, celem jest przedstawienie eksperymentów wraz z wynikami. Wszystkie scenariusze zostaną poddane ocenie efektywności pojazdów autonomicznych, w warunkach ruchu mieszanego w sieci miejskiej. Na końcu zostaną wyciągnięte wnioski oraz podsumowana cała praca.

3 Metoda

Na potrzeby tej pracy zdefiniujemy kilka ważnych pojęć, które przydadzą się nam do scharakteryzowania modelu. Z racji braku ogólnie przyjętej definicji, np. agenta programowego [25], będą prezentowane definicje intuicyjne.

3.1 Podstawowe definicje i własności

Definicja 3.1. *Agent* to byt, który jest w stanie interpretować zmieniające się otoczenie i obiekty, które się w nim znajdują, a także potrafi podejmować niezależne decyzje mające na nie wpływ.

Definicja 3.2. *Środowisko* jest światem otaczającym agenta w którym egzystuje. Agenci mogą wchodzić w interakcje ze środowiskiem poprzez wykonywanie akcji, jednak nie mogą wpływać na zasady lub dynamiki środowiska poprzez te akcje. Dla pojazdów autonomicznych oznacza to, że mają one ustalone drogi po których mogą jechać, natomiast nie są w stanie np. wypaść z trasy. *Środowisko* dostarcza także nagrody dla agenta stanowiące ocenę skuteczności jego działania.

Definicja 3.3. *Pozycja*, *stan* lub *obserwacja* jest to układ danego środowiska. *Stany* mogą być zmieniane pod wpływem wykonywania akcji przez agenta. Dla pojazdów może być to wektor ich bezwzględnej pozycji w sieci drogowej.

3.2 Matematyczne podstawy uczenia ze wzmocnieniem

Modelem matematycznym problemu uczenia się ze wzmocnieniem jest proces decyzyjny Markowa [36]. Praca skupia się jedynie na skończonych procesach, czyli takich gdzie zbiór stanów, akcji oraz nagrody mają skończoną liczbę elementów.

3.3 Skończony proces decyzyjny Markowa

Pierwszą informacją, którą otrzymuje agent w środowisku, to stan początkowy S_0. Wykonując dowolną możliwą akcję A_0, wchodzi on w interakcję z nim (Rysunek 2), czego konsekwencją jest otrzymana nagroda R_1 oraz informacja o kolejnym stanie S_1. Jeśli ciąg jest skończony, to ostatnim elementem jest stan końcowy S_T, gdzie T jest terminalnym krokiem agenta. W praktyce, stan terminalny musi być zawsze określony, ponieważ każdy przeprowadzony eksperyment jest skończony. W przypadku tej pracy, może być on np. wypadkiem w którym udział biorą dwa pojazdy, lub ostatnim krokiem symulacji. Taki sekwencyjny ciąg zdarzeń τ z dyskretnym czasem t jest nazywany trajektorią, lub interakcją agenta ze środowiskiem.

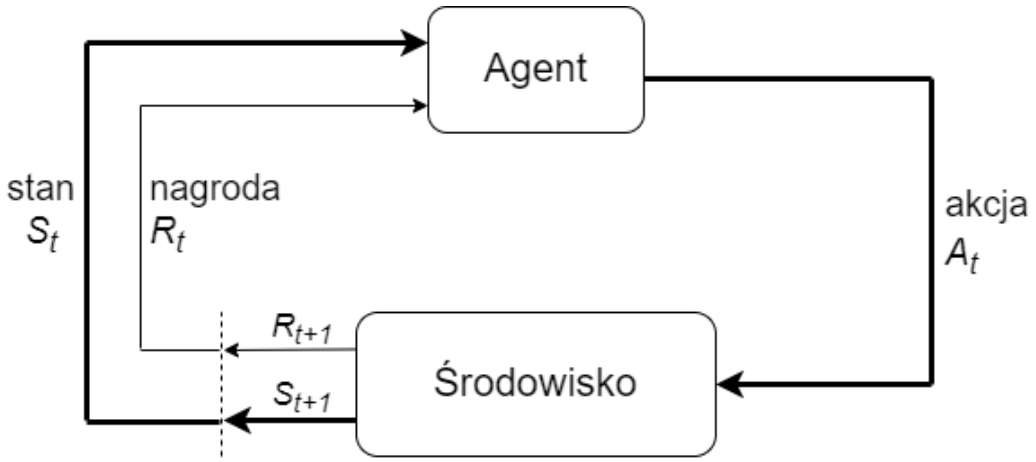
$$\tau = S_0, A_0, R_1, S_1, A_1, R_2, S_3, ... \tag{1}$$

MDP jest prostym ujęciem problemu uczenia się na podstawie interakcji, aby osiągnąć cel. Definiuje on w sposób precyzyjny informacje zwracane przez środowisko, na podstawie wybranych akcji przez agenta programowego. Proces ten jest klasyczną formalizacją sekwencyjnego podejmowania decyzji oraz nagradzania ich poprzez przyszłościowe nagrody.

Definicja 3.4. Skończony proces decyzyjny Markowa jest czwórką (S, A, p, r), gdzie

1. S jest zbiorem wszystkich możliwych stanów w których agent może się znaleźć. Stan $s \in S$ w środowisku w czasie t jest zmienną losową, którą oznaczamy S_t.

2. A jest zbiorem wszystkich możliwych akcji, które agent może wykonać. Akcja $a \in A$ wykonana w czasie t jest zmienną losową, zapisywaną jako A_t.



Rysunek 2. Interakcja agenta ze środowiskiem w *procesie decyzyjnym Markowa* [36].

Obiektem uczącym się jest agent, a wszystko poza nim jest środowiskiem. W każdym kroku czasowym $t \in \mathcal{N}_0$, agent wybierając daną akcję ma wpływ na środowisko, które po każdej wykonanej akcji wynagradza lub karze agenta wartością liczbową $R_t \in \mathbb{R}$ nazywaną sygnałem wzmocnienia. Konsekwencją tego jest znalezienie się agenta w kolejnym stanie S_{t+1}

3. $p : S \times \mathbb{R} \times S \times A \to [0, 1]$ jest tzw. funkcją *przejścia stanów*.
   Dla wszystkich $(s, r, s, a) \in S \times \mathbb{R} \times S \times A$, niech

$$p(s', r|s, a) := P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a), \tag{2}$$

   będzie prawdopodobieństwem warunkowym znalezienia się w stanie s' oraz otrzymaniu nagrody r w kolejnym kroku czasu t, po wykonaniu akcji a w stanie s.

4. $r : S \times A \to \mathbb{R}$ jest *funkcją nagrody* (*wzmocnienia*).
   Dla wszystkich $(s, a, s', t, x) \in S \times A \times S \times N_{\geq 0} \times \mathbb{R}$, niech:

$$r(s, a) = \mathbb{E}\left[R_t|S_{t-1} = s, A_{t-1} = a\right] = \sum_{x \in \mathbb{R}} x \sum_{s' \in S} p(s', r|s, a), \tag{3}$$

   będzie nagrodą otrzymaną za przejście ze stanu s do s' pod wpływem wykonania akcji a.

W *procesie decyzyjnym Markowa*, prawdopodobieństwa określone przez funkcję p całkowicie charakteryzują dynamikę środowiska. Oznacza to, że prawdopodobieństwo każdej możliwej wartości dla S_t oraz R_t zależy tylko od bezpośrednio poprzedzającego stanu S_{t-1} oraz akcji A_{t-1}. Z takimi ograniczeniami model zachowuje własność Markowa, tzn. funkcje p oraz r nie zależą od przeszłości agenta. W każdym kroku czasowym, nagroda oraz kolejny stan zależą od aktualnego stanu oraz akcji w nim wykonanej.

3.4 Nagrody

Główną zaletą modelu uczenia ze wzmocnieniem jest to, że nie opiera się ono na wcześniej zdefiniowanym zbiorze danych. Generowanie nagrody nie zależy od wiedzy o tym, jakie powinne być prawidłowe działania agenta. Sukces aplikacji uczącej się ze wzmocnieniem zależy jednak od tego, jak dobrze funkcja nagrody określa cel projektanta aplikacji i jak dobrze ocenia on postępy w osiąganiu tego celu. Zaprojektowanie jej jest krytyczną częścią każdej aplikacji wykorzystującej uczenie ze wzmocnieniem. Funkcja ta zwraca tzw. *sygnał wzmocnienia*, który jest wzorowany na pojęciu wzmocnienia z psychologii, co przekłada się także na nazwę metody. Wzmocnienie zakłada, że zachowania nagradzające daną osobę, będą najprawdopodobniej powtórzone, natomiast te pociągające za sobą karę wręcz przeciwnie, czego skutkiem będzie zmiana zachowania [13].

Często natrafiamy na problem uczenia ze skąpych danych (ang. *sparse reward*). Nawet jeśli mamy prosty oraz łatwy do zidentyfikowania cel, który agent musi osiągnąć z wielu warunków

początkowych, może to być ciężkim wyzwaniem. Funkcja nagrody powinna dostarczać agentowi sygnał, który odzwierciedla w rzeczywistości jak blisko jest celu. Niestety, agent może błądzić bez sensu przez długi czas, co Minsky już w 1961 nazwał *problemem plateau* [20]. W praktyce, projektowanie funkcji nagrody jest często pozostawione nieformalnemu poszukiwaniu metodą prób i błędów, który daje akceptowalne wyniki, co zostało pokazane w Rozdziale 7.

W uczeniu ze wzmocnieniem cel agenta jest sformalizowany. Dąży on do zmaksymalizowania *oczekiwanej nagrody*, która w najprostszej postaci jest sumą wszystkich nagród. Jeśli wyciągniemy sekwencję nagród $R_1, R_2, R_3, ...$ z trajektorii (1), otrzymujemy:

$$G_t := R_{t+1} + R_{t+2} + ... + R_T. \tag{4}$$

Intuicyjnym celem jest zmaksymalizowanie ogólnej nagrody, a nie tylko tej która jest zwracana z aktualnego stanu. Dlatego, cały koncept jest uzupełniony o tzw. współczynnik dyskontowania γ (z ang. *discount factor*). Zgodnie z tym podejściem, agent stara się wybrać te akcje, które dają mu największą możliwą wartość sumy zdyskontowanej nagród.

$$G_t := R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{5}$$

gdzie γ jest parametrem $0 \leq \gamma \leq 1$ nazwanym współczynnikiem dyskontowania. Współczynnik ten determinuje wartość przyszłych nagród względem obecnego stanu. Nagroda otrzymana w k-tym kroku w przyszłości jest warta tylko γ^{k-1} razy mniej od nagrody otrzymanej natychmiast. Im bardziej parametr zbliża się do zera, tym bardziej agent będzie starał się zmaksymalizować nagrody, które otrzymuje natychmiastowo.

Bardzo ważnym aspektem obliczenia G_t jest fakt, iż nagrody w kolejnych krokach czasowych są ze sobą powiązane, co jest kluczowym faktem w tworzeniu algorytmów uczenia ze wzmocnieniem:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + ... \right) \\ &= R_{t+1} + \gamma G_{t+1}. \end{aligned} \tag{6}$$

3.5 Strategie

Sygnał wzmacniający, który dostaje agent, zależy od tego jakie działania podejmie. Zmaksymalizowanie wyrażenia (6), polega na odpowiednim doborze akcji o czym mówi strategia działania agenta.

Definicja 3.5. Strategią π nazywamy odwzorowanie $S \times A \mapsto [0, 1]$. Jeśli agent podąża strategią π w czasie t, wtedy

$$\pi(a|s) = P(A_t = a|S_t = s), \tag{7}$$

jest rozkładem prawdopodobieństwa wykonania akcji a dla każdego stanu s.

Definicja 3.6. *Funkcja wartości* dla stanu *s* korzystająca ze strategii π, zapisywana jest jako $v_\pi(s)$ i mówi nam jaki jest oczekiwany zwrot, kiedy agent zaczyna ze stanu s i korzysta ze strategii π. Dla procesów decyzyjnych Markowa, funkcja $v_\pi(s)$ jest opisana formalnie przez równanie:

$$v_\pi(s) := \mathbb{E}[G_t|S_t = s] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right], \forall s \in S \tag{8}$$

Prawie wszystkie algorytmy uczenia ze wzmocnieniem wymagają estymacji funkcji wartości stanu, które szacują jak dobre jest położenie agenta w dowolnym stanie. Definiuje to przyszła

nagroda, której możemy oczekiwać (6) podążając pewną strategią π. W związku z tym funkcje wartości są definiowane w odniesieniu do konkretnych sposobów działania agenta.

Podobnie definiujemy wartość dobierania akcji *a* w stanie *s* korzystając z polityki π:

$$q_\pi(s,a) := \mathbb{E}[G_t|S_t = s, A_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right], \forall s \in S, a \in A \quad (9)$$

Fundamentalną własnością tych funkcji wykorzystywanych w algorytmach uczenia ze wzmocnieniem jest to, że spełniają one zależności rekurencyjne takie jak (6). Dla dowolnej strategii π oraz stanu *s* otrzymujemy:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}\left[G_t|S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1}|S_t = s\right] \\ &= \sum_{a\in A} \pi(a|s) \sum_{s'\in S} \sum_{r\in R} p(s', r|s, a) \left[r + \gamma \mathbb{E}[G_{t+1}|S_{t+1} = s']\right] \\ &= \sum_{a\in A} \pi(a|s) \sum_{s'\in S} \sum_{r\in R} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right] \end{aligned} \quad (10)$$

Równanie to nazywane jest równaniem Bellmana dla v_π. Pokazuje to związek pomiędzy wartością, a stanem oraz wartościami stanu kolejnego oraz jest podstawą prawie wszystkich algorytmów *RL*. Za pomocą wprowadzonych funkcji możemy zdefiniować funkcję korzyści. Dla wszystkich $(s, a) \in S \times A$,

$$A^\pi(s_t, a_t) = q^\pi(s_t, a_t) - v^\pi(s_t). \quad (11)$$

Funkcja ta opisuje o ile lepsza lub gorsza jest wykonana akcja od innych, w stosunku do obecnej polityki.

3.6 Optymalność strategii

Rozwiązanie problemu uczenia się ze wzmocnieniem oznacza znalezienie strategii, która osiąga maksymalnie duże nagrody w określonym przedziale czasu. Posługując się powyższymi definicjami można sformalizować pojęcie optymalności strategii π.

Definicja 3.7. Mówimy, że strategia π' jest nie gorsza od strategi π, jeśli jej oczekiwany zwrot jest większy lub równy od zwrotu π dla każdego stanu. Dowolna strategia π' jest większa bądź równa π wtedy i tylko wtedy, gdy $v'_\pi(s)$ jest większa bądź równa od $v_\pi(s)$, dla każdego $s \in S$. Istnieje co najmniej jedna strategia optymalna, zapisywana jako π_* [36]. Wszystkie optymalne strategie dzielą tą samą funkcję stan-wartość, nazywaną *optymalną funkcją wartości* zapisywaną jako v_* [36].

Twierdzenie 3.1. *Strategią optymalną jest każda strategia maksymalizująca poniższą wartość:*

$$v_*(s) := \max_\pi v_\pi(s), \textit{dla każdego } s \in \mathcal{S}. \quad (12)$$

Optymalne strategie współdzielą także takie same funkcje wartości akcji, które są definiowane jako:

$$q_*(s,a) := \max_\pi q_\pi(s,a). \quad (13)$$

Zdefiniowane funkcje dążą do maksymalizacji oczekiwanej nagrody dla dowolnej trajektorii, na co pozwala zdefiniowana optymalna strategia działania. Jednak w praktyce jest to prawie zawsze niemożliwe. Dlatego w większości przypadków interesuje nas strategia, która jest przybliżeniem strategii optymalnej.

4 Metody optymalizacji strategii

Metody gradientowe strategii, to rodzina algorytmów rozwiązujących problemy związane z uczeniem ze wzmocnieniem poprzez bezpośrednią optymalizację strategii w jej przestrzeni. Głównym celem agenta będzie nauczenie się, w jaki sposób zoptymalizować stochastyczną funkcję strategii:

$$\max_{\theta} \mathbb{E}_{\pi_\theta} R(\tau), \tag{14}$$

gdzie θ jest parametrami strategii. Otrzymanie wyniku wyrażenia (14) jest równoznaczne ze znalezieniem najlepszych parametrów θ, które dają optymalną strategią. Zakładając, że dalsza część pracy bierze pod uwagę jedynie strategie, które są różniczkowalne względem θ (przeważnie są to sieci neuronowe, a θ to jej parametry), można to zrobić to za pomocą metodą gradientu prostego:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau), \tag{15}$$

gdzie α jest współczynnikiem długości kolejnych kroków.

Niech P($\tau|\theta$) będzie prawdopodobieństwem trajektorii τ pod warunkiem strategii π_θ, otrzymujemy:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) &= \nabla_\theta \sum_\tau P(\tau|\theta) R(\tau) \\
&= \sum_\tau \nabla_\theta P(\tau|\theta) R(\tau) \\
&= \sum_\tau \frac{P(\tau|\theta)}{P(\tau|\theta)} \nabla_\theta P(\tau|\theta) R(\tau) \\
&= \sum_\tau P(\tau|\theta) \nabla_\theta log P(\tau|\theta) R(\tau) \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta log P(\tau|\theta) R(\tau)]
\end{aligned} \tag{16}$$

Teraz możemy rozwinąć prawdopodobieństwo trajektorii τ jak poniżej:

$$P(\tau|\theta) = p(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t), \tag{17}$$

gdzie $p(s_0)$ jest rozkładem prawdopodobieństwa stanu początkowego. Biorąc gradient prawdopodobieństwa logarytmu trajektorii dostajemy:

$$\begin{aligned}
\nabla_\theta log P(\tau|\theta) &= \nabla_\theta \left(\log p(s_0) + \sum_{t=0}^{T-1} \left(\log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t) \right) \right) \\
&= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t).
\end{aligned} \tag{18}$$

Podstawiając (18) pod (16), otrzymujemy:

$$\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t|s_t) R(\tau) \right], \tag{19}$$

ponieważ równanie (19) jest wartością oczekiwaną, może być estymowane metodami Monte Carlo:

$$\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) \approx \frac{1}{L} \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t|s_t) R(\tau) \right], \tag{20}$$

gdzie L jest liczbą trajektorii używanej dla jednej aktualizacji gradientu.

4.1 Inne formy gradientu strategii

Zdefiniujmy funkcję, która jest reprezentacją ogólnej formy strategii gradientu:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t)\phi_t\right], \tag{21}$$

gdzie ϕ_t może być jednym z

$$\begin{aligned}
&\phi_t = R(\tau)\\
&\phi_t = \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})\\
&\phi_t = \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) - v(s_t)\\
&\phi_t = q_{\pi_\theta}(s_t, a_t)\\
&\phi_t = A_{\pi_\theta}(s_t, a_t)
\end{aligned} \tag{22}$$

Pomimo zróżnicowania, wszystkie te wybory prowadzą do tej samej oczekiwanej wartości dla gradientu strategii [2]. Jednak formułowanie gradientów strategii za pomocą funkcji korzyści jest niezwykle powszechne.

4.2 Proximal Policy Optimization (PPO)

Algorytm *PPO* został stworzony głównie po to, aby uniknąć destrukcyjnie dużych aktualizacji strategii. Jest motywowany pytaniem, jaki jest największy możliwy krok poprawy strategii przy użyciu danych, które aktualnie posiadamy, bez posuwania się tak daleko, że przypadkowo spowodujemy załamanie wydajności. *PPO* ma kilka sztuczek, które utrzymują nowe strategie blisko starych. Metody *PPO* są znacznie prostsze od innych algorytmów (np. TRPO [32]), które próbują osiągnąć ten sam cel, a także ma lepszą wydajność [31].

Zdefiniujmy funkcję

$$\mathcal{L}^{CLIP}(s, a, \theta_k, \theta) = min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s, a), clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A_{\pi_{\theta_k}}(s, a)\right), \tag{23}$$

gdzie ϵ jest parametrem, który z mówi jak daleko nowa strategia może oddalić się od starej. Równanie (23) można uprościć do

$$\mathcal{L}^{CLIP}(s, a, \theta_k, \theta) = min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s, a), g\left(\epsilon, A^{\theta_k}(s, a)\right)\right), \tag{24}$$

gdzie

$$g(\epsilon, A) = \begin{cases} 1+\epsilon & A \geq 0 \\ 1-\epsilon & A < 0. \end{cases} \tag{25}$$

Algorytm 1: Proximal Policy Optimization [31]

Wejście: Inicjalizacja parametrów θ oraz progu odcięcia ϵ

for $k = 0, 1, 2, ...$ **do**

- Zbierz zbiór trajektorii $\mathcal{D}_k = \{\tau_i\}$ podążając polityką $\pi_k = \pi(\theta_k)$
- Oblicz A_t
- Zaktualizuj strategię

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} \mathcal{L}^{CLIP}(\theta),$$

4.3 Matematyczny model kierowcy

Zachowanie autonomicznych pojazdów jest zdefiniowane przez Algorytm 1, natomiast dodatkowo potrzeba przedstawienia modelu, który imituje zachowania kierowców ludzkich. Taką podstawową dynamikę *HV* można zdefiniować za pomocą modeli matematycznych, które opisują jakie przyspieszenie powinny nadać pojazdowi. Poprzez obserwacje odpowiadających pojazdów prowadzących - ich prędkości, względnej odległości oraz odstępu między pojazdami można określić podstawowy wzór, który jest opisany następująco:

$$a_i = f(h_i, \dot{h_i}, v_i), \tag{26}$$

gdzie a_i oznacza przyspieszenie pojazdu, które jest obliczane za pomocą nieliniowej funkcji f, v_i to prędkość pojazdu poprzedzającego, $\dot{h_i}$ prędkość względna, a h_i droga między pojazdami.

Mimo, że w literaturze istnieje wiele modeli kierowców, to jednak żaden z nich nie jest w stanie uchwycić wszystkich ludzkich zmysłów, zdolności poznawczych, odruchów, czy nawyków [40]. Wybór modelu zależy więc od zakresu badań symulacyjnych. W niniejszej pracy skupiono się na jednopasmowych skrzyżowaniach, a więc potrzebny jest model, który najlepiej będzie odzwierciedlał zachowania kierowców w takich sytuacjach.

W tym celu został stworzony model inteligentnego kierowcy (ang. *Intelligent Driver Model*, *IDM*), który jest prawdopodobnie najprostszym, kompletnym i bezwypadkowym modelem tworzącym realistyczne profile przyspieszeń oraz wiarygodnie odwzorowuje zachowania w zasadniczo wszystkich sytuacjach ruchu jednopasmowego [38]. Polecenie nadania prędkości pojazdu, jest wyrażone w następujący sposób:

$$a_{IDM} = a \left[1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right], \tag{27}$$

gdzie v_0 oznacza docelową prędkość, δ jest wykładnikiem przyspieszenia, s oznacza odległość między pojazdami, a $s^*(v, \delta v)$ zalecaną odległość, która opisywana jest wzorem:

$$s^*(v, \delta v) = s_0 + max \left(0, vT + \frac{v \delta v}{2\sqrt{ab}} \right), \tag{28}$$

gdzie s_0 oznacza minimalną odległość, T jest to czas dojechania do pojazdu prowadzącego, a Δv różnicą pomiędzy aktualną prędkością, a docelową.

Tabela 1. Parametry dla kontrolera Intelligent driver model (IDM) użyte w symulacjach

Parametr	Oznaczenie	Wartość
Współczynnik przyspieszania	δ	4.0
Komfortowe zwalnianie (m/s^2)	b	$1.5m/s^2$
Odstę (s)	a	$1s$
Zalecana odległość (m)	s_0	$2m$
Przyspieszenie (m/s^2)	s	$1.0m/s^2$
Docelowa prędkość (m/s^2)	v_0	$30.0m/s^2$

5 Opis narzędzi potrzebnych do symulacji

Wszystkie symulacje, tworzone są za pomocą biblioteki *Flow*, która wykorzystuje API programu *SUMO - TraCI*. Środowiska są implementowane za pomocą *OpenAI Gym*, który jest interfejsem *MDP* dla zadań *RL*. *Ray* jest narzędziem służącym do uczenia oraz symulacji *CAVs*, w którym zaimplementowane są użyte algorytmy, m.in *PPO*. Poniżej przedstawiony jest opis poszczególnych bibliotek.

5.1 OpenAI Gym

OpenAI Gym [9] jest biblioteką *open source*, zawierającą zestaw narzędzi do prac związanych z uczeniem ze wzmocnieniem. Umożliwiają one tworzenie i porównywanie algorytmów, zarówno jak i tworzenie własnych środowisk, aby w jednolity sposób testować uczących się agentów. Biblioteka dostarcza standardowe *API* do komunikacji pomiędzy algorytmami uczenia oraz środowiskami.

Bardzo dużą zaletą biblioteki, jest zbiór wielu gotowych środowisk, które odzwierciedlają gry wideo. Do najpopularniejszego zbioru należy *Atari 2600*. Nazwa pochodzi od konsoli gier wideo *Atari 2600*, która została wyprodukowana przez firmę Atari w 1977 roku. Konsola zawiera popularne gry takie jak *Breakout*, *Pac-Man* czy *Space Invaders*. Od czasu wprowadzenia *Depp Q-Network* przez Volodymyra Mniha [21], *Atari 2600* stało się standardem w testowaniu nowych algorytmów uczenia ze wzmocnieniem. Głównie przez to, że jest wymagającym środowiskiem testowym ze względu na wielowymiarowe wejście wideo oraz rozbieżność zadań pomiędzy grami.

Autorzy mieli negatywne doświadczenia z porównywaniem algorytmów *RL* oraz innymi narzędziami, które nie były w tej dziedzinie ustandaryzowane. Postanowili oni więc stworzyć w pełni niezależne oraz kompletne narzędzie, które od momentu wydania stało się standardem w społeczności *RL* do testowania algorytmów. Udostępniona została tablica wyników, która pozwala użytkownikom porównywać wydajność swoich algorytmów. Inspiracją jest *Kaggle*, który prowadzi zestaw konkursów uczenia maszynowego. Autorzy podkreślają, że celem nie jest stworzenie konkurencji, lecz wzajemna współpraca, dzielenie się wynikami oraz kodem [9].

5.2 SUMO

Eclipse SUMO to pakiet do symulacji ruchu. Jest to również projekt *open source*, przez co nie ma problemu z odtworzeniem wyników, czy tworzeniem własnych symulacji. Jest to powód dla którego pakiet ten jest chętnie używany w pracach badawczych. Został on zaprojektowany do obsługi dużych sieci, które pozwalają na symulację wielu pojazdów, a nawet pieszych. Jest również wyposażony w duży zestaw narzędzi do tworzenia scenariuszy, w tym graficzny interfejs. Jednakże dla bardziej zaawansowanych schematów, tworzone są bezpośrednio dokumenty *xml*, które również są generowane w *GUI* (z ang. *graphical user interface*).

Kontrolowanie zachowania wszystkich obiektów symulacyjnych podczas symulacji na żywo umożliwia interfejs *TraCI* (**Tra**ffic **C**ontrol **I**nterface). Pozwala on na pobieranie wartości symulowanych obiektów i manipulowanie ich zachowaniem *on-line*. Wykorzystując architekturę klient-serwer opartą na TCP, zapewnia dostęp do *SUMO*, które działa jako serwer. Jest to główny sposób na kontrolowanie agentów w symulacji, którym podawane są określone akcje które mają wykonać.

5.3 Ray

Ray to ujednolicony *framework* obliczeniowy *open source*, który ułatwia skalowanie obliczeń związanych ze sztuczną inteligencją oraz *Pythonem*. Autorzy pokazali skalowanie powyżej 1.8 miliona zadań na sekundę oraz lepszą wydajność niż istniejące systemy dla kilku wymagających aplikacji uczenia ze wzmocnieniem [23].

RLlib jest standardowym *frameworkiem* dla uczenia ze wzmocnieniem, który jest częścią *Raya* [17]. Oferuje on wsparcie dla wysoce rozproszonych obliczeń na poziomie produkcyjnym, przy jednoczesnym zachowaniu ujednoliconych i prostych *API* dla wielu różnych aplikacji. Zawiera implementacje najnowszych algorytmów, które działają również w trybie wieloagentowym. Posiada prosty interfejs dla środowisk stworzonych na bazie biblioteki *OpenAI Gym*, przez co można w bardzo prosty sposób zintegrować swoje istniejące środowiska z *RLlib*. Bardzo pomocnym narzędziem jest definiowanie danych historycznych zapisywanych do pliku oraz stanów uczenia, tzw. punktów kontrolnych (z ang. *checkpoint*). Biblioteka *Rllib* jest stosowana do kontroli działań autonomicznych pojazdów, w oparciu o wyuczone polityki.

5.4 Flow

Flow, które zostało wprowadzone przez UC Berkeley, działa jako most pomiędzy symulatorem ruchu drogowego *SUMO* oraz biblioteką uczenia się przez wzmacnianie *RLlib* [45]. Zapewnia on interfejs, który pozwala trenować agentów na niestandardowej sieci drogowej, bez konieczności martwienia się o połączenie z symulatorem ruchu i biblioteką do nauki strategii działań agenta. Narzędzie to, pozwala implementować liczne typy sieci drogowych bezpośrednio w kodzie *Pythona*, przez co nie musimy uczyć się jak działa symulator ruchu. Poprzez informacje o parametrach sieci oraz stanie wszystkich pojazdów, zwracane są cechy przewagi stanu dla agentów w celu uzyskania optymalnej strategii jazdy.

Aby rozpocząć symulację za pomocą *Flow* konieczne jest zdefiniowanie sieci. Obsługuje on uczenie się w zasadzie na dowolnych, zdefiniowanych przez użytkownika sieciach, które są poprawne wg. *SUMO*. *Siecią* jest właściwie sieć dróg, w której przechowywane są wszystkie informacje potrzebne do symulacji jak i nauki agentów. Opisuje ona drogi (pozycja, długość, ilość pasów, limit prędkości itp.), połączenia pomiędzy nimi (np. skrzyżowania) oraz inne informacje (np. światła drogowe).

Flow pozwala tworzyć środowiska za pomocą różnych *frameworków* jednak wszystkie symulacje, które ukazano w pracy, stworzone zostały za pomocą *OpenAI Gym*. Zatem *środowisko* jako obiekt, powinno definiować następujące rzeczy:

1. Przestrzeń stanów - opisuje aktualny stan systemu, który jest symulowany. Na przykład mogą to być pozycje wszystkich pojazdów znajdujących się w sieci.

2. Przestrzeń akcji - opisuje wszystkie możliwe akcje, które mogą dokonać agenci. Wszystkie pojazdy wymagają zdefiniowanych jasno praw, mogą one być wcześniej określone lub wyuczone. Prawo sterowania może reprezentować kierowcę w formie ludzkiej, autonomiczne auto lub cały zbiór pojazdów. Standardowa akcja jaką mogą zrobić agenci, to nadanie pewnej prędkości pojazdowi.

3. Funkcja nagrody - opisuje co agent powinien robić, aby osiągnąć swój cel. Standardowa funkcja będzie zwracała średnią prędkość wszystkich pojazdów w sieci (co mówi tyle, że algorytm uczenia stara się zmaksymalizować prędkość wszystkich pojazdów). Można także karać agentów za np. zbyt duże zużycie paliwa lub dużą emisję szkodliwych substancji z pojazdu.

Mając zdefiniowane środowisko oraz opisaną sieć, możemy zaobserwować w niej ruch pojazdów.

Jeśli chcemy dodać do sieci pojazdy autonomiczne, wystarczy zdefiniować konkretny algorytm wraz z jego parametrami. W naszym przypadku jest to *PPO* [31], zaimplementowane w bibliotece *RLlib* [17].

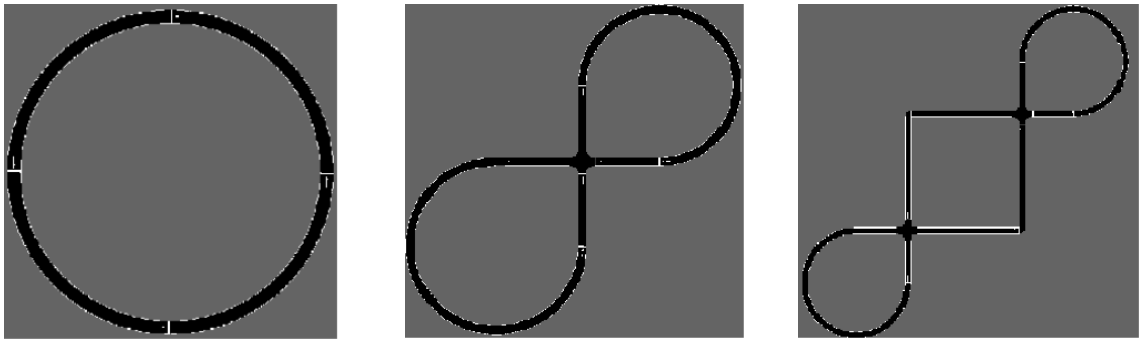
6 Opis eksperymentów

Do przedstawienia eksperymentów wykorzystywany jest symulator *SUMO*. Dzięki niemu, jesteśmy w stanie zaobserwować ruch *HVs* z ustalonymi drogami. Interfejs *TraCI* daje możliwość uruchamiać je w *Pythonie*, a nawet umożliwia sterowanie pojazdami w czasie rzeczywistym. Pozwala to na stworzenie *CAVs*, którym podawane są instrukcje w jaki sposób mają się zachowywać. Strategie te wyuczane są za pomocą biblioteki *Ray[rllib]*, w której zaimplementowany jest algorytm *PPO* z możliwością zmiany jego parametrów. Wszystkie te narzędzia pozwalają stworzyć eksperyment, w którym uczone jest kilka *CAVs*, a po skończonym procesie nauki jest możliwość odtworzenia ich działania w symulatorze.

W tym rozdziale zostanie szczegółowo opisana struktura sieci drogowych (Rysunek 3) oraz możliwość poruszania się po niej pojazdów, co zostanie odpowiednio uzupełnione o zrzuty ekranów, które są pojedynczą klatką z symulacji. Autorzy *Flow* przygotowali szablon dwóch pierw-

szych sieci, a ostatnia jest zaprezentowana po raz pierwszy w niniejszej pracy. Opisane zostaną także zdefiniowane stany, możliwe akcje oraz w jaki sposób nagradzane są pojazdy autonomiczne.

W każdym kroku czasowym symulacji pobierane są dane ze środowiska. Podczas tego kroku, pojazd autonomiczny stara się wykonać pewną akcje, czyli zmianę kierunku drogi lub nadanie odpowiedniej prędkości. Definicja stanu powinna zawierać wszystkie informacje potrzebne dla agenta do podjęcia optymalnej decyzji. Poza tym, powinien on być bezpośrednio skorelowany z nagrodą, co ułatwia proces uczenia się. Reprezentacja stanu w tak skondensowanej formie jest ciężka do uzyskania. Również sformułowanie akcji oraz nagrody mają kluczowy wpływ na działanie agentów, co zostanie przedstawione w eksperymentach. Konkretne stany, akcje oraz nagrody są zdefiniowane indywidualnie dla każdego eksperymentu.



(a) (b) (c)

Rysunek 3. Sieci drogowe użyte w eksperymentach

Eksperymenty przeprowadzone zostały na trzech sieciach drogowych, w której pojazdy poruszają się przeciwnie do ruchu wskazówek zegara. Aby uprościć scenariusze, zostały one zaprojektowane w ten sposób, żeby pojazdy nie mogły opuścić sieci. Są to jednopasmowe drogi, które naśladują najczęstsze doświadczenia, spotykane w rzeczywistym świecie. Droga w kształcie pierścienia ma bardzo podobne właściwości jak pasy na autostradach, a dwie kolejne są odwzorowaniem prostych skrzyżowań bez sygnalizacji świetlnej.

6.1 Symulacja eksperymentu Sugiyamy

Autorzy *FLOW* [45] zaproponowali rozwiązanie eksperymentu zaprezentowanego w 2008 roku [35] oraz opisanego we wstępie pracy (Rozdział 1.1). Przedstawiona została próba jego odtworzenia, która niestety nie działa. W wynikach nastąpi zaprezentowanie alternatywnego rozwiązania tego problemu, który został opisany wraz z symulacją. Parametry potrzebne do zdefiniowania problemu są przedstawione w Tabeli 2, a parametry użyte do nauki w Tabeli 3. Wszystkie elementy *MDP* (3.4) są zdefiniowane w następujący sposób:

1. **Stan** składa się z prędkości oraz pozycji bezwzględnej wszystkich pojazdów w sieci.

$$\left(\frac{v}{v_{max}}, \frac{v_l - v}{v_{max}}, \frac{(s_l - s) \bmod s}{s_{max}}\right), \tag{29}$$

   gdzie v to prędkość pojazdu, v_{max} prędkość maksymalna, v_l prędkość pojazdu poprzedzającego, a s oraz s_l to pozycja bezwzględna pojazdu autonomicznego, oraz poprzedzającego.

2. **Akcja** to liczba zmiennoprzecinkowa z przedziału $[a_{min}, a_{max}]$, oznaczająca nadawane przyspieszenie pojazdu.

3. **Nagroda** jest wyrażona poniższym wzorem:

$$r = \alpha * \bar{v} + \gamma \min(0, \beta - \|a\|_1), \tag{30}$$

   gdzie a to przyspieszenie nadawane pojazdowi autonomicznemu, α, β, γ to parametry, a $\bar{v}$ jest średnią prędkością wszystkich pojazdów. Nagradzane są średnio duże prędkości natomiast karane przyspieszenia pojazdu *RL*.

Parametr	Wartość
v_{max}	15
s_{max}	270
a_{min}	1
a_{max}	1
α	0.2
β	0
γ	4

Tabela 2. Parametry użyte w symulacjach na drodze kołowej.

Parametr	Wartość
γ	0.999
MLP	$[3, 3]$
Funkcja aktywacji	tanh
λ	0.97
α	$5e-05$

Tabela 3. Parametry algorytmu PPO.

6.2 Ósemka

W porównaniu do poprzedniego scenariusza, sieć przypominająca ósemkę (Rysunek 4) jest bardziej wymagająca. Jest to sieć zamknięta z jednopasmowym skrzyżowaniem bez sygnalizacji, które przypomina jedno z najprostszych skrzyżowań pojawiających się w codziennym życiu. Zostało ono tak skonstruowane, aby pojazdy w nim pozostające, przejeżdżały wielokrotnie poprzez wjazdy oraz wyjazdy w kształcie pierścienia. Oczywistym wąskim gardłem jest tutaj skrzyżowanie na którym zatrzymują się pojazdy - im jest ich więcej, tym większy jest korek. Poprzez większe skomplikowanie całej sieci, ustawienie parametrów środowiska nie jest oczywiste, a w szczególności problematyczna jest funkcja generująca nagrodę. Dlatego w pierwszej kolejności, została ona dobrana na podstawie największej średniej prędkości wszystkich pojazdów w sieci. Zbadane zostało pięć następujących propozycji:

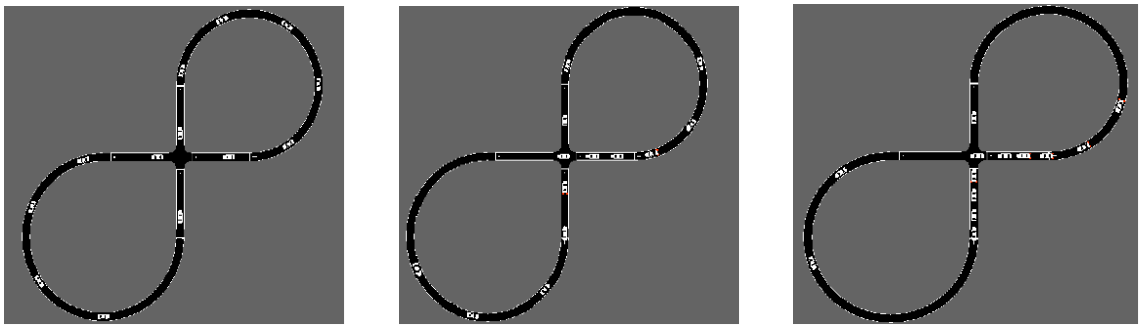
$$
\begin{aligned}
r_0 &= \frac{\max(\|v_t\|_2 - \|v_t - v\|_2, 0)}{\|v_t\|_2}, \\
r_1 &= \bar{v}, \\
r_2 &= \frac{n - \|\frac{v_{max}-v}{v_{max}}\|_2}{n}, \\
r_3 &= \frac{\|\frac{v_{max}-v}{v_{max}}\|_1}{n}, \\
r_4 &= -\alpha\|v - v_t\|_2 - \beta|v| - \gamma\|v\|_2,
\end{aligned}
\tag{31}
$$

gdzie v_t to prędkość docelowa, v prędkość wszystkich pojazdów w sieci, a v_{max} jest prędkością maksymalną. Ponadto, n jest liczbą wszystkich pojazdów w symulacji. Funkcja r_0 mierzy odchylenie prędkości pojazdów od określonej przez nas pożądanej prędkości. r_1 zwraca średnią prędkość wszystkich pojazdów w sieci, co czyni ją jedną z najbardziej uniwersalnych oraz intuicyjnych funkcji nagrody. r_2 zachęca pojazdy do minimalizacji całkowitego opóźnienia na drodze, a r_3 uśrednia to opóźnienie. r_4 jest propozycją autorów pracy [26], w której badają nieco prostszy scenariusz. Ponadto, wszystkie funkcje w naturalny sposób karzą za kolizje, co przekłada się na bezpieczną, bezwypadkową jazdę. W Tabeli 4 są przedstawione użyte parametry do symulacji, a w Tabeli 5 parametry potrzebne do nauki agentów. Elementy *MDP* (3.4) są zdefiniowane w następujący sposób:

1. **Stanem** jest wektor $(v_1, .., v_n, x_1, ..., x_n) \in R^{2n}$, który opisuje aktualną pozycję oraz prędkość wszystkich pojazdów w sieci, gdzie v_i to prędkość i-tego pojazdu, x_i jednowymiarowa reprezentacja pozycji pojazdu w sieci, a n to liczba pojazdów w sieci.

2. **Akcje** to wektor $(a_1, ..., a_n) \in [a_{min}, a_{max}]^{2n}$ mówiący o nadaniu przyspieszenia dla każdego pojazdu autonomicznego w każdej jednostce symulacji.

3. **Nagroda** wybrana z (31) dająca największą prędkość średnią wszystkich pojazdów w sieci.



Rysunek 4. Symulacja ósemki

Scenariusz jest przeprowadzony za pomocą *HDV* (z ang. *Human Driven Vehicles*) oznaczone kolorem białym. Bez świateł drogowych, ludzkie zachowania podczas jazdy skutkują niską wydajnością. Na początku wszystkie pojazdy ustawione są równomiernie. Po chwili, zaczynają się one krzyżować, a przez nieskoordynowane działanie, rozpoczynają hamowanie przez co korkują skrzyżowanie. Po kilkudziesięciu sekundach, jest ono nie do odratowania przez mechaniczne zachowanie kierowców.

Parametr	**Wartość**
v_{max}	30
v_t	20
a_{min}	3
a_{max}	3
n	14
α	10
β	1
γ	0.1

Tabela 4. Parametry użyte w symulacjach na drodze ósemkowej.

Parametr	**Wartość**
γ	0.999
MLP	$[256, 256]$
Funkcja aktywacji	tanh
λ	0.97
α	$5e-05$

Tabela 5. Parametry algorytmu PPO.

6.3 Cukierek

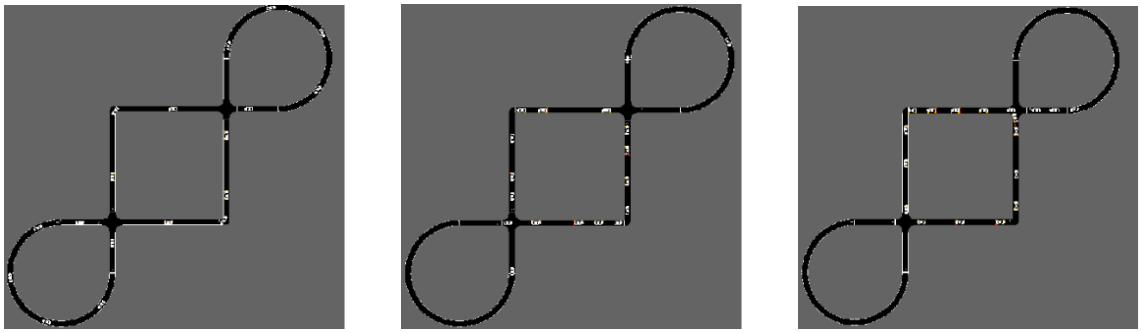
Sieć w kształcie cukierka (Rysunek 5) jest najbardziej rozbudowanym scenariuszem spośród opisywanych. Tak jak ósemka, ma dwie zapętlone drogi w rogach, jednak po środku jest jeden duży kwadrat, który je łączy. Pojazdy teraz oprócz dostosowania odpowiedniej prędkości do aktualnej sytuacji, muszą zdecydować na skrzyżowaniu, czy chcą zmienić kierunek jazdy. *HVs* skręcają z pewnym ustalonym prawdopodobieństwem, a *CAVs* wybierają drogę zgodnie z nauczoną strategią jazdy. Parametry potrzebne do symulacji są pokazane w Tabeli 6, a do nauki w Tabeli 7. Elementy *MDP* (3.4) są zdefiniowane w następujący sposób:

1. **Stanem** jest wektor $(v_1, ..., v_n, x_1, ..., x_n) \in R^{2n}$, który opisuje aktualną pozycję oraz prędkość wszystkich pojazdów w sieci, gdzie v_i to prędkość i-tego pojazdu, x_i jednowymiarowa reprezentacja pozycji pojazdu w sieci, a n to liczba pojazdów w sieci.

2. **Akcje** to wektor $(a_1, ..., a_n, y_1, ..., y_n)$, gdzie $a_i \in [a_{min}, a_{max}]$ to przyspieszenie pojazdy, a y_i jest liczbą 0 lub 1, która oznacza odpowiednio brak akcji oraz zmianę drogi jazdy.

3. **Nagroda** dana jest wzorem:

$$r = \frac{\max(\|v_t\|_2 - \|v_t - v\|_2, 0)}{\|v_t\|_2}$$



(a) (b) (c)

Rysunek 5. Symulacja 18 *HVs* na drodze przypominającej cukierka

Sieć jest rozwinięciem ósemki, z dodatkowym utrudnieniem - został dodany kwadrat między dwoma zamknięciami dróg. Intuicyjnie dostrzegamy możliwość poprawy ruchu w sieci o ile samochody nie będą zmieniały swojej drogi. Jednak każdy pojazd zmienia drogę z ustalonym prawdopodobieństwem, co niszczy cały przepływ. Decyzje podejmowane są indywidualnie, tak aby jak najlepiej odwzorować zachowania ludzi w rzeczywistym świecie. Niestety, skrzyżowania bez świateł skutkują częstymi wypadkami z udziałem *HVs*.

Parametr	**Wartość**
v_t	20
a_{min}	3
a_{max}	3
n	18

Tabela 6. Parametry użyte w symulacjach na drodze cukierkowej.

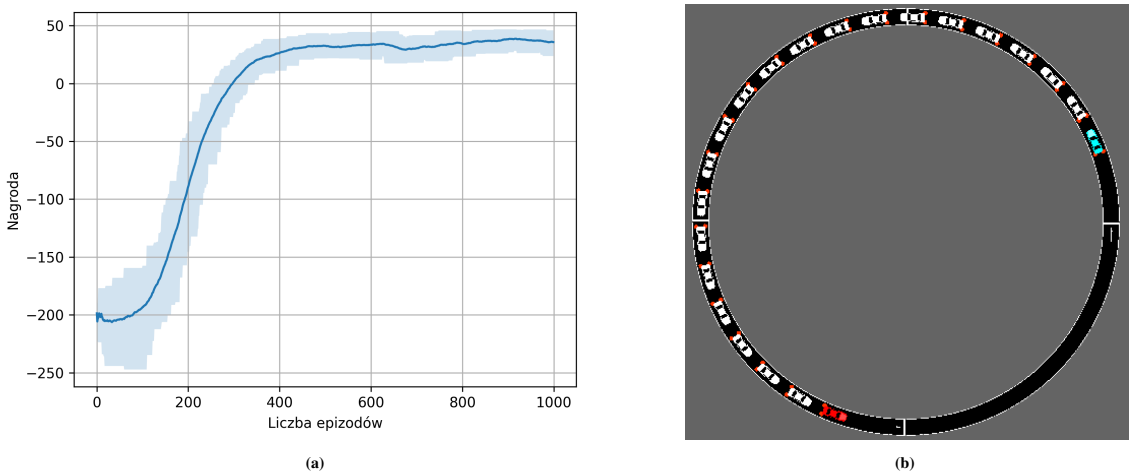
Parametr	**Wartość**
γ	0.99
MLP	$[256, 256, 256]$
Funkcja aktywacji	tanh
λ	0.95
α	$5e-05$

Tabela 7. Parametry algorytmu PPO.

7 Wyniki

7.1 Pierścień

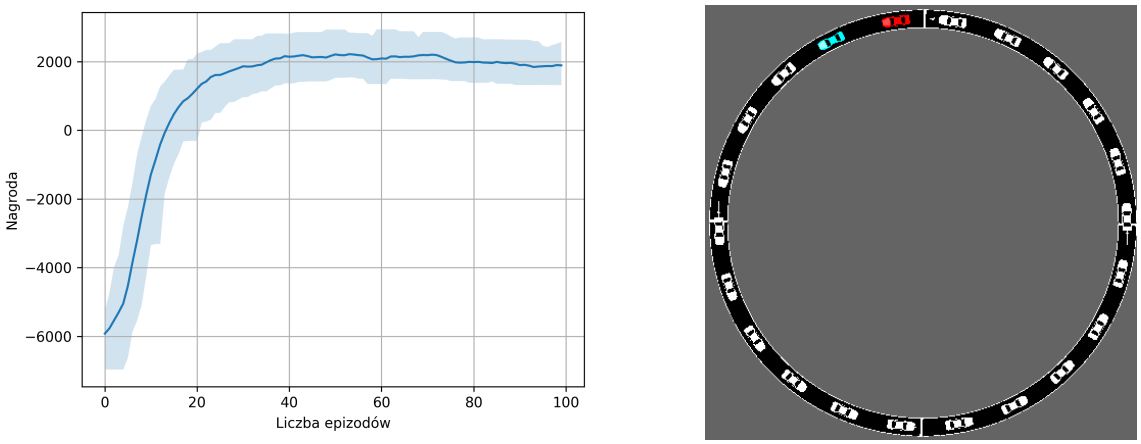
Na Rysunku 6 została opisana próba odtworzenia wyników zaproponowanych przez autorów *FLOW* [45]. Niestety parametry dobrane przez zespół, nie są w stanie nauczyć poprawnego zachowania agenta. Dopiero po poprawieniu parametrów algorytmu *PPO* agentowi udało się ustabilizować sieć (Rysunek 7). Wszystkie auta podążają za sobą mniej więcej w równych odstępach oraz mają prędkość równą średniej. Najbardziej satysfakcjonującym jest fakt, iż wyniki pokrywają się z pracą, która była przeprowadzana na realnie poruszających się samochodach [34]. Szczegółowy opis wyników został zamieszczony pod Rysunkiem 8.



(a) (b)

Rysunek 6. Odtworzenie wyników autorów *FLOW* [45]

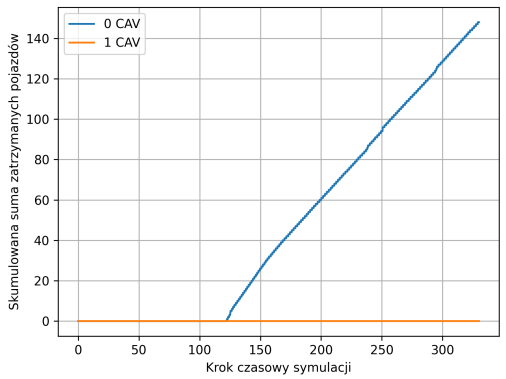
Jeden z 22 pojazdów poruszających się na drodze kołowej, zostaje zastąpiony jednym pojazdem autonomicznym zaznaczonym kolorem czerwonym. Na początku symulacji pojazdy są równomiernie rozmieszczone w sieci z początkową prędkością równą $0m/s$. Tak jak w bazowym eksperymencie [35], pojazdy symulujące zachowania ludzkie, są kontrolowane przez model *IDM*. Widzimy, że około przy 400 iteracji, algorytm stabilizuje się. Niestety, zaproponowane parametry w instrukcji nie były w stanie rozwiązać problemu korków. Pojazd *RL* zatrzymuje się w miejscu, co jednoznacznie nam mówi, że nie rozumie on środowiska w którym się znajduje.



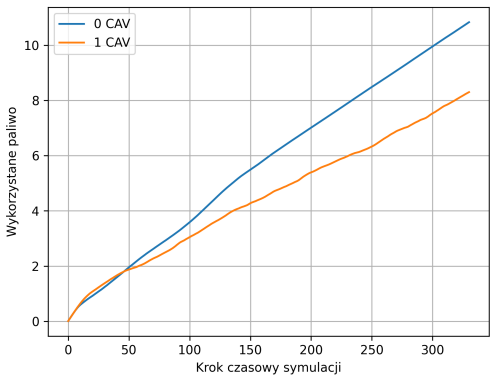
(a) (b)

Rysunek 7. Agent stabilizujący ruch na drodze kołowej

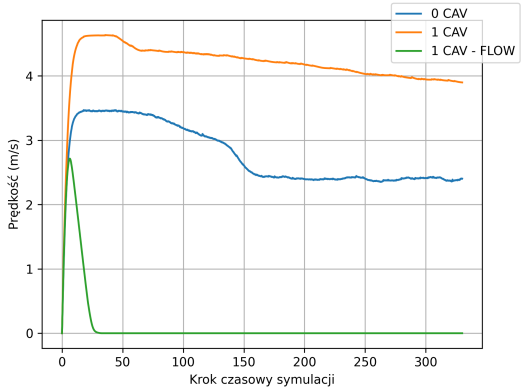
Pojazdem czerwonym jest agent *RL*, seledynowym jest pojazd obserwowany, którego parametry są użyte do obliczenia elementów *MDP*, a kolorem białym są *HVs*. Już przy 40 iteracji agent jest w stanie ustabilizować uczenie się. Po ustawieniu parametrów pokazanych w Tabeli 3, pojazdy na drodze kołowej poruszają się z jednostajną prędkością.



(a)



(b)



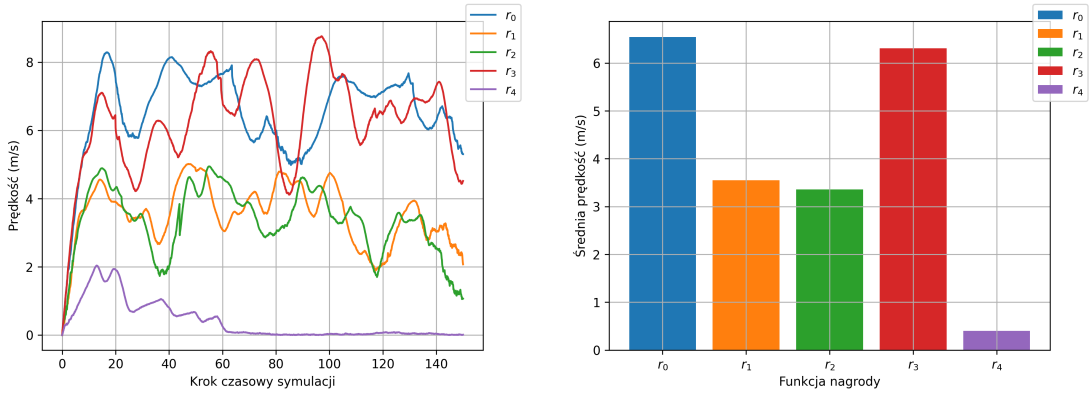
(c)

Rysunek 8. Porównanie wyników na drodze kołowej

Wykres przedstawia benefity dodania jednego pojazdu autonomicznego. Potrafił on ustabilizować całą sieć na tyle, że podczas symulacji żaden pojazd się nie zatrzymał. Zatrzymany pojazd jest definiowany prędkością mniejszą od $0.01 m/s$. Symulacja w poradniku niestety zatrzymuje wszystkie pojazdy w sieci, jednak po przeanalizowaniu parametrów i ulepszeniu procesu uczenia się, pojazd autonomiczny był w stanie przyspieszyć ruch na drodze o 51 procent, zaś sumaryczne zużycie paliwa jest mniejsze o 30 procent. Wyniki pokrywają się z eksperymentem, który został przeprowadzony w rzeczywistości [34], co udowadnia poprawność symulacji oraz fakt, że nawet jeden pojazd autonomiczny jest w stanie usprawić ruch w całej sieci.

7.2 Ósemka

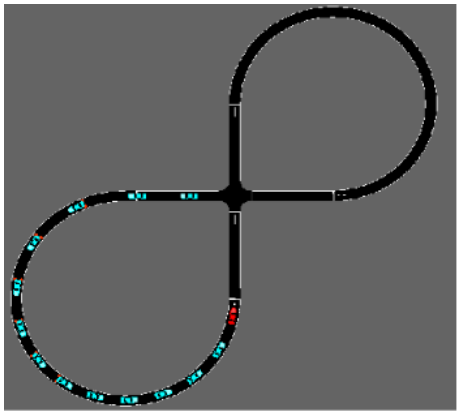
Funkcja nagrody, która będzie używana w reszcie symulacji została zbadana na Rysunku 9. Ostatecznie najbardziej satysfakcjonującą okazała się funkcja r_0. Rysunek 10 przedstawia symulację z jednym autonomicznym pojazdem, który osiąga bardzo podobne wyniki jak 14 *CAVs* pokazanych na Rysunku 11. Jednym z najbardziej ciekawych wyników jest utworzenie się całkowicie nowej strategii przy zamianie przestrzeni akcji na jedynie trzy wybory - przyspieszenie, brak akcji oraz hamowane pojazdu (Rysunek 12) Wszystkie wyniki zostały przeanalizowane na Rysunku 13.



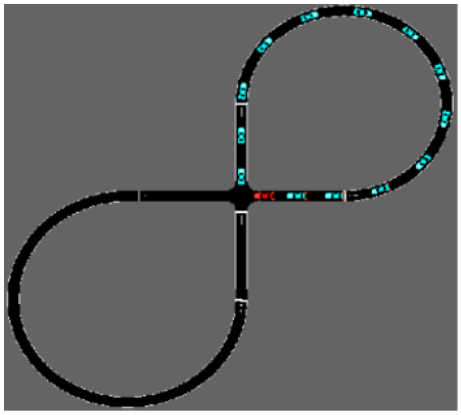
(a) (b)

Rysunek 9. Analiza zaproponowanych funkcji wyliczających nagrodę (31)

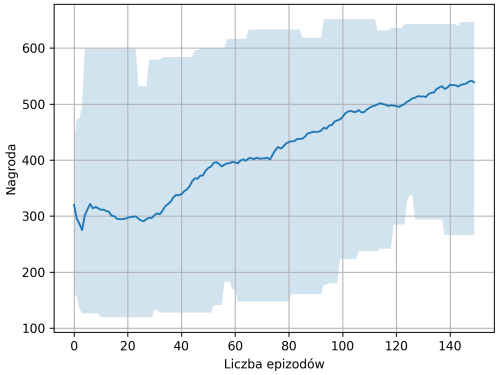
Analiza została przeprowadzona na najbardziej wymagającym eksperymencie w którym zostało użyte 14 *AV*, co pozwoliło wydobyć najbardziej miarodajne wyniki. Najlepszą funkcją okazała się r_3, która oblicza ... Jednak jest ona zaledwie lepsza o 3 procent, co można uznać za granicę błędu. W przypadku dwóch porównywalnych wyników, okazuje się ważniejszym wyznacznikiem większa stabilność sieci, którą jak widać gwarantuje funkcja r_0 i ona została wybrana ostatecznie do wyuczenia agentów. Średnia prędkość (r_1)niestety przynosi za mało informacji o tym co się dzieje aktualnie w środowisku, jak i również r_2. Najgorszym okazała się ... (r_4), jednak jest ona bardzo podatna na parametry, które trzeba odpowiednio dobierać do sieci.



(a)



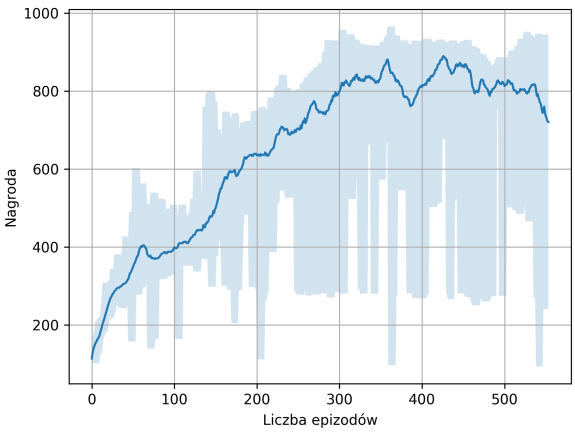
(b)



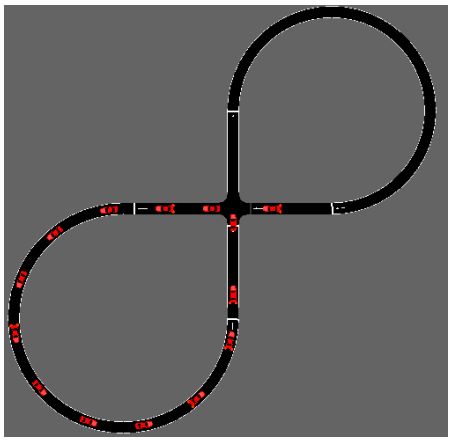
(c)

Rysunek 10. Eksperyment z udziałem 1 *AV*

Czerwonym kolorem został zaznaczony agent *RL*, a seledynowym pojazdy imitujące ludzi. Algorytm ustabilizował się mniej więcej w okolicach 130 epizodu. Wyuczona strategia polegała na zebraniu tzw. *węża* złożonego z pojazdów i kierowaniu nimi w ten sposób, aby nie mogły się spotkać ze sobą na skrzyżowaniu. Kiedy pojazd autonomiczny dojeżdżał do skrzyżowania, ostatni *HV* zjeżdżał z niego. Pozwoliło to osiągnąć bezwypadkowy scenariusz, jednak pojazdy nie mogły wykorzystać w pełni potencjału tej sieci, ponieważ lider blokował ich prędkość.



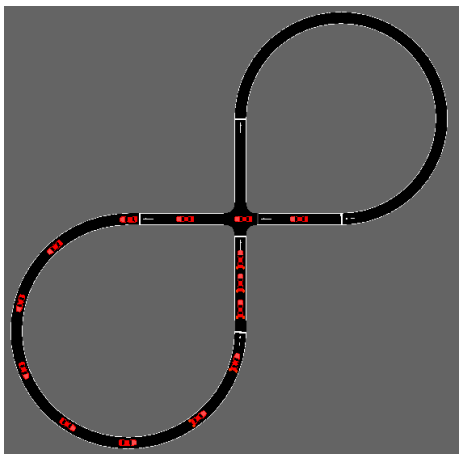
(a)



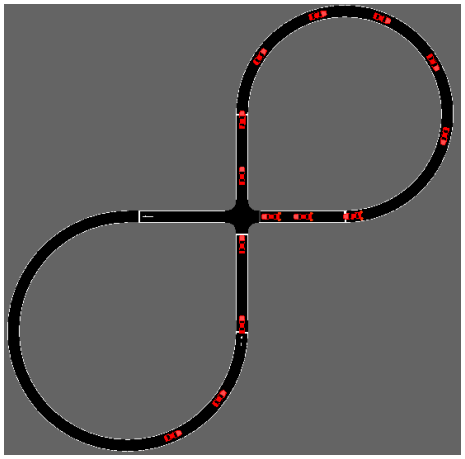
(b)

Rysunek 11. Eksperyment z udziałem 14 *AVs*

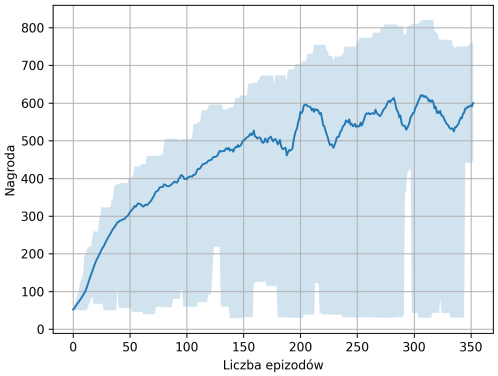
Algorytm osiąga sufit przy 300 epizodzie. Oczekiwanym wynikiem było przecinanie się po kolei pojazdów przy wysokiej prędkości, jednak scenariusz okazał się za trudny. Podczas oglądania symulacji można było zobaczyć, że *CAVs* są bardzo blisko znalezienia strategii optymalnej, jednak wystarczy minimalnie za duża prędkość lub za ostre hamowanie, aby zatrzymać wszystkie pojazdy. Niektórym pojazdom raz na jakiś czas, udawało się bezpiecznie przejechać pomiędzy poruszające się inne pojazdy.



(a)



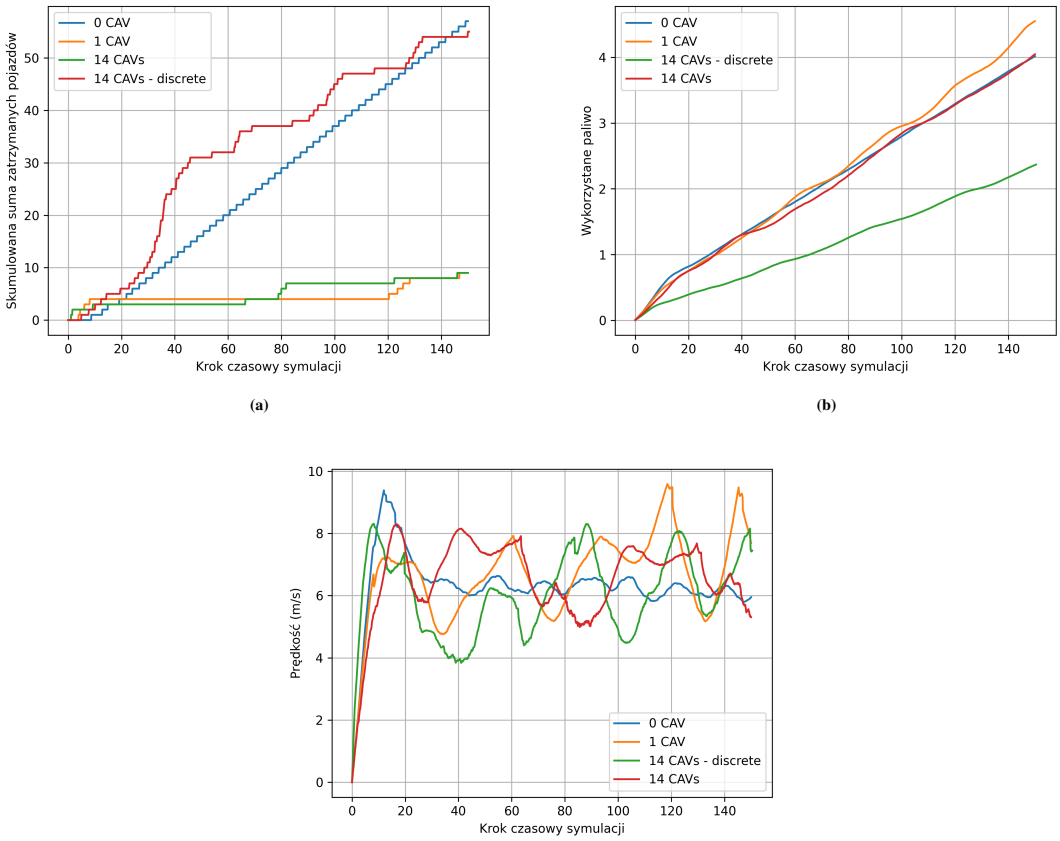
(b)



(c)

Rysunek 12. Symulacja toru ósemkowego z dyskretnym zbiorem akcji

Zwężenie zbioru możliwych akcji do wykonania do 3, czyli hamowanie z prędkością $3m/s^2$, brak akcji lub nadanie prędkości $1m/s^2$ tworzy całkowicie inną strategię dla agenta. Jeden pojazd zatrzymuje się przed skrzyżowaniem, dopóki nie przejadą wszystkie pojazdy. Prowadzi to do tworzenia się zatoru drogowego na jednej stronie skrzyżowania, zamiast na dwóch tak jak jest to w przypadku 14 *HVs*. Te akcje nie stabilizują sieci, jednak pokazują jak ogromną korelacje ma wybór strategii ze zdefiniowanymi elementami *MDP*.



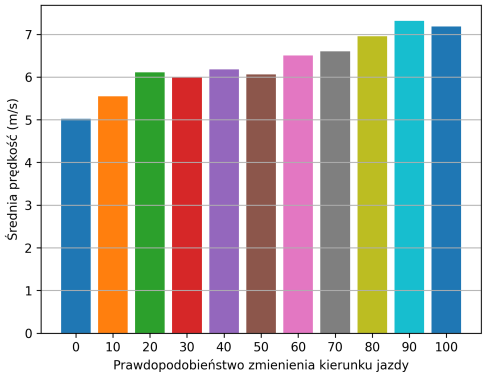
Rysunek 13. Wyniki wytrenowanych agentów na sieci przypominającej ósemkę

Zgodnie z przewidywaniami, ludzie bardzo często zatrzymują swoje pojazdy, czyli osiągają prędkość mniejszą od $1m/s$. Podobna ilość występuje w scenariuszu z dyskretnymi akcjami, ponieważ agent zbiera pojazdy przy skrzyżowaniu i wstrzymuje ich ruch. Dobry, lecz nie idealny wynik reprezentują symulację z 1 oraz 14 CAVs. Średnia prędkość pojazdów z dyskretną przestrzeni akcji oscyluje wokół $6m/s$, a inne scenariusze mają średnio 10 procent wyższą prędkość na przestrzeni całej symulacji. Mają one też podobne wykorzystanie paliwa, które jest znacznie mniejsze w dyskretnym przypadku.

7.3 Cukierek

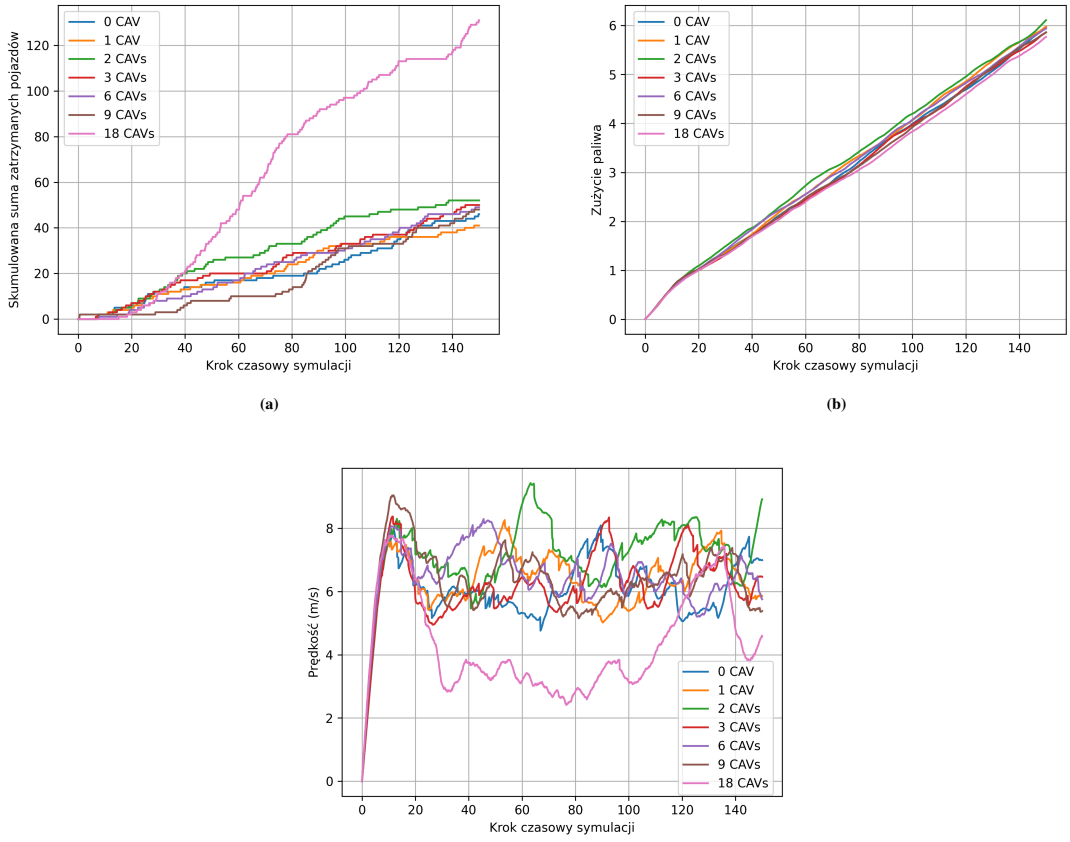
Zanim agenci zostali wytrenowani, trzeba było zbadać najważniejszą cechę sieci cukierkowej, a mianowicie jak bardzo zmiana drogi wpływa na prędkość w sieci. Okazało się, że im częściej pojazdy zmieniają drogę, tym większa jest średnia prędkość. Rysunek 14 przedstawia dokładną analizę tego problemu.

Dla porównania zostało przeprowadzonych 7 różnych eksperymentów, w którym bierze udział różna liczba *CAVs*. Zostały one dodane do sieci w ten sposób, aby były równomiernie rozłożone po całej sieci. Okazało się, że liczba pojazdów autonomicznych koreluje z liczbą wypadków w symulacji. Bardzo ciężko jest zdobyć próbkę statystyczną dla eksperymentów o mniejszej liczbie pojazdów autonomicznych, ponieważ *HVs* powodują sporą ilość wypadków, co jest równoznaczne z końcem symulacji. Przy sieci pełnej *CAVs*, jeżdżą one praktycznie bezkolizyjnie, co prowadzi do zaskakującego wyniku. Pomimo najgorszych parametrów w sieci są one najbezpieczniejszym rozwiązaniem. Wyniki przeprowadzonego eksperymentu pokazano na Rysunku 15.



Rysunek 14. Analiza wpływu zmiany kierunku ruchu pojazdów na średnią prędkość sieci

Podczas symulacji, dowolny pojazd *HV* lub *AV* może mieć wpływ na prędkość z jaką się porusza oraz może wybrać jedną z dwóch możliwych dróg na każdym skrzyżowaniu. Ze względu na większą złożoność sieci, potrzeba było przeanalizować jak bardzo zmiana kierunków ruchu wpływa na prędkość w niej. Jeśli pojazdy zostają na tej samej drodze, prędkość jest znacznie niższa niż w pozostałych przypadkach. Zdecydowanie można powiedzieć, że jeżeli pojazdy nie wjeżdżają na pierścień, prędkość ta jest wyższa o około 40 procent. Dla prawdopodobieństw w granicach 20-70 procent wyniki są bardzo podobne, a więc w praktyce te eksperymenty są równoważne. Dzieje się tak prawdopodobnie dlatego, że pojazdy nie są w stanie przejeżdżać naprzemiennie, tak jak zostało to pokazane w Rozdziale 7.2.



Rysunek 15. Wyniki wytrenowanych agentów na sieci przypominającej cukierka

Prawdopodobieństwo zmienienia drogi przez *HVs* zostało ustawione na 50 procent podczas trwania każdego eksperymentu. Wyraźnie widać, że najgorzej radzi sobie 18 *AVs*. Samochody bardzo często się zatrzymują, co jest identyfikowane z prędkością poniżej $1m/s$. Oczywiście, bezpośrednio przekłada się to na średnią prędkość wszystkich pojazdów w sieci, która jest także najniższa. We wszystkich innych przypadkach, parametry są niemal identyczne. *CAVs* nie poprawiają znacznie parametrów sieci, a jeśli są jakieś odchylenia, są to granice błędu. Pojazdy autonomiczne nie były w stanie poprawić żadnego ze sprawdzonych parametrów na wykresach, jednakże poprawiły one bezpieczeństwo, co jest o wiele bardziej ważniejszym czynnikiem. Potrafią one jeździć co najmniej tak dobrze jak ludzie nie powodując wypadków.

8 Podsumowanie

Symulacje są coraz częściej wykorzystywane do kontroli jazdy autonomicznej jako doskonała szansa na ocenę potencjału różnorodnych algorytmów uczenia ze wzmocnieniem. Testowanie w izolowanym środowisku jest skuteczne, a co najważniejsze bezpieczne. Programy starają się coraz bardziej odwzorować rzeczywiste doświadczenia, jednak jest to prawie niemożliwe do zrealizowania. W rzeczywistości są to bardzo odmienne scenariusze z różnymi czynnikami, które zakłócają symulacje takie jak nagłe wejście pieszego na drogę lub nierozważni kierowcy, którzy naruszają zasady ruchu drogowego. Dlatego symulacje opisane w pracy nie mogły pokryć rzeczywistych scenariuszy, a tylko ich idealne odwzorowanie.

CAVs mogą poprawić niektóre parametry sieci poprzez działanie jako jeden organizm. Istnieje szansa, że pojazdy znajdą optymalną strategie dla siebie, co pozwoli zmaksymalizować prędkość, zminimalizować zużycie paliwa, a także poprawić bezpieczeństwo na drogach. Eksperymenty pokazały, że nawet dodanie niewielkiej liczby autonomicznych pojazdów może zmniejszyć liczbę korków oraz zniwelować skutki nieporzadanych akcji u ludzi. W przypadku gdzie pojazdy nie poprawiły średniej prędkości, spowodowały zmniejszenie liczby wypadków, co było zaskakującym, a jednocześnie bardzo obiecującym rezultatem.

Wciąż istnieje wiele otwartych pytań dla wykonanych eksperymentów, przede wszystkim scharakteryzowanie optymalnego rozwiązania. Nie udało się dobrać takich parametrów, które pozwolą na zmaksymalizowanie prędkości. Co więcej, ze względu na deficyt sprzętowy, symulacje nie wykorzystywały w pełni potencjału biblioteki *Ray*, co z pewnością polepszyłoby cały proces uczenia się. Większość eksperymentów trwało kilkanaście godzin, a przeskalowanie tych problemów na świat rzeczywisty mogłoby wymagać o wiele większej mocy obliczeniowej. Kolejną problematyczną częścią było dobranie odpowiednich parametrów algorytmu *PPO*. Metodą prób i błędów charakteryzowane były także elementy procesu decyzyjnego Markowa. Nie da się przeszukać całej przestrzeni parametrów, jednak w żaden sposób nie można stwierdzić, czy ich wybór jest bliski optymalnego. Jeśli chodzi o większą ilość autonomicznych pojazdów, to nie została poprawiona dla nich wydajność sieci prawdopodobnie dlatego, że scenariusze wieloagentowe są bardziej skomplikowane do wyuczenia.

Niewątpliwie uczenie ze wzmocnieniem dla pojazdów autonomicznych w ruchu mieszanym jest obszernym tematem do eksploracji. Jest to dziedzina relatywnie młoda, więc pojawiają się ciągle nowe badania i pomysły usprawniające istniejące metody. Aktualna popularność sztucznej inteligencji sprzyja wzrostowi zainteresowania również wśród badaczy zajmujących się innymi dziedzinami niż informatyka. Tak jak zostało to podkreślone we wstępie, korzyści które niesie ze sobą wprowadzenie pojazdów autonomicznych sprawiają, iż pojawienie się ich na ulicach miast jest nieuniknione. Podstawą do ich wprowadzenia są przede wszystkim symulacje, które ocenią ich skuteczność, co zostało zbadane w niniejszej pracy.

Literatura

[1] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. “Reinforcement learning for true adaptive traffic signal control”. In: *Journal of Transportation Engineering* 129.3 (2003).

[2] Joshua Achiam. “Spinning Up in Deep Reinforcement Learning”. In: (2018).

[3] Aimsun. *Aimsun Next 22 User’s Manual*. Aimsun Next 22.0.1. Barcelona, Spain, Aug. 2022. URL: https://docs.aimsun.com/next/22.0.1/.

[4] John H Andreae and Peter M Cashin. “A learning machine with monologue”. In: *International Journal of Man-Machine Studies* 1.1 (1969), pp. 1–20.

[5] Masako Bando et al. “Dynamical model of traffic congestion and numerical simulation”. In: *Physical review E* 51.2 (1995), p. 1035.

[6] Christopher Berner et al. “Dota 2 with large scale deep reinforcement learning”. In: *arXiv preprint arXiv:1912.06680* (2019).

[7] Kent C Berridge and Morten L Kringelbach. “Affective neuroscience of pleasure: reward in humans and animals”. In: *Psychopharmacology* 199.3 (2008), pp. 457–480.

[8] Michele Bertoncello and Dominik Wee. “Ten ways autonomous driving could redefine the automotive world”. In: *McKinsey & Company* 6 (2015).

[9] Greg Brockman et al. “Openai gym”. In: *arXiv preprint arXiv:1606.01540* (2016).

[10] Alexey Dosovitskiy et al. “CARLA: An open urban driving simulator”. In: *Conference on robot learning*. PMLR. 2017, pp. 1–16.

[11] Morris R Flynn et al. “Traffic modeling-Phantom traffic jams and traveling jamitons”. In: *Traffic* 8.29 (2009), p. 2016.

[12] Laura Garcıéa Cuenca et al. “Autonomous driving in roundabout maneuvers using reinforcement learning with Q-learning”. In: *Electronics* 8.12 (2019), p. 1536.

[13] Kordziński Jarosław. “Siła motywacji - jak dopingować siebie i ludzi, z którymi pracujesz”. In: *Helion, Gliwice* (2012).

[14] Hubert Klüpfel, Tim Meyer-König, and Michael Schreckenberg. “Comparison of an evacuation exercise in a primary school to simulation results”. In: *Traffic and granular flow’01*. Springer, 2003, pp. 549–554.

[15] Adrienne LaFrance. “Self-driving cars could save 300,000 lives per decade in America”. In: *The Atlantic* 29 (2015).

[16] Matthew Lai. “Giraffe: Using deep reinforcement learning to play chess”. In: *arXiv preprint arXiv:1509.01549* (2015).

[17] Eric Liang et al. “RLlib: Abstractions for distributed reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3053–3062.

[18] Pablo Alvarez Lopez et al. “Microscopic Traffic Simulation using SUMO”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2575–2582. DOI: 10.1109/ITSC.2018.8569938.

[19] Patrick Mannion, Jim Duggan, and Enda Howley. “Parallel reinforcement learning for traffic signal control”. In: *Procedia Computer Science* 52 (2015), pp. 956–961.

[20] Marvin Minsky. “Steps toward artificial intelligence”. In: *Proceedings of the IRE* 49.1 (1961), pp. 8–30.

[21] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[22] Ana I Moreno-Monroy, Marcello Schiavina, and Paolo Veneri. “Metropolitan areas in the world. Delineation and population trends”. In: *Journal of Urban Economics* 125 (2021), p. 103242.

[23] Philipp Moritz et al. “Ray: A distributed framework for emerging {AI} applications”. In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 2018, pp. 561–577.

[24] OpenAI. *Openai five defeats dota 2 world champions*. Sept. 2020. URL: https://openai.com/blog/openai-five-defeats-dota-2-world-champions/.

[25] Marcin Paprzycki. “Agenci programowi jako metodologia tworzenia oprogramowania”. In: *Computer Science Department, Oklahoma State University, Tulsa, OK* 74106 (2003).

[26] Bile Peng et al. “Connected autonomous vehicles for improving mixed traffic efficiency in unsignalized intersections with deep reinforcement learning”. In: *Communications in transportation research* 1 (2021), p. 100017.

[27] Louis A Pipes. “An operational analysis of traffic dynamics”. In: *Journal of applied physics* 24.3 (1953), pp. 274–281.

[28] Duy Quang Tran and Sang-Hoon Bae. “Proximal policy optimization through a deep reinforcement learning framework for multiple autonomous vehicles at a non-signalized intersection”. In: *Applied Sciences* 10.16 (2020), p. 5722.

[29] Av Reuschel. “Fahrzeugbewegungen in der Kolonne”. In: *Osterreichisches Ingenieur Archiv* 4 (1950), pp. 193–215.

[30] Jackeline Rios-Torres and Andreas A Malikopoulos. “A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.5 (2016), pp. 1066–1077.

[31] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[32] John Schulman et al. “Trust region policy optimization”. In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[33] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.

[34] Raphael E Stern et al. “Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments”. In: *Transportation Research Part C: Emerging Technologies* 89 (2018), pp. 205–221.

[35] Yuki Sugiyama et al. “Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam”. In: *New journal of physics* 10.3 (2008), p. 033001.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[37] Quang-Duy Tran and Sang-Hoon Bae. “An efficiency enhancing methodology for multiple autonomous vehicles in an Urban network adopting deep reinforcement learning”. In: *Applied Sciences* 11.4 (2021), p. 1514.

[38] Martin Treiber and Arne Kesting. “Traffic flow dynamics”. In: *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (2013), pp. 983–1000.

[39] Joseph Treiterer and Jeffrey Myers. “The hysteresis phenomenon in traffic flow”. In: *Transportation and traffic theory* 6 (1974), pp. 13–38.

[40] Ali Y Ungoren and Huei Peng. “An adaptive lateral preview driver model”. In: *Vehicle system dynamics* 43.4 (2005), pp. 245–259.

[41] Eugene Vinitsky et al. “Benchmarks for reinforcement learning in mixed-autonomy traffic”. In: *Conference on robot learning*. PMLR. 2018, pp. 399–409.

[42] Zia Wadud, Don MacKenzie, and Paul Leiby. “Help or hindrance? The travel, energy and carbon impacts of highly automated vehicles”. In: *Transportation Research Part A: Policy and Practice* 86 (2016), pp. 1–18.

[43] Christopher John Cornish Hellaby Watkins. “Learning from delayed rewards”. In: (1989).

[44] Cathy Wu et al. “Emergent behaviors in mixed-autonomy traffic”. In: *Conference on Robot Learning*. PMLR. 2017, pp. 398–407.

[45] Cathy Wu et al. “Flow: A modular learning framework for mixed autonomy traffic”. In: *IEEE Transactions on Robotics* (2021).

[46] Bernhard Wymann et al. “Torcs, the open racing car simulator”. In: *Software available at http://torcs. sourceforge. net* 4.6 (2000), p. 2.