Wyższa Szkoła Zarządzania i Bankowości

w Krakowie

***PRACA INŻYNIERSKA***

***Robert Smoter***

***Symulacja ruchu drogowego z zastosowaniem algorytmów optymalizacji sterowania sygnalizacją świetlną.***

PROMOTOR

**dr hab. inż. Rafał Dreżewski**

KRAKÓW 2025

ITS (Intelligent transportation system)  
RL

Actor-Critic (A2C),

SUMO (Simulation of Urban Mobility)

Deep Neural Network, DNN, sieci głebokie

δ, Δ – delta

θ,w – theta, parametry sieci neuronowej (tensora)

1. Wstęp

Ruch drogowy odgrywa kluczową rolę w funkcjonowaniu wysoko zurbanizowanych społeczności, stanowiąc fundament ich gospodarki i życia społecznego. Dynamiczny wzrost liczby pojazdów wywiera ciągłą presję na istniejącą infrastrukturę transportową. Kongestia drogowa generuje wymierne straty finansowe, przyczynia się do zwiększonej emisji spalin, pogarsza jakość środowiska. W sposób pośredni utrudnia i obniżają poziom życia mieszkańców aglomeracji. Zatory drogowe wpływają na czas reakcji służb ratunkowych (straż pożarna, policja, służba zdrowia). Wraz ze wzrostem obciążenia infrastruktury drogowej, rośnie zapotrzebowanie na efektywne metody kontroli ruchu. Ponieważ fizyczna rozbudowy dróg, jest bardzo kosztowna, a często niemożliwa, jednym z kluczowych narzędzi poprawy dynamiki ruchu są sygnalizatory świetlne a ich optymalizacja jest kluczowa dla minimalizowania opóźnień drogowych.

Nowoczesne systemy transportowe (ITS), oferują szereg usprawnień podnoszących płynność ruchu drogowego w porównaniu do systemów statycznych, nie uwzględniających dynamicznie zmieniających się warunków środowiska. Systemy takie jak SCATS, SCOOT czy RHODES, pozwalają na adaptacyjne dostosowywanie cykli sygnałów do bieżących warunków drogowych. Mimo ich skuteczności, wciąż istnieje przestrzeń do ich udoskonaleń. W tym kontekście, modele uczenia maszynowego mogą odegrać kluczową rolę w dalszym rozwoju tych systemów.

Nagroda Nobla z dziedzinie fizyki w 2024 roku, jest dowodem, że badania nad algorytmami sztucznej inteligencji pozostają w centrum zainteresowana świata nauki. John J. Hopfield i Geoffrey E. Hinton otrzymali to najwyższe naukowe wyróżnienie za „*fundamentalne odkrycia i wynalazki umożliwiające uczenie maszynowe przy użyciu sztucznych sieci neuronowych*”[[1]](#footnote-1). Ich prace przyczyniły się do stworzenia mechanizmu wstecznej propagacja błędów, co dało impuls do rozwinięcie neuronowych sieci wielowarstwowych, które stały się podwaliną współczesnych systemów uczenia maszynowego.

Sukces finansowy takich projektów jak CHAT GPT,[[2]](#footnote-2) AlphaFold, Tesla Autopilot, powoduje, że ta dyscyplina wiedzy przeżywa kolejny renesans.

1. **Uzasadnienie wyboru tematu**

Obecnie jesteśmy świadkami rewolucji AI. Powstają nowe typy jednostek obliczeniowych TPU v6 o prędkości 1836 TOPS (Tera Operations Per Second). Rozwój technologii AI zaczyna być blokowany przez ograniczoną ilość sklasyfikowanych danych niezbędnych do trenowania modeli.  
 Systemy takie jak AlphaGo**,** opracowane przez DeepMind, uświadamiają nam, że maszyny mogą przekroczyć poziom ludzkich umiejętności. System AlphaGo Zero,[[3]](#footnote-3) osiągnął po 3 godzinach treningu mistrzowski poziom w grze w Go, a po 70 godzinach nauki zaproponował rozwiązania przekraczające dotychczasowe ludzkie doświadczenie.

Nowoczesne systemy sterowania ruchem, w połączeniu z technologią autonomicznych pojazdów, mogą znacząco poprawić efektywność komunikacji drogowej. Informacje generowane przez autonomiczne pojazdy oraz inne efektory, mogą stanowić źródło danych do tworzenia zaawansowanych strategii zarządzania ruchem, zwiększając płynność   
i bezpieczeństwo na drogach.

Wybór tematu pracy jest uzasadniony aktualnymi kierunkami badań w dziedzinie sztucznej inteligencji, potencjałem technologii sieci neuronowych oraz próbą wykorzystania wiedzy teoretycznej z zakresu algorytmów uczenia maszynowego w praktycznym zastosowaniu. Jako osoba zafascynowana możliwościami AI i jej potencjałem w rozwiązywaniu realnych problemów, postanowiłem skupić się na tej tematyce, aby nie tylko pogłębić swoją wiedzę teoretyczną, ale także sprawdzić się w praktycznym zastosowaniu tych technologii. Badania w dziedzinie AI cechują się dużą dynamiką, co sprawia, że jest to niezwykle ekscytujące i wymagające pole do eksploracji

1. **Cel pracy**

Celem pracy jest zbadanie, w jaki sposób algorytmy RL, takie jak aktor**-**krytyk   
**(**Actor-Critic (AC)), mogą zostać wykorzystane do sterowania sygnalizacją świetlną na obszarach o dużym natężeniu ruchu. Symulacje przeprowadzone w środowisku SUMO pozwolą na ocenę potencjału oraz efektywności takiego rozwiązania.

Zakres pracy obejmuje:

1. Przedmiotowy: Optymalizacje sterowania sygnalizacją świetlną na skrzyżowaniach przy użyciu algorytmu aktor-krytyk.   
   Analizę i interpretację wyników symulacji komputerowej.
2. Czasowy:   
   - Analizę literatury i istniejących rozwiązań semestr 5.  
   - Projektowanie i implementację algorytmu semestr 6.  
   - Testowanie i analizę wyników w środowisku symulacyjnym SUMO semestr 7.
3. Przestrzenny: Symulacje zostanie przeprowadzona w wirtualnym środowisku SUMO. Ruch drogowy będzie generowany syntetycznie, z uwzględnieniem scenariuszy, które koncentrują się na tworzeniu zatorów drogowych.

Stosowane metody

W pracy zastosowane zostaną następujące metody:

1. Analiza źródeł: Przegląd istniejących systemów sterowania ruchem oraz prac naukowych związanych z zastosowaniem sztucznej inteligencji w tej dziedzinie.
2. Modelowanie i symulacja: Implementacja algorytmu AC w środowisku SUMO, pozwalająca na symulację sterowania sygnalizacją świetlną.
3. Metody oceny efektywności: Analiza wyników symulacji, w tym pomiar opóźnień, czasu oczekiwania pojazdów, przepustowości, ilość zużytego paliwa i wyemitowanego CO2.

Opis zawartości poszczególnych rozdziałów pracy

Rozdział 4: Sterowanie ruchem świetlnym, omówienie aktualnych metod sterowania sygnalizacją świetlną.

Rozdział 5: Uczenie maszynowe; analiza literatury naukowej, opis procesów RL, AC.

Rozdział 6: Pakiet symulatora SUMO.

Rozdział 7: Przygotowanie środowiska testowego.

Rozdział 8: Zastosowanie algorytmu AC w środowisku testowym.

Rozdział 9: Analiza zgromadzonych danych.

Podsumowanie

Praca stanowi połączenie teorii algorytmów sztucznej inteligencji z praktycznym ich zastosowaniem. Celem jest implementacja algorytmu AC (aktor-krytyk) do sterowania sygnalizacją świetlną w modelowanym środowisku SUMO (Simulation of Urban MObility). Przeprowadzone symulacje będą stanowić cenne doświadczenie edukacyjne, umożliwiające zgłębienie złożonej tematyki algorytmów uczenia ze wzmacnianiem, sieci neuronowych oraz modelowania systemów transportowych. Projekt pozwoli na praktyczne zastosowanie wiedzy teoretycznej oraz rozwinięcie umiejętności w zakresie implementacji i optymalizacji systemów opartych na sztucznej inteligencji.

Do komunikacyjnych między algorytmem a symulatorem SUMO wykorzystane zostaną skrypty w języku Python, co zwiększy funkcjonalność i elastyczność całego rozwiązania.

Uzyskane wnioski z przeprowadzonych symulacji mogą stać się podstawą dla dalszego pogłębiania wiedzy w poruszanych obszarach.

1. Sterowanie ruchem świetlnym

Pierwsze zastosowanie sygnalizacji świetlnej w sterowaniu ruchem drogowym miało miejsce w 1868 roku w Londynie. Latarnie wyposażone były w lampy gazowe. Elektryczna sygnalizacja została po raz pierwszy zastosowana w 1914 roku w Cleveland.[[4]](#footnote-4) Do roku 1918 sygnalizatory były dwukolorowe, tj. wyposażone w światło czerwone i zielone. Trójkolorową, sygnalizacja zawierającą również światło żółte, zainicjowano w Londynie.

Sterowanie sygnalizacją ewoluowało od systemów stałoczasowych do systemów zmiennoczasowych. Systemy stałoczasowe działają na podstawie historycznych danych, bez sprzężenia zwrotnego, zmiennoczasowe dopasowują długość faz lub zmieniając sekwencje faz sygnalizacji do parametrów ruchu.

Nowoczesne systemy obejmują nie tylko pojedyncze skrzyżowania, ale także całe sieci drogowe. Lokalne sterowniki świetlne, działające w zdecentralizowany sposób, są wystarczające w warunkach niskiego ruchu, jednak przy większej gęstości ich wydajność jest niewystarczająca. Skuteczność lokalnych decyzji nie zawsze przekłada się na globalną optymalizację. Obecne trendy to tworzenie scentralizowanych i hierarchicznych systemów sterowania, uwzględniających współpracę między skrzyżowaniami.

Najnowsze metody, oparte na modelach predykcyjnych, nie tylko dopasowują sterowanie do bieżących warunków, ale także starają się przewidywać przyszłe sytuacje, co pozwala na lepsze planowanie i podejmowanie decyzji.

4.1 Sterowanie ruchem drogowym: szczegółowy podział systemów.

Poniżej przedstawiono podział systemów sterowania ruchem drogowym.[[5]](#footnote-5)

4.1.1 Podział według struktury sterowania:

Systemy zdecentralizowane:

Lokalne sterowniki sterują ruchem na pojedynczym skrzyżowaniu.

Brak koordynacji między skrzyżowaniami, co ogranicza ich skuteczność w zarządzaniu ruchem w dużych obszarach.

*Zastosowanie:* Mniejsze miasta lub obszary o niskim natężeniu ruchu, gdzie nie jest konieczna synchronizacja sygnalizacji.

Systemy scentralizowane:

Zarządzanie ruchem z jednego centralnego punktu, gdzie zbierane i analizowane są dane z całej sieci drogowej. Centralny system optymalizuje sygnalizację świetlną w czasie rzeczywistym, synchronizując działanie wielu skrzyżowań.

*Zalety:* Globalna optymalizacja, efektywne zarządzanie ruchem w skali całej sieci.

*Wady:* Wysokie wymagania infrastrukturalne i obliczeniowe.

Systemy hierarchiczne:

Struktura wielopoziomowa, w której każdy poziom odpowiada za inne aspekty sterowania ruchem.

*Przykład:* Lokalny poziom zarządza sygnalizacją na pojedynczych skrzyżowaniach, a poziom nadrzędny koordynuje większe obszary.

*Zastosowanie:* Rozległe sieci miejskie z różnymi poziomami złożoności ruchu.

4.1.2 Podział według rodzaju sterowania:

Stałoczasowe systemy sterowania:

Działają w oparciu o ustalone cykle sygnałów świetlnych, niezależne od aktualnego natężenia ruchu.

*Zalety:* Prostota implementacji i niski koszt wdrożenia.

*Wady:* Brak elastyczności, szczególnie w warunkach zmiennego ruchu.

Zmiennoczasowe systemy sterowania:

Systemy akomodacyjne:

Zmienna długość faz sygnalizacji bez zmiany ich kolejności. Dostosowują się do lokalnych warunków ruchu, ale nie synchronizują z innymi skrzyżowaniami.

Systemy adaptacyjne:

Dynamicznie dostosowują zarówno długość, jak i sekwencję faz sygnalizacji. Wykorzystują dane z czujników w czasie rzeczywistym, co pozwala na optymalizację w zmieniających się warunkach.

SCATS: System stosowany w Sydney, który dynamicznie dostosowuje sygnalizację w oparciu o lokalne dane ruchowe.

SCOOT: System używany w Wielkiej Brytanii, optymalizujący sygnalizację w czasie rzeczywistym na podstawie prognoz ruchu.

4.1.3 Podział według technologii i metod działania:

Systemy heurystyczne:

Wykorzystują reguły oparte na doświadczeniu lub wcześniej zdefiniowane algorytmy zarządzania ruchem.

*Zalety*: Łatwe do implementacji i zrozumienia.

*Wady:* Ograniczone możliwości optymalizacji w złożonych warunkach ruchu.

Systemy optymalizacyjne:

Stosują modele matematyczne i algorytmy optymalizacyjne, takie jak programowanie dynamiczne, algorytmy genetyczne czy metody Monte Carlo.

Mogą uwzględniać różne kryteria optymalizacji, np. minimalizację opóźnień, długości kolejek czy emisji spalin.

Systemy bazujące na uczeniu maszynowym wykorzystujące modele takie jak:

Uczenie przez wzmacnianie:  
Algorytmy uczą się optymalnych strategii sterowania na podstawie interakcji z rzeczywistym środowiskiem.

Sieci neuronowe:  
Pozwalają na analizę złożonych zależności w danych o ruchu drogowym.

4.2 Krótki opis działających systemów sterowania ruchem

1. Urban Traffic Control System (UTCS) to inicjatywa Departamentu Transportu USA, rozwijana od lat 70. XX wieku, obejmująca cztery generacje strategii sterowania ruchem drogowym:

Pierwsza generacja: Oparta na historycznych danych o ruchu, z planami sterowania zmienianymi co 15 minut.

Czwarta generacja: Oparta na aktualizacjach w czasie rzeczywistym, obliczając moment zmiany fazy sygnalizacji w każdym cyklu.

Ewolucja strategii zmierzała od statycznego do dynamicznego dostosowywania sterowania ruchem, umożliwiając lepszą reakcję na bieżące warunki ruchowe.

1. SCATS (Sydney Coordinated Adaptive Traffic System):

SCATS (Sydney Coordinated Adaptive Traffic System), opracowany przez australijskich naukowców, to adaptacyjny system sterowania ruchem zaliczany do metod trzeciej generacji. W przeciwieństwie do SCOOT, SCATS nie korzysta z modelu ruchu ani optymalizatora planów sterowania, ale wybiera najlepszy plan sterowania na podstawie bieżących warunków ruchu. Struktura systemu jest hierarchiczna, obejmując trzy poziomy: lokalne sterowniki, urządzenia regionalne oraz centralne centrum sterowania odpowiedzialne za monitorowanie całego systemu.

SCATS dostosowuje długość cyklu, split i offset sygnałów świetlnych, wykorzystując dane z detektorów. Zmiany parametrów, takie jak długość sygnału zielonego, odbywają się w małych krokach co ±6 sekund, co pozwala na dynamiczną adaptację do warunków ruchu.

SCATS jest stosowany w wielu miastach, w tym w Polsce, gdzie został wdrożony w Rzeszowie, Łodzi i Olsztynie[[6]](#footnote-6).

1. SCOOT (Split Cycle Offset Optimization Technique):  
   SCOOT (Split Cycle and Offset Optimization Technique) to metoda sterowania ruchem czwartej generacji, zaprojektowana do dynamicznej optymalizacji sygnalizacji świetlnej w oparciu o aktualne dane o ruchu. W systemie tym skrzyżowania są grupowane w pod obszary, a sterowniki w każdym pod obszarze operują na wspólnym cyklu. System dokonuje częstych, niewielkich zmian parametrów, takich jak długość sygnałów, czas trwania faz i offset, w celu minimalizacji opóźnień i zatrzymań.  
   SCOOT korzysta z trzech procedur optymalizacyjnych:

Optymalizatora splitów, który analizuje czas sygnałów czerwonych i zielonych, dostosowując ich długość w krokach co 1-4 sekundy.

Optymalizatora długości cyklu, który raz na 5 minut zmienia czas cyklu w zależności od nasycenia skrzyżowań w regionie.

Optymalizatora offsetu, pracującego raz na cykl dla każdego skrzyżowania, w celu zapewnienia płynności ruchu.

System jest szeroko stosowany w Wielkiej Brytanii i na świecie, a jego najnowsza wersja, SCOOT MC3[[7]](#footnote-7), wprowadza priorytety dla autobusów i inne udoskonalenia​.

1. RHODES (Real-Time Hierarchical Optimized Distributed Effective System):

Hierarchiczny system sterowania, który dynamicznie dostosowuje sygnalizację w czasie rzeczywistym, wykorzystując dane z czujników.  
Algorytm ten został nazwany sterowaną optymalizacją faz (COP – Controlled Optimization of Phases). Podobnie jak systemy **DYPIC PRODYN, OPAC** jest oparty na metodzie programowania dynamicznego.

1. **GASCAP, SPPORT**[[8]](#footnote-8) Sterowanie ruchem drogowym z wykorzystaniem logiki rozmytej opiera się na analizie długości kolejek i napływu ruchu, które są przekształcane na wartości przynależności do zbiorów rozmytych, takich jak Krótka, Średnia czy Długa. Decyzje sterujące, np. przedłużenie fazy zielonej, wynikają z reguł rozmytych, które uwzględniają siłę aktywacji (FS) dla każdego przypadku. Zaletą logiki rozmytej jest niski koszt obliczeniowy i zdolność lepszego odzwierciedlenia aktualnych warunków ruchu w porównaniu do metod stałoczasowych czy zmiennoczasowych. Przykładowo, długość kolejki o wartości 7 może należeć jednocześnie do zbiorów Średnia i Długa z przynależnością 0,6, co zwiększa możliwości generalizacji. Dzięki temu logika rozmyta jest skuteczną i elastyczną metodą sterowania ruchem drogowym.
2. PIACON [[9]](#footnote-9) to metoda inteligentnego sterowania ruchem drogowym, opracowana w 2008 roku przez AGH i holenderskiego producenta sterowników, wdrożona w Lubinie. Bazuje na systemach ekspertowych oraz algorytmach optymalizacyjnych i działa na trzech poziomach: lokalnym, arterialnym i sieciowym. Wykorzystując dane z detektorów ruchu, takie jak liczba pojazdów czy długość kolejek. Uwzględnia wielokryterialne podejście, analizując m.in. straty czasu, zatory i emisję zanieczyszczeń, by dynamicznie dostosowywać sygnalizację świetlną do aktualnych warunków drogowych.
3. ~~Systemy oparte na AI:~~
   * ~~DRL (Deep Reinforcement Learning): Wykorzystywane do sterowania sygnalizacją świetlną w oparciu o rzeczywiste dane ruchowe.~~
   * ~~Metody multi-agentowe: Agenci zarządzający poszczególnymi skrzyżowaniami uczą się współpracy w celu optymalizacji globalnego ruchu.~~

Metody adaptacyjnego sterowania ruchem często mają złożoną hierarchiczną budowę i wymagają skomplikowanych algorytmów o dużej złożoności czasowej. Systemy takie jak SCATS i SCOOT są rozwijane i skutecznie sterują ruchem w miejskich sieciach liczących tysiące skrzyżowań. Obecnie dąży się do tworzenia systemów zdolnych do przetwarzania dużych ilości danych w krótkim czasie uwzględniających nietypowe sytuacje takiej jak kolizje czy remonty.

Z badań i wdrożeń przeprowadzonych w różnych aglomeracjach wynika, że zastosowanie zaawansowanych systemów zarządzania ruchem jest korzystne zarówno dla kierowców, pieszych, jak i środowiska naturalnego.

Nowoczesny i wydajny system sterowania ruchem to dziś;

* krócenie czasu przejazdu,
* większa płynność ruchu,
* zwiększenie bezpieczeństwa,
* monitorowanie rejestracja i analiza ruchu,
* priorytetowanie pojazdów uprzywilejowanych i komunikacji zbiorowej,
* ograniczenie zużycia paliwa i emisji spalin,
* personalizowane, planowanie tras,
* dostęp do danych statystycznych.

1. Uczenie maszynowe

Sztuczna inteligencja (AI), uczenie maszynowe (ML) to dynamicznie rozwijające się dziedziny, które odgrywają kluczową rolę w dzisiejszym świecie technologii informatycznych. Za ojca sztucznej inteligencji i informatyki uznaje się Alana Turing, który w 1943 roku postawił fundamentalne pytanie: "Czy maszyny mogą myśleć?". Jego prace nad maszynami obliczeniowymi zapoczątkowały ideę tworzenia inteligentnych systemów informatycznych.   
Kilka lat później, w 1956 roku, John McCarthy ukuł termin "sztuczna inteligencja" podczas legendarnej konferencji w Dartmouth College, która formalnie rozpoczęła badania nad AI.

W 1959 Arthura Samuela wprowadził termin uczenie maszynowe (machine learning) w kontekście programowania komputerów zdolnych do uczenia się na podstawie danych. Samuel jest również autorem pierwszego samodzielnie uczącego się systemu, programu grającego w warcaby.[[10]](#footnote-10)

Uczenie maszynowe aktualnie dzieli się na trzy główne typy; uczenie nadzorowane, uczenie bez nadzoru oraz uczenie ze wzmocnieniem[[11]](#footnote-11). W uczeniu nadzorowanym model uczy się na danych z oznaczonymi etykietami, co pozwala na realizację zadań takich jak klasyfikacja czy regresja. W uczeniu bez nadzoru system analizuje nieoznakowane dane, odkrywając ukryte wzorce, na przykład poprzez klasteryzację lub redukcję wymiarowości. Natomiast uczenie ze wzmocnieniem polega na interakcji modelu z otoczeniem, gdzie agent uczy się podejmować decyzje optymalizujące przyszłe nagrody.

5.1. Wprowadzenie do uczenia ze wzmocnieniem (RL**)**

Uczenie ze wzmacnianiem to rodzaj technik stosowanych w systemach uczących się, w których agent podejmuje działania prowadzące do zmaksymalizowania nagrody płynącej ze środowiska, poprzez wykonywanie określonej sekwencji kroków.

Początki uczenia przez wzmacnianie sięgają lat 50. XX wieku. Są silnie zakorzenione w badaniach nad zachowaniem adaptacyjnym, dynamicznym programowaniem i Procesami Decyzyjnymi Markowa. Istnieje wiele obszarów, które są związane z uczeniem przez wzmacnianie. Najistotniejsze przedstawione są na rysunku 1.

*Rysunek 1 - Źródło: schemat pochodzi z książki* Głębokie uczenie przez wzmacnianie. Praca z chatbotami oraz robotyka, optymalizacja dyskretna i automatyzacja sieciowa w praktyce. S.31.

Podstawowy model RL (Reinforcement Learning) wykazuje liczne analogie do modeli psychologicznych z dziedziny warunkowania klasycznego. Eksperymenty przeprowadzone przez Iwana Pawłowa z psami demonstrują zdolność zwierząt do kojarzenia sygnałów środowiskowych, takich jak dźwięk dzwonka, z bodźcami nagradzającymi, np. jedzeniem. Pawłow określił ten mechanizm jako 'wzmocnienie', odnosząc się do bodźca nagradzającego, który wzmacniał pożądane zachowania psa (agenta). [[12]](#footnote-12)

Istnieje dużo algorytmów tego modelu, ale szczególną popularność zyskały obecnie 2 z nich: sieć deep-Q (deep Q-network, DQN) oraz deep deterministic policy gradient (DDPG). Oba są łatwe do wdrożenia, a jednocześnie oferują bardzo duże możliwości adaptacji do środowiska.[[13]](#footnote-13)

 Na rysunku 2 znajduje się taksometria współczesnych algorytmów RL, zaproponowana przez Josha Achiama, naukowca z OpenAI. Diagram daje pogląd na rozległość dziedziny.

Rysunek 2 Josha Achiama,OpenAI; taksonomia algorytmów we współczesnym uczeniu przez wzmacnianie (RL), https://spinningup.openai.com/en/latest/spinningup/rl\_intro2.html

5. 2 Formalne podstawy i terminologia

Głównymi elementy uczenia przez wzmacnianie są; agent (Agent) i środowisko (Enviroment), kanały interakcji — akcje (action), nagrody (reward) i stany (state).

Rysunek 1. Schemat blokowy algorytmu RL

Rysunek 3, Żródło: Schemat pochodzi z ksiązki „Reinforcement Learning: An Introduction” Second edition, in progres November 5, 2017, stron 38 [[14]](#footnote-14)

**Agent i środowisko**

Agent to podmiot, który wchodzi w interakcję ze środowiskiem w dyskretnych krokach czasowych *t,* agent znajduje się w stanie *st∈S ,* gdzie *S* jest zbiorem wszystkich możliwych stanów środowiska*.* W każdym kroku *t* agent wykonuje akcję *at∈A*, odbiera obserwację stanu *st+1* oraz otrzymuje nagrodę *rt+1∈R*, gdzie *A* jest zbiorem dostępnych akcji, a *R* zbiorem możliwych nagród.   
Środowisko reprezentuje wszystko, co otacza agenta, dostarcza mu informacji st+1 i reagując na jego działania at.

**Akcje**

Akcje to działania, jakie agent może wykonywać w środowisku, np. ruchy w grze. Decyzje podejmowane mogą być dyskretne (np. ruch w lewo) lub ciągłe (ustaw czas świecenia światła zielonego na sygnalizatorze na [10,60] s).  
Akcje są częścią **trajektorii**, czyli sekwencji stanów, akcji i nagród, którą agent generuje podczas eksploracji środowiska. Trajektoria zaczyna się od początkowego stanu i kończy się, gdy agent osiągnie stan końcowy lub gdy **epizod** zostanie przerwany po ustalonej liczbie kroków.

**Obserwacje**

Obserwacje to informacje przekazywane agentowi przez środowisko, opisują aktualny stan. Mogą być użyteczne do przewidywania przyszłych nagród.

**Nagroda**

Nagroda w uczeniu przez wzmacnianie to skalarna wartość, którą agent okresowo otrzymuje ze środowiska jako informację zwrotną o jakości swoich działań. Może być pozytywna lub negatywna, ale zawsze ma charakter lokalny, odzwierciedlając niedawne działania agenta, a nie całokształt jego sukcesów. Celem nagrody jest wzmocnienie pożądanych zachowań agenta.

Nagrody pozostają kluczowym elementem procesu uczenia, napędzającym postępy agenta.

5.3 **Procesy Decyzyjne Markowa (MDP)**

Procesy Decyzyjne Markowa (MDP) to model matematycznym używany w uczeniu przez wzmacnianie. Umożliwia formalne modelowanie środowiska oraz interakcji środowiska z agentem. Jest on rozszerzeniem *klasycznego procesu Markowa* dodając do niego terminy akcja i nagroda.

MDP można zdefiniować jako 5-eleentową krotkę:

*MDP = (S,A,P,R,γ) (wzór 1)*

gdzie:  
S: zbiór stanów środowiska,  
A: zbiór działań agenta,  
P(s′∣s,a) : prawdopodobieństwo przejścia z s do s' po wykonaniu akcji a,  
R(s,a): funkcja nagród, określająca wartość nagrody dla stanu s i akcji a,   
γ∈[0,1): współczynnik dyskontowania, który kontroluje znaczenie przyszłych nagród.

MDP opisuje, jak działania agenta wpływają na zmiany stanu środowiska oraz na otrzymywane nagrody. Kluczowe na tym etapie są dwie funkcje:

**5.4. Funkcja przejścia *P(s′|s, a)*:**

Funkcja ta definiuje prawdopodobieństwo, przejścia do stanu (s′) po wykonaniu akcji (a) w stanie (s):

(wzór 2)

Funkcja przejścia opisuje dynamikę środowiska oraz określenie wpływu działań agenta na przyszłe stany.

**5.4 Funkcja nagrody *R(s,a)*:**

Funkcja nagrody *R(s,a)* określa oczekiwaną wartość nagrody rt+1, którą agent otrzymuje po podjęciu akcji (*a*) w stanie (*s*). Jest to wartość średnia, uwzględniająca wszystkie możliwe wyniki (nagrody), jakie mogą wystąpić w przyszłości po tej decyzji.

(wzór 3)

gdzie   
 [⋅]: Operator wartości oczekiwanej, obliczający średnią ważoną wszystkich możliwych wyników.

Nagroda jest kluczowym elementem kierującym działaniami agenta, ponieważ określa, które stany i akcje są pożądane.

**5.5 Współczynnik dyskontowania nagród γ (gamma).**

Współczynnik określa, jak bardzo agent ceni przyszłe nagrody w porównaniu z bieżącymi. Jeśli γ jest bliskie 0, agent skupia się na natychmiastowych nagrodach, ignorując długoterminowe konsekwencje. Gdy γ jest bliskie 1, przyszłe nagrody są równie ważne jak bieżące, co pozwala na bardziej strategiczne podejmowanie decyzji.”

Agent wybiera akcje tak, aby zmaksymalizować skumulowaną zdyskontowaną nagrodę (G) otrzymywaną w przyszłości. Skumulowana nagroda (lub zdyskontowany zwrot) jest definiowana jako:  
[[15]](#footnote-15) (wzór 4 można pominąć)

gdzie:  
Gt: skumulowana zdyskontowana nagroda począwszy od chwili t,  
Rt+k+1R: nagroda otrzymana w kroku t+k+1t+k+1t+k+1,  
γ: współczynnik dyskontowania, który zmniejsza znaczenie nagród otrzymanych w odległej przyszłości.

**5.6 Polityka.**

Polityka (π) definiuje sposób, w jaki agent podejmuje decyzje w środowisku. Jest to funkcja określająca prawdopodobieństwo wyboru akcji (*a)* w stanie (*s)*:

(wzór 5)

Polityka określa strategię agenta, wpływając na osiąganie celu: maksymalizację skumulowanej nagrody. Polityka optymalna prowadzi do maksymalizacji oczekiwanej skumulowanej nagrody w długim horyzoncie czasowym.

Polityka może być;

- Stochastyczna: Losowy wybór akcji z przypisanymi prawdopodobieństwami, np. eksploracja środowiska.  
- Deterministyczna: Zawsze wybiera tę samą akcję w danym stanie (π(a∣s)=1).

5.7 Równania Bellmana

Równania Bellmana są wykorzystywane do **rekurencyjnego wyznaczania wartości stanu** (V(s)) lub **optymalnej polityki** (π∗(s)) w danym stanie (s). Ich uniwersalność polega na możliwości zastosowania w różnych technikach optymalizacyjnych, takich jak iteracja wartości, iteracja polityki czy Q-Learning.

**Równanie Bellmana dla wartości stanu (Vπ(s)):**

Wzór na oczekiwana suma zdyskontowanych nagród, zaczynając od stanu s i postępując zgodnie z polityką π.

(wzór 6)

gdzie

Vπ(s) - wartość stanu s przy danej polityce π.  
Ea∼π,s′∼P[.] oczekiwanie (średnia wartość) po losowych zmiennych:  
 a∼π Akcjaa jest wybierana zgodnie z polityką π(a∣s), czyli prawdopodobieństwem   
 wybrania akcji a w stanie s.  
 s′∼P : Nowy stan s′ jest losowany z rozkładu P(s′∣s,a), który opisuje przejścia między   
 stanami w środowisku.  
r(s,a) - Nagroda natychmiastowa za wykonanie akcji a w stanie s.  
γ: Współczynnik dyskontowania (0≤γ≤1).  
Vπ(s′) - Wartość stanu s′, do którego przechodzi system po wykonaniu akcji aaa.

****Równanie Bellmana dla optymalnej wartości stanu (V∗(s)):****

(wzór 7)

Określa maksymalną możliwą wartość stanu s, gdy agent działa w sposób optymalny.

W przeciwieństwie do wersji on-policy, tu dodany jest operator max⁡\maxmax, który reprezentuje wybór akcji a maksymalizującej wartość.

**Techniki wykorzystujące równania Bellmana**

****Iteracja wartości**:**

Rekurencyjnie oblicza V(s) dla wszystkich stanów, aż do zbieżności.

Po zakończeniu procesu wyznacza optymalną politykę π∗(s).

****Iteracja polityki:****

Naprzemienne kroki oceny polityki (Vπ(s)) i jej ulepszania (π′(s)).

Równania Bellmana są używane w obu etapach.

Równania Bellmana są podstawą algorytmów uczenia przez wzmacnianie, ponieważ umożliwiają propagację informacji o nagrodach w czasie i ocenę długoterminowych konsekwencji działań agenta

5.8 Algorytm Aktor-Krytyk (Actor-Critic)

Rysunek 4 A brief review of Actor Critic Methods, <https://www.youtube.com/watch?v=aODdNpihRwM>

**Algorytm aktor-krytyk** jest połączeniem algorytmów aproksymacji funkcji polityki (policy function) i funkcji wartości (value function) (Rysunek 4). W algorytmach opartych na polityce typu REINFORCE, funkcja polityki jest aktualizowana na końcu epizodu, co jest mało efektywne. Wysoka wariancja gradientu (rezultat sumowania wszystkich zdarzeń z epizodu) powoduje, że potrzeba więcej próbek (epizodów) celem stabilizacji modelu.  
 Algorytm aktor-krytyk rozwiązuje ten problem, korzystając z **metody różnicy czasowej** (ang. *Temporal Difference*). Dzięki temu uczy się przy każdym kroku, a nie tylko na końcu epizodu. (rysunek 5 przedstawia dynamikę procesu)



Rysunek 5, Timothée Carayol Deep reinforcement learning in python, https://campus.datacamp.com/courses/deep-reinforcement-learning-in-python/introduction-to-policy-gradient-methods?ex=7

Pomysł polega na wprowadzeniu agenta zbudowanego z dwóch elementów:

**Aktora** - uczy się polityki π(a∣s), która określa, jakie akcje należy podejmować w danych stanach.

**Krytyka** - Szacuje wartość stanu V(s) i ocenia, jak dobra była decyzja aktora.

**Różnica czasowa** - Krytyk oblicza błąd różnicy czasowej δt ​, który służy jako sygnał wzmocnienia do ulepszania polityki w aktorze.

, (wzór 8)

gdzie:

δt ​ to błąd różnicy czasowej (TD-error),

rt​ to nagroda natychmiastowa,

V(s) to funkcja wartości stanu,

γ to współczynnik dyskontowania.

Algorytm aktor-krytyk łączą zalety metod opartych na wartościach (redukcja wariancji dzięki krytykowi), oraz metod opartych na politykach (elastyczność w modelowaniu przestrzeni ciągłych). Na rysunku 7 widzimy dokładniej przebieg algorytmu aktor-krytyk

Rysunek 6: Ha jime Kimura, Shigenobu Kobayashi An Analysis of Actor/Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imp erfect Value Functions: <http://users.umiacs.umd.edu/~hal/courses/2016F_RL/Kimura98.pdf>

Opis formalny algorytmu uwzględniającego wykorzystanie sieci neuronowych zaczerpnięty z „Reinforcement Learning: An Introduction”[[16]](#footnote-16)

**Wejście:**

*π(a∣s,θ), różniczkowalna funkcja prawdopodobieństwa wyboru akcji a w stanie s.*

*V(s,w), różniczkowalna funkcja szacująca wartość stanu s.*

*Współczynniki uczenia: αθ>0, αw>0.*

**Inicjalizacja:**

*Parametry polityki: θ∈R.*

*Wagi funkcji wartości: w∈R.*

**ALGORYTM:**

**Pętla nieskończona (dla każdego epizodu):**

*1. Inicjalizuj* ***s*** *pierwszy stan epizodu.*

*2. I←1*

**Pętla czasowa (dopóki sss nie jest terminalny):**

*3. Wybierz akcję a∼π(⋅∣s,θ).  
4. Wykonaj akcję a, zaobserwuj nowy stan s′ i nagrodę r.  
5. Oblicz błąd TD (δ):*

*(nawiązanie do wzoru 8)*

\*(Jeśli st+1 jest stanem terminalnym, to V(s′,w)=0.

6. Zaktualizuj wagi funkcji wartości:

7. Zaktualizuj parametry polityki:

8. Zaktualizuj współczynnik wpływu I:

*I←γI*

9. Przejdź do następnego stanu:

s←st+1

5.10 Deep Learning w kontekście RL

**Uczenie głębokie (Deep Learning, DL)** to dziedzina sztucznej inteligencji, która korzysta z wielowarstwowych sieci neuronowych (rysunek 8), pozwalających na efektywne przetwarzanie i predykcje złożonych funkcji. W uczeniu przez wzmacnianie, metody DL odgrywają kluczową rolę w rozwiązywaniu problemów związanych z dużymi i złożonymi przestrzeniami stanów i akcji.Klasyczne metody, wyznaczanie polityki lub wartości, polegają na iteracyjnym wykonywaniu równań Bellmana (wzór 6,7) w celu propagacji nagród w czasie. Dzięki wykorzystaniu sieci neuronowych, takie obliczenia mogą zostać „nauczone”, co redukuje koszt obliczeniowy do jednorazowego wytrenowania modelu.



Rysunek 8 [Maximilian Pichler and Florian Hartig](https://www.uni-regensburg.de/biologie-vorklinische-medizin/theoretische-oekologie/mitarbeiter/maximilian-pichler/), Machine Learning and Deep Learning with R, [Maximilian Pichler and Florian Hartig](https://www.uni-regensburg.de/biologie-vorklinische-medizin/theoretische-oekologie/mitarbeiter/maximilian-pichler/), https://theoreticalecology.github.io/machinelearning/

W 2015 Firma Google DeepMind zaprezentowała, jak głębokie konwolucyjne sieci neuronowe (Convolutional Neural Network) mogą automatyzować ekstrakcję cech, umożliwiając RL radzenie sobie z zadaniami wymagającymi rozumienia zdarzeń w przestrzeni.[[17]](#footnote-17) Przełomowym okazało się opracowanie sieci Deep Q-Network (DQN), która łączyła Q-learning z głęboką CNN. Architektura ta pozwoliła DQN na uczenie się wartości Q(s,a) bezpośrednio z surowych danych wejściowych, takich jak piksele. DQN udowodniła swoje możliwości, ucząc się grać w 49 różnych gier Atari i osiągając lub przewyższając poziom człowieka w wielu z nich.

Na rysunku (9) pokazano ogólny schemat zastosowania sieci neuronowej (DNN) do predykcji polityki πθ, (*algorytm Policy Gradient***)** gdzie agent wchodzi w interakcję ze środowiskiem. Należy zwrócić uwagę na symbol θ będący parametrem sieci.



Rysunek 9 Reinforcement Learning with policy represented via DNN, Hongzi Mao, Mohammad Alizadeh, Ishai Menache, Srikanth Kandula; https://people.csail.mit.edu/hongzi/content/publications/DeepRM-HotNets16.pdf

1. **Pakiet SUMO** [[18]](#footnote-18)

Eclipse SUMO to darmowy, otwartoźródłowy pakiet do modelowania systemów transportu intermodalnego, w tym pojazdów drogowych, transportu publicznego oraz ruchu pieszych. Projekt został zainicjowany w 2001 roku przez pracowników Instytutu Systemów Transportowych Niemieckiego Centrum Lotnictwa i Kosmonautyki (DLR).

SUMO jest zestawem aplikacji oferując narzędzia do generowania i importowania sieci drogowych z różnych formatów, a także do tworzenia scenariuszy o dużej skali, takich jak symulacje ruchu w miastach. Symulacje w SUMO są mikroskalowe co oznacza, że każdy pojazd jest modelowany osobno, ma swoją własną trasę i porusza się indywidualnie. Scenariuszach mają możliwość wprowadzana losowości zdarzeń.

SUMO znajduje zastosowanie w badaniach nad komunikacją V2X (pojazd-pojazd i pojazd-infrastruktura). Generowane scenariusze służą do oceniania algorytmów wyboru tras, dynamicznej nawigacji i optymalizacji sygnalizacji świetlnej.

Platforma posiada modele emisji hałasu oraz zanieczyszczeń powietrza, umożliwiając ocenę ekologicznych skutków transportu. Obsługuje również wsparcie dla pojazdów autonomicznych.

Do komunikacji z SUMO w czasie rzeczywistym najczęściej wykorzystuje się interfejs TraCI (Traffic Control Interface) [[19]](#footnote-19), działający jako usługo TCP/IP. TraCI umożliwiający odczytywanie parametrów symulacji oraz inicjowanie zmieniających się parametrów środowiska.

SUMO jest popularny dzięki wszechstronności, otwartemu kodowi źródłowemu oraz wsparciu dla dużych symulacji. Dzięki API platformę można integrować z innymi narzędziami poprzez biblioteki w języku Python, C++, JavaMATLABPoczątek formularza

1. Przygotowanie środowiska testowego.

7.1 Pliki konfiguracyjne

Najważniejsze elementy modelu, do którego zaimplementuję sterowanie oświetleniem, zostały określone w kilku kluczowych plikach konfiguracyjnych:

**Plik 2x2.net.xml** jest rdzeniem modelu, stanowi mapę drogową dla symulacji ruchu.

Główny element <net> definiuje całą sieć drogową. Znajdują się w nim atrybuty takie jak opisów narożników skrzyżowań (junctionCornerDetail), maksymalne dopuszczalna prędkość skrętów (limitTurnSpeed). Dodatkowo określony jest offset sieci (netOffset) oraz granice konwersji i oryginalne granice sieci, co umożliwia właściwe pozycjonowanie i skalowanie całej symulacji.

• **Definicja dróg (krawędzi):**  
Każda droga <edge> jest opisana jako element XML, który zawiera informacje o jej funkcji oraz o krawędziach ruchu. Wewnątrz każdego elementu <edge> znajdują się elementy <lane>, które określają:

* Identyfikator pasa ruchu (id) oraz indeks pasa
* Maksymalną prędkość (speed)
* Długość pasa (length)
* Geometrię pasa (shape) – zestaw współrzędnych (x,y) opisujących krzywą drogi.

**Definicja skrzyżowań (węzłów):**  
W sieci znajdują się różne typy skrzyżowań, reprezentowane przez elementy <junction>. Każdy skrzyżowanie posiada:

* Unikalny identyfikator (id), dzięki czemu można jednoznacznie odwoływać się do danego węzła.
* Typ skrzyżowania (np. "dead\_end" dla końców dróg lub "traffic\_light" dla skrzyżowań sterowanych sygnalizacją świetlną).
* Pozycję w układzie współrzędnych (atrybuty x i y),
* Listę pasów wchodzących (incLanes) oraz wewnętrznych (intLanes)
* Dokładny kształt skrzyżowania (shape), który może być reprezentowany jako wielokąt, odzwierciedlający rzeczywiste rozmiary i kształt węzła.

• **Logika sterowania ruchem na skrzyżowaniach:**  
Skrzyżowania sterowane sygnalizacją świetlną, takie jak P4, P5, P8 i P9, są wyposażone w rozbudowaną sterowania. Dla każdego z nich zdefiniowany jest element <tlLogic>, który zawiera:

* Identyfikator sygnalizacji (id), przypisany do konkretnego skrzyżowania.

• **Połączenia między elementami sieci:**  
Plik zawiera także elementy <connection>, które definiują, w jaki sposób pasy ruchu łączą się pomiędzy skrzyżowaniami. Te połączenia określają kierunki skrętów (np. skręt w lewo, w prawo lub jazda prosto) oraz warunki przejazdu przez węzeł, co jest kluczowe dla realistycznej symulacji ruchu.



• Plik 2x2.rou.xml definiuje przepływy pojazdów. W tym pliku określono parametry generowania pojazdów, takie jak: – Prawdopodobieństwo pojawienia się pojazdu na określonej trasie, – Parametry takie jak „departLane” (np. wartość „free”, co oznacza dowolny pas startowy) oraz „departSpeed” ustawione na „random” (co odzwierciedla naturalne różnice w prędkościach pojazdów), – Okres symulacji (np. od 0 do 3600 sekund).

Dzięki temu model odzwierciedla zmienność i losowość zachowań kierowców, co jest kluczowe przy analizie dynamiki ruchu drogowego.

7.2 Szczegółowy opis modelu

Model testowy symuluje ruch pojazdów w sieci drogowej, obejmując węzły, drogi, ograniczenia prędkości, trajektorie pojazdów oraz sposób ich generowania. Struktura sieci składa się z węzłów wylotowych, skrzyżowań sterowanych sygnalizacją oraz węzłów wewnętrznych. Drogi podzielono na zewnętrzne (13.89 m/s, *50 km/h*) oraz wewnętrzne o zróżnicowanych prędkościach dostosowanych do manewrów. Model definiuje 12 przepływów ruchu o różnych poziomach natężenia, co pozwala na realistyczne odwzorowanie rzeczywistych warunków drogowych.



Rysunek 10 Badany model

Węzły

Zdefiniowano 3 rodzaje węzłów:

1. **Węzły wylotowe „dead\_end”:**  
   Miejsca wejścia/wyjścia z sieci występują w punktach:  
   P1(E12), P2(E20), P3(E0), P6(E3), P7(E4), P10(E7), P11(E14), P12(E22)  
   (łącznie 8 punktów)
2. **Skrzyżowań sterowanych sygnalizacją (typ „traffic\_light”):**  
   Występują w punktach: P4, P5, P8, P9 (łącznie 4 skrzyżowania)
3. **Węzły wewnętrznych (internal):**  
   Aby precyzyjnie odwzorować geometrię i przepływ ruchu wewnątrz skrzyżowań, dla każdego skrzyżowania sterowanego (P4, P5, P8, P9) utworzono 4 węzły wewnętrzne (np. :P4{12–15}\_0, :P5{12–15}\_0, :P8{12–15}\_0, :P9{12–15}\_0), co daje łącznie 16 dodatkowych punktów.

Drogi

Drogi zostały podzielone według dwóch kryteriów

1. **Drogi zewnętrzne (łączące główne węzły):**  
   W pliku zdefiniowano 24 drogi zewnętrzne – po 12 o identyfikatorach dodatnich E0, E1, E2, E3, E4, E7, E12, E13, E14, E20, E21, E22 oraz 12 o identyfikatorach ujemnych (np. –E0, –E1, –E2, –E3, –E4, –E7, –E12, –E13, –E14, –E20, –E21, –E22.
2. **Drogi wewnętrzne (definiujące szczegółowy przebieg ruchu wewnątrz skrzyżowań):**  
   Dla każdego skrzyżowania sterowanego stworzono 16 dróg wewnętrznych w celu określanie obciążenia występującego na skrzyżowaniu.

Prędkości na drogach:

W celu zwiększenia realizmu na drogach występują różne ograniczenia prędkości

1. **Drogi zewnętrzne:** **(główne arterie)**  
   Wszystkie pasy ruchu na drogach łączących główne węzły mają zadeklarowaną prędkość 13.89 m/s, co odpowiada około 50 km/h.
2. **Drogi wewnętrzne:** **(w obrębie skrzyżowań)**  
   W obrębie skrzyżowań prędkości są zróżnicowane, aby odwzorować manewry skrętu i hamowanie: Te różnice pozwalają na realistyczne odwzorowanie zachowania pojazdów przy wjeżdżaniu w skrzyżowania i wykonywaniu manewrów.

6.51 m/s (~23.4 km/h) – pasy skrętu i manewrów hamowania

8.00 m/s (~28.8 km/h) – łagodne zakręty i przejścia między pasami

13.89 m/s (~50 km/h) – proste odcinki wewnętrzne

Trajektorie ruchu

Model ruchu opiera się na 12 zdefiniowanych przepływach, w których określono:

1. **Kierunki ruchu (atrybuty from i to):**Każdy przepływ wskazuje, z którego zewnętrznego pasa (drogi) pojazdy wchodzą do sieci, a do którego ją opuszczają.  
   Przykładowo, *flow\_random1* definiuje ruch z krawędzi E0 (droga wychodząca z węzła P3, kierunek do P4) do krawędzi E3 (droga łącząca P5 z P6).  
   Niektóre przepływy zawierają atrybut *via*, który wymusza przejazd przez określone fragmenty sieci (np. *flow\_random7* przechodzi przez krawędzie -E2 oraz E1).

Parametry generowania pojazdów:

Atrybut *probability* określa szansę pojawienia się pojazdu na danym przepływie w każdej jednostce czasu.

W modelu występują dwa poziomy intensywności:

1. **Probability 0.1** – oznacza wyższą częstotliwość generowania pojazdów (około 0.1 pojazdu na sekundę, co daje średnio około 360 pojazdów na godzinę),
2. **Probability 0.01** – oznacza rzadszy ruch (około 36 pojazdów na godzinę).

Pozostałe parametry, takie jak *departLane* ustawione na "free" (dowolny pas startowy) oraz *departSpeed* ustawione na "random" (losowa prędkość początkowa), wprowadzają element losowości, symulując naturalne zachowania kierowców.

7.3 Podsumowanie modelu

Model obejmuje 8 węzłów wylotowych, które stanowią punkty wejścia i wyjścia z sieci, oraz 4 skrzyżowania sterowane sygnalizacją świetlną. Dodatkowo zdefiniowano 16 węzłów wewnętrznych, które precyzyjnie odwzorowują dynamikę ruchu wewnątrz skrzyżowań.   
W modelu występują 24 drogi zewnętrzne, na których obowiązuje prędkość 13.89 m/s (50 km/h), oraz 64 drogi wewnętrzne, które uwzględniają szczegóły przepływu pojazdów w obrębie skrzyżowań i posiadają zróżnicowane ograniczenia prędkości.

Model definiuje 12 przepływów ruchu (flows), które określają kierunki przemieszczania się pojazdów pomiędzy różnymi krawędziami sieci. Każdy przepływ posiada atrybuty from i to, a niektóre także via, wymuszający przejazd przez dodatkowe odcinki. Wprowadzono dwa poziomy natężenia ruchu.

Dodatkowe pliki 2x2.dat.xml oraz 2x2.add.xml zostały przygotowane jako rozszerzenie modelu, jednak w obecnej konfiguracji nie zawierają dodatkowych danych.

Taki model umożliwia realistyczną symulację ruchu drogowego, pozwalając na analizę przepływów pojazdów, badanie ich zachowań na skrzyżowaniach oraz ocenę skuteczności sterowania ruchem za pomocą sygnalizacji świetlnej. Dzięki zróżnicowanym ograniczeniom prędkości i losowości w generowaniu pojazdów, model dobrze odwzorowuje rzeczywiste warunki drogowe i może być używany do testowania różnych strategii zarządzania ruchem

1. Implementacja algorytmu AC w środowisku testowym

8.1 TaCI

W implementowanym modelu symulacji ruchu drogowego wykorzystano interfejs **TraCI (Traffic Control Interface)**, który umożliwia komunikację pomiędzy kodem sterującym a symulatorem **SUMO (Simulation of Urban Mobility)**. Dzięki TraCI możliwe jest dynamiczne sterowanie sygnalizacją świetlną, monitorowanie parametrów ruchu oraz interaktywna optymalizacja przepływu pojazdów w czasie rzeczywistym.

W omawianym kodzie połączenie z SUMO realizowane jest poprzez **moduł TraCI w Pythonie**. SUMO jest uruchamiane w trybie serwera za pomocą polecenia:

traci.start([SUMO\_BINARY, "-c", CONFIG\_FILE])

gdzie SUMO\_BINARY określa, czy używana jest wersja sumo czy sumo-gui, a CONFIG\_FILE wskazuje na plik konfiguracji symulacji (2x2.sumocfg).

Kod wykorzystuje **TraCI do sterowania sygnalizacją świetlną**, modyfikując fazy świateł na skrzyżowaniach (P4, P5, P8, P9). Fazy te są wybierane na podstawie strategii **uczenia ze wzmocnieniem (Reinforcement Learning)**, a ich ustawienie odbywa się za pomocą:

traci.trafficlight.setRedYellowGreenState(tls\_id, phases[action])

Funkcja get\_state() pobiera dane o długościach kolejek pojazdów i czasie oczekiwania na poszczególnych skrzyżowaniach, które następnie są przekazywane do modelu uczenia maszynowego jako wejściowy stan środowiska.

W procesie uczenia modelu **Actor-Critic**, wybór akcji odbywa się na podstawie prawdopodobieństwa wyznaczonego przez warstwę aktora. W przypadku, gdy przez zbyt długi czas nie następuje zmiana faz świateł, algorytm wprowadza losową fazę sygnalizacji, co symuluje adaptację do warunków ruchu.

Podczas symulacji kod **wykonuje kroki symulacyjne** SUMO, przechodząc do następnej iteracji modelu:

traci.simulationStep()

Po zakończeniu epizodu symulacji, połączenie TraCI jest zamykane za pomocą:

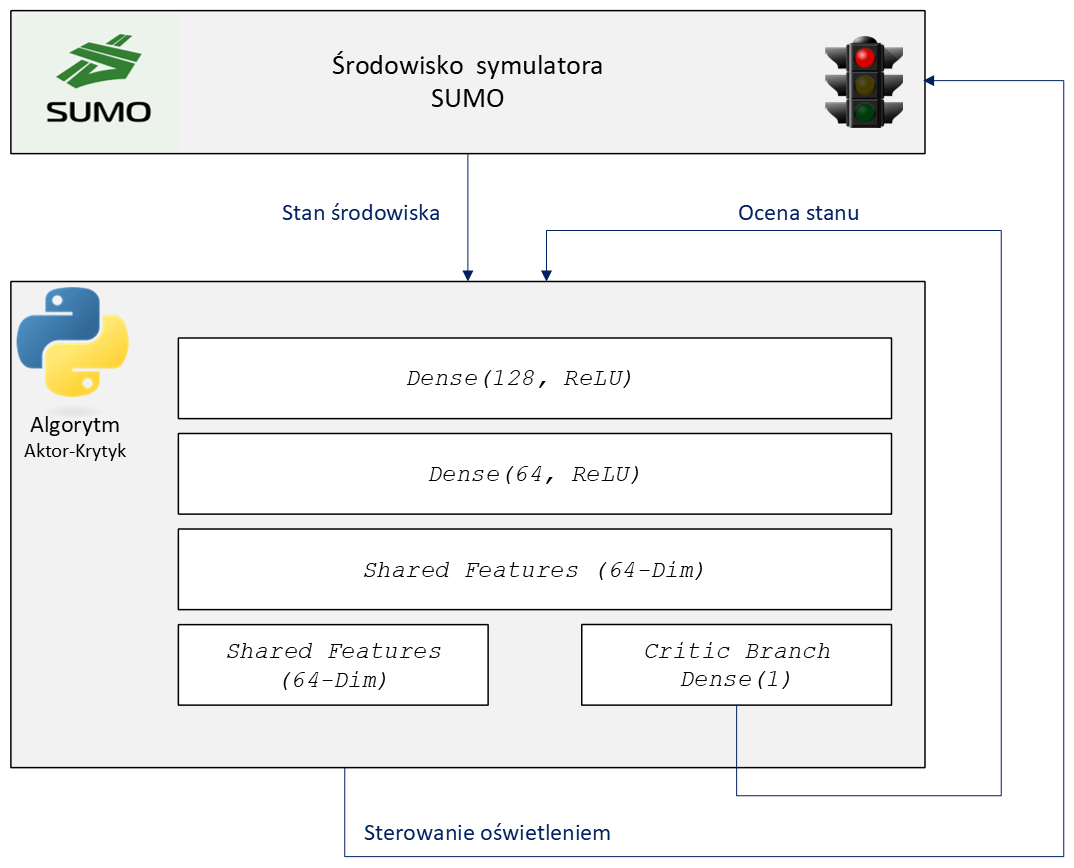
traci.close()

Dzięki temu rozwiązaniu symulator SUMO działa jako środowisko interaktywne, które w czasie rzeczywistym reaguje na decyzje podejmowane przez model uczący się. Pozwala to na badanie efektywności różnych strategii sterowania ruchem drogowym oraz optymalizację sygnalizacji świetlnej pod kątem minimalizacji zatorów i skrócenia czasu oczekiwania pojazdów.



Schemat blokowy przepływu komunikacji

8.2 Sieć neuronowa



**Warstwy i ich funkcje**

1. **Warstwa wspólna (self.common)**
   * Składa się z **dwóch warstw gęstych** (*Dense*), które przetwarzają dane wejściowe dotyczące stanu ruchu.
   * Pierwsza warstwa zawiera **128 neuronów**, a druga **64 neurony**, obie z funkcją aktywacji **ReLU** (relu), co umożliwia modelowi skuteczne odwzorowanie nieliniowych zależności.
   * Wykorzystano inicjalizator wag **He Normal**, który poprawia stabilność uczenia i przyspiesza zbieżność.
   * Ta część sieci pełni funkcję ekstrakcji cech, które następnie są przekazywane do warstw aktora i krytyka.

self.common = tf.keras.Sequential([

layers.Dense(128, activation="relu", kernel\_initializer="he\_normal"),

layers.Dense(64, activation="relu", kernel\_initializer="he\_normal")

])

1. **Warstwa aktora (self.actor)**
   * Odpowiada za przewidywanie prawdopodobieństw wyboru poszczególnych faz sygnalizacji świetlnej.
   * Liczba neuronów w tej warstwie wynosi **num\_tls × num\_phases**, gdzie num\_tls to liczba sygnalizatorów, a num\_phases to liczba możliwych faz świateł.
   * Zastosowano funkcję aktywacji **softmax**, dzięki czemu wyjściowe wartości można interpretować jako rozkład prawdopodobieństwa wyboru każdej fazy.
   * Model na podstawie tych wartości wybiera najodpowiedniejszą fazę świateł, minimalizując korki i czas oczekiwania pojazdów.

self.actor =

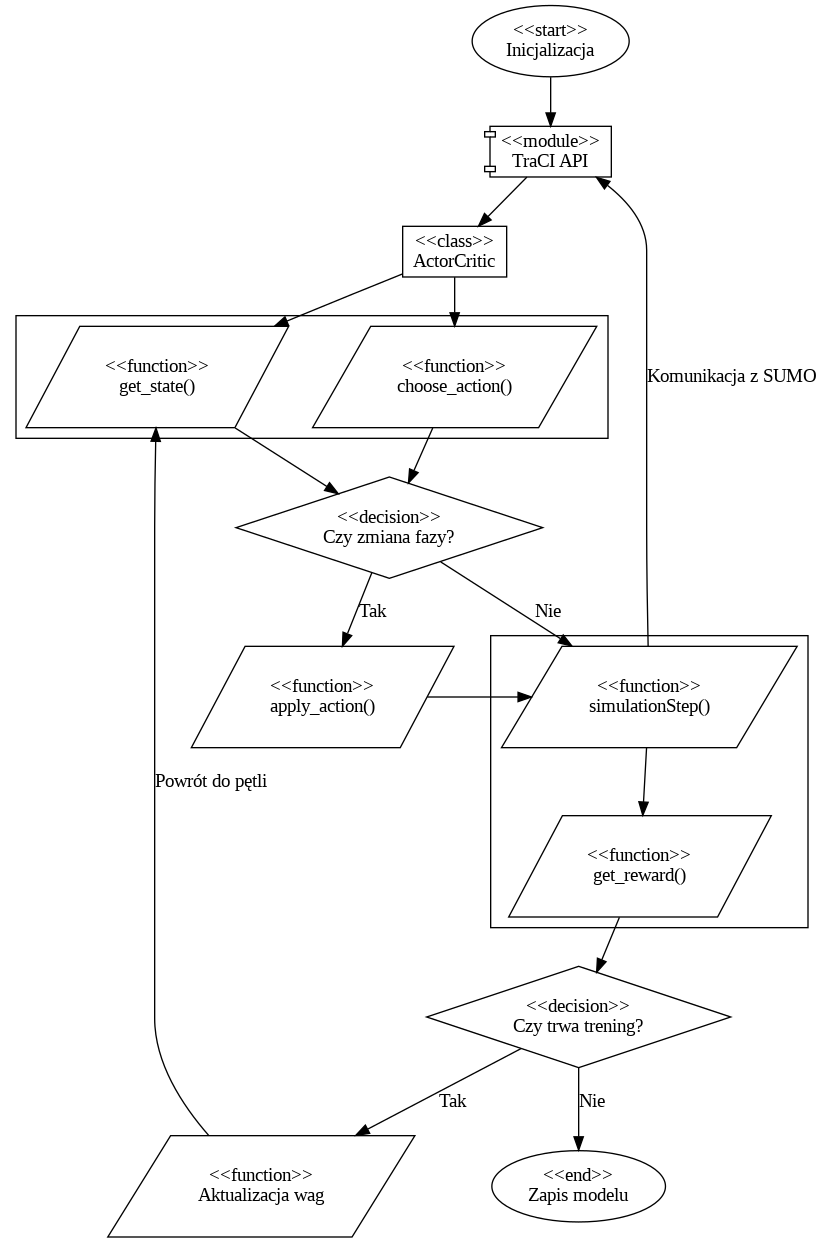
layers.Dense(num\_tls \* num\_phases, activation="softmax", name="actor")

1. **Warstwa krytyka (self.critic)**
   * Służy do oceny wartości danego stanu ruchu drogowego, pomagając modelowi optymalizować decyzje podejmowane przez aktora.
   * Zawiera tylko **jeden neuron**, który zwraca **skalarną wartość**, reprezentującą oczekiwaną przyszłą nagrodę dla aktualnego stanu.
   * Nie stosuje się tutaj funkcji aktywacji, ponieważ wartość stanu może przyjmować dowolne wartości rzeczywiste.
   * Informacje z tej warstwy są wykorzystywane do aktualizacji polityki sterowania ruchem, tak aby w dłuższej perspektywie osiągać lepsze wyniki w zakresie płynności ruchu.

self.critic = layers.Dense(1, name="critic")

Ogólnie rzecz biorąc, taka architektura sieci (z warstwą wspólną, aktora i krytyka) jest optymalna, ponieważ umożliwia efektywne przetwarzanie dynamicznych danych wejściowych, elastyczne podejmowanie decyzji sterowania poprzez generowanie rozkładu prawdopodobieństwa oraz stabilną ocenę wartości stanu, co łącznie wspiera szybkie i trafne reagowanie systemu na zmieniające się warunki ruchu drogoweg

8.3 Implementacja algorytmu systemu sterowania



****

**Adaptacyjne Sterowanie Ruchem na Skrzyżowaniach P4, P5, P8 i P9**

Kod realizuje adaptacyjne sterowanie ruchem przy wykorzystaniu symulacji SUMO oraz agenta uczenia ze wzmocnieniem opartego na architekturze Actor-Critic. Dzięki integracji modelu symulacji z algorytmem uczenia, system iteracyjnie optymalizuje ustawienia faz sygnalizacyjnych, co przekłada się na poprawę przepustowości skrzyżowań i redukcję opóźnień.

**1. Konfiguracja i Inicjalizacja**

1.1. **Definicja stałych konfiguracyjnych:**

1. SUMO\_BINARY: Ustawione na "sumo". Dla wizualizacji symulacji można zmienić na "sumo-gui".
2. CONFIG\_FILE: Ścieżka do pliku konfiguracyjnego SUMO, który definiuje model sieci drogowej.
3. TLS\_IDS: Lista identyfikatorów sygnalizatorów – sterowane są skrzyżowania P4, P5, P8 i P9.
4. NUM\_PHASES: Liczba dostępnych faz (ustawiona na 4) oraz parametry określające warunki zmiany fazy, takie jak:
   * UNCHANGE\_LIMIT: Maksymalna liczba kroków (np. 50), po których, przy braku zmiany faz, następuje wymuszenie losowej zmiany.
   * FORCED\_DURATION: Okres (np. 30 kroków), przez który wymuszana jest losowa zmiana.
   * PENALTY: Kara (np. -0.1) za wymuszenie losowej zmiany.

1.2. Powyższe ustawienia umożliwiają elastyczne zarządzanie dynamiką symulacji oraz adaptacyjnym sterowaniem ruchem.

**2. Architektura Modelu Actor-Critic**

2.1. Klasa ActorCritic dziedziczy po tf.keras.Model i składa się z:

1. Części wspólnej (common): Sieć neuronowa złożona z dwóch warstw Dense (128 i 64 neurony) z aktywacją ReLU, która przetwarza stan wejściowy.
2. Warstwy aktora: Warstwa Dense z aktywacją softmax, której wyjście ma wymiar równy liczbie sygnalizatorów pomnożonej przez liczbę faz (4 × 4 = 16). Generuje rozkład prawdopodobieństwa wyboru poszczególnych faz dla każdego skrzyżowania.
3. Warstwy krytyka: Pojedyncza warstwa Dense, która ocenia jakość danego stanu (przewidując jego wartość).

2.2. Dzięki tej architekturze model uczy się, które akcje (ustawienia faz) poprawiają przepływ ruchu, jednocześnie oceniając wartość aktualnej sytuacji na skrzyżowaniach.

**3. Pozyskiwanie Stanu Symulacji**

3.1. Funkcja **get\_state()** zbiera informacje o aktualnym stanie skrzyżowań:

1. Dla każdego sygnalizatora obliczana jest suma pojazdów zatrzymanych (queue\_lengths) oraz łączny czas oczekiwania (waiting\_times) na pasach kontrolowanych przez dany sygnalizator.
2. Uzyskane wartości są normalizowane przy użyciu ustalonych maksymalnych wartości (np. max\_queue\_length = 250, max\_waiting\_time = 1000), a następnie łączone w jeden wektor stanu.

3.2. W ten sposób agent otrzymuje reprezentację sytuacji na skrzyżowaniach, co stanowi dane wejściowe do modelu.

**4. Wybór Akcji (Ustawienia Fazy)**

4.1. **Sekwencyjność operacji:**

1. Najpierw wywoływana jest funkcja **get\_state()**, która pobiera aktualny stan systemu.
2. Następnie, na podstawie tego stanu, funkcja **choose\_action()** dokonuje wyboru akcji.

4.2. **Funkcja choose\_action():**

1. Przyjmuje rozkład prawdopodobieństwa (output z warstwy aktora) i przekształca go do macierzy o wymiarach (liczba sygnalizatorów, liczba faz).
2. Wartości są klipowane (aby wyeliminować ewentualne wartości ujemne) oraz normalizowane w każdym wierszu.
3. Dla każdego sygnalizatora losowana jest akcja (numer fazy) zgodnie z otrzymanym rozkładem prawdopodobieństwa.

4.3. Wybrane akcje decydują o tym, która z czterech możliwych sekwencji świateł (np. "GGgrrrGGgrrr", "yyyyyyyyyyyy" itp.) zostanie zastosowana na danym skrzyżowaniu.

**5. Aplikacja Akcji w Symulacji**

5.1. Funkcja **apply\_action()**:

1. Przyjmuje listę akcji (indeksów faz) i dla każdego sygnalizatora ustawia odpowiednią sekwencję świateł za pomocą interfejsu **traci**.
2. Po zmianie faz kod wypisuje informację o bieżącym kroku symulacji oraz zastosowanych ustawieniach, co ułatwia monitorowanie przebiegu symulacji.

**6. Obliczanie Nagrody**

6.1. Funkcja **get\_reward()**:

1. Oblicza nagrodę na podstawie całkowitej długości kolejek (suma pojazdów zatrzymanych) oraz łącznego czasu oczekiwania.
2. W przypadku wymuszonej zmiany fazy (gdy forced\_steps > 0), do nagrody dodawana jest kara określona przez wartość **PENALTY**.
3. Nagroda stanowi kombinację premii za „wolny przepływ” (free\_flow\_bonus) oraz kar wynikających z długich kolejek i wysokiego czasu oczekiwania, co motywuje agenta do utrzymania płynności ruchu.

**7. Proces Treningu**

7.1. Funkcja **train\_actor\_critic()** realizuje główną pętlę treningową, obejmującą:

1. Uruchomienie symulacji przez interfejs **traci**. Dla każdej z 30 epizodów wykonywanych jest 5000 kroków symulacji.
2. Aktualizację wag modelu, która odbywa się w losowo wybranym przedziale 1000 kroków (learning\_duration).
3. Co 10 kroków symulacji podejmowaną jest decyzję o zmianie faz. Jeśli przez 50 kolejnych kroków (UNCHANGE\_LIMIT) fazy pozostają niezmienione, następuje wymuszenie losowej zmiany na określony czas (FORCED\_DURATION), co ma na celu zapobieżenie utknięciu w suboptymalnym stanie – taka zmiana jest karana obniżeniem nagrody.
4. Po każdej zmianie faz symulacja wykonuje krok (traci.simulationStep()), pobierany jest nowy stan, a nagroda jest obliczana.
5. Aktualizacja modelu obejmuje obliczenie wartości docelowej (target) przy użyciu bieżącej nagrody oraz przewidywanej wartości stanu następnego (z dyskontowaniem przy gamma = 0.9). Dzięki mechanizmowi GradientTape obliczane są straty aktora i krytyka, które następnie są minimalizowane przy użyciu optymalizatora Adam. Gradienty są przycinane (clip by global norm) dla stabilności procesu uczenia.
6. Po zakończeniu każdego epizodu, wyświetlana jest całkowita uzyskana nagroda, a wagi modelu zapisywane są na dysku.

7.2. Dzięki temu agent uczy się, które akcje w danym stanie poprawiają przepływ ruchu, i modyfikuje swoje decyzje na podstawie uzyskanych nagród.

**8. Integracja z Symulacją SUMO**

8.1. Cały kod opiera się na interakcji z symulacją SUMO poprzez moduł **traci**, który umożliwia:

1. Uruchomienie symulacji na podstawie pliku konfiguracyjnego SUMO (np. 2x2.sumocfg).
2. Pobieranie bieżących danych o ruchu (kolejki, czasy oczekiwania) dla poszczególnych sygnalizatorów.
3. Dynamiczną zmianę ustawień sygnalizacji świetlnej w trakcie symulacji.

8.2. W ten sposób kod łączy model symulacji ruchu z algorytmem uczenia, umożliwiając iteracyjną optymalizację sterowania ruchem w symulowanym środowisku.  
**Podsumowanie**

9.1. Kod KOD\_A1.py integruje model SUMO z agentem uczenia ze wzmocnieniem, który:

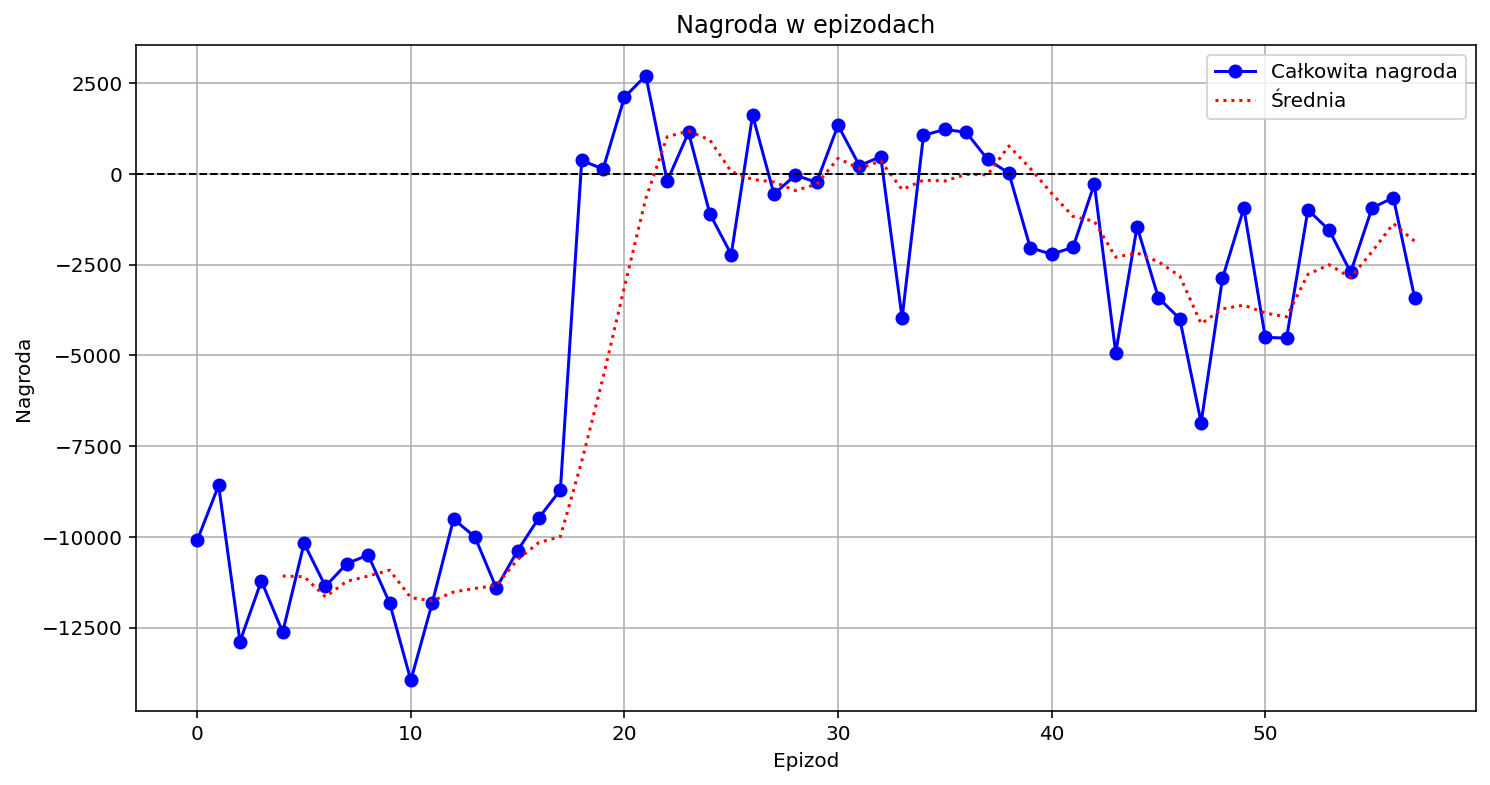
1. Pobiera dane dotyczące kolejek i czasu oczekiwania na skrzyżowaniach.
2. Wykorzystuje model Actor-Critic (zbudowany w TensorFlow) do wyboru optymalnych faz sygnalizacyjnych.
3. Aktualizuje swoje decyzje na podstawie uzyskiwanych nagród, modyfikując strategię sterowania ruchem.
4. Zawiera mechanizmy zapobiegające utknięciu w suboptymalnych stanach, takie jak wymuszenie losowej zmiany fazy po długim okresie bez zmian.

9.2. Dzięki tej integracji możliwe jest dynamiczne i adaptacyjne sterowanie ruchem, co przekłada się na poprawę przepustowości skrzyżowań oraz redukcję opóźnień w symulacji.

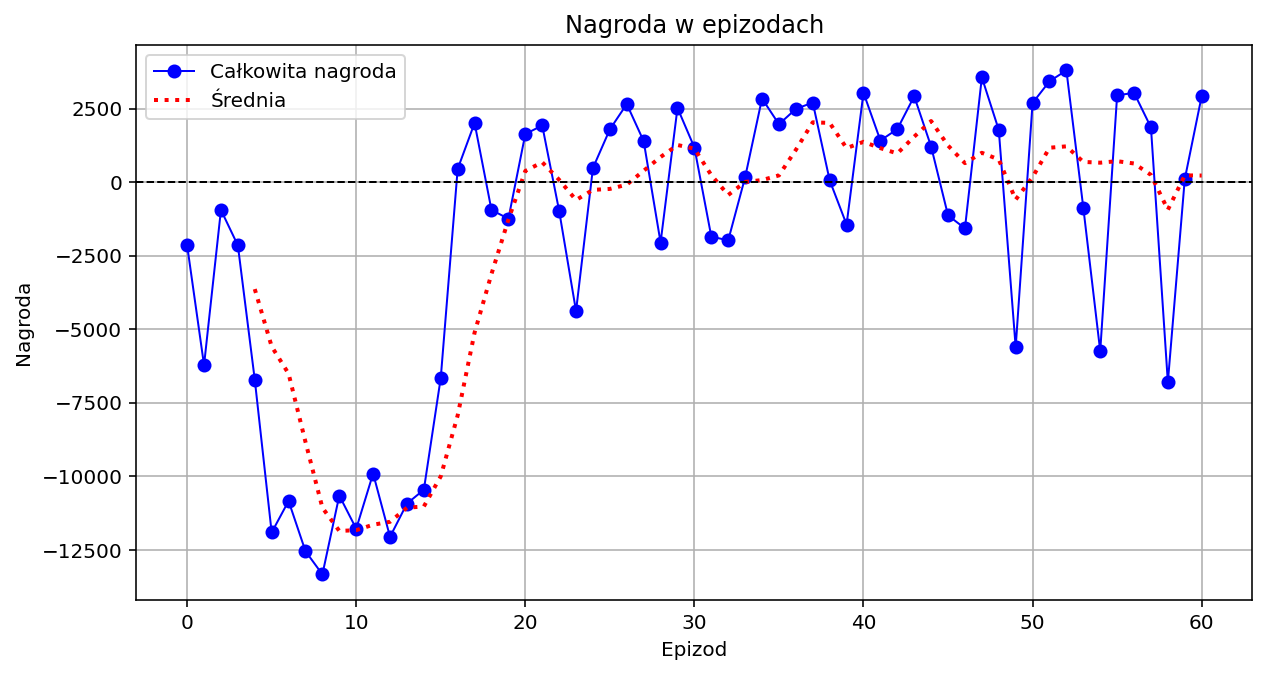
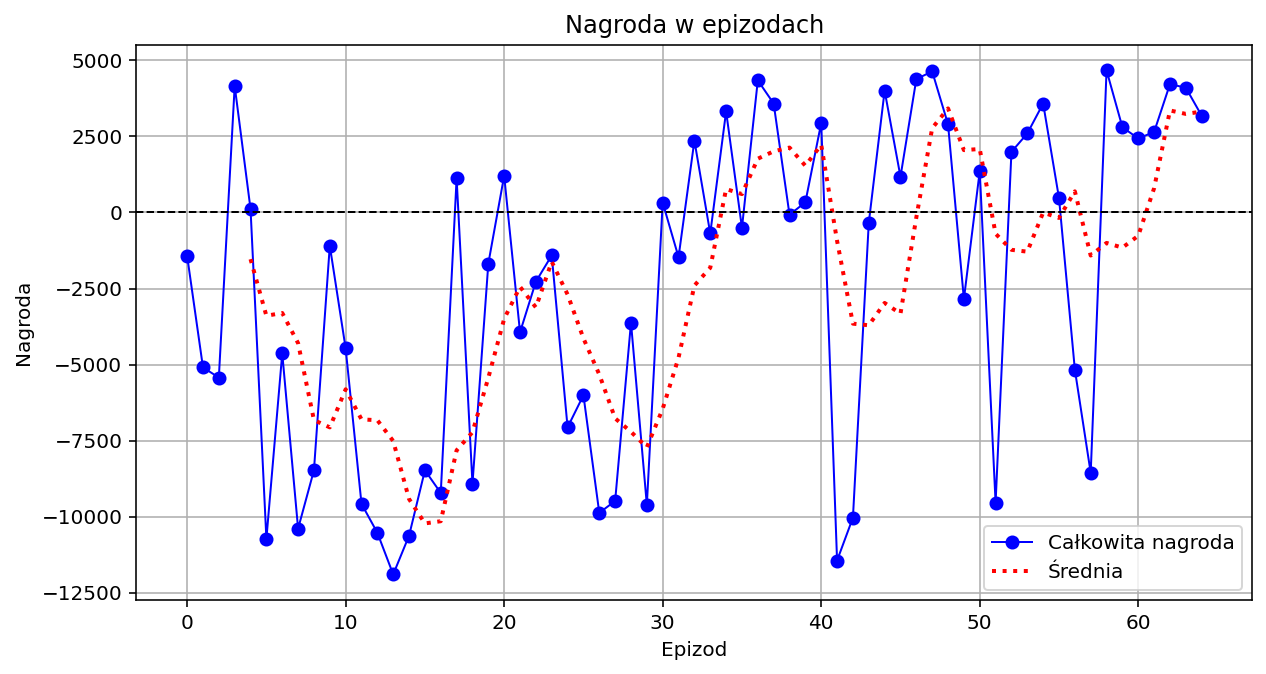
8.4 Uczenie modelu

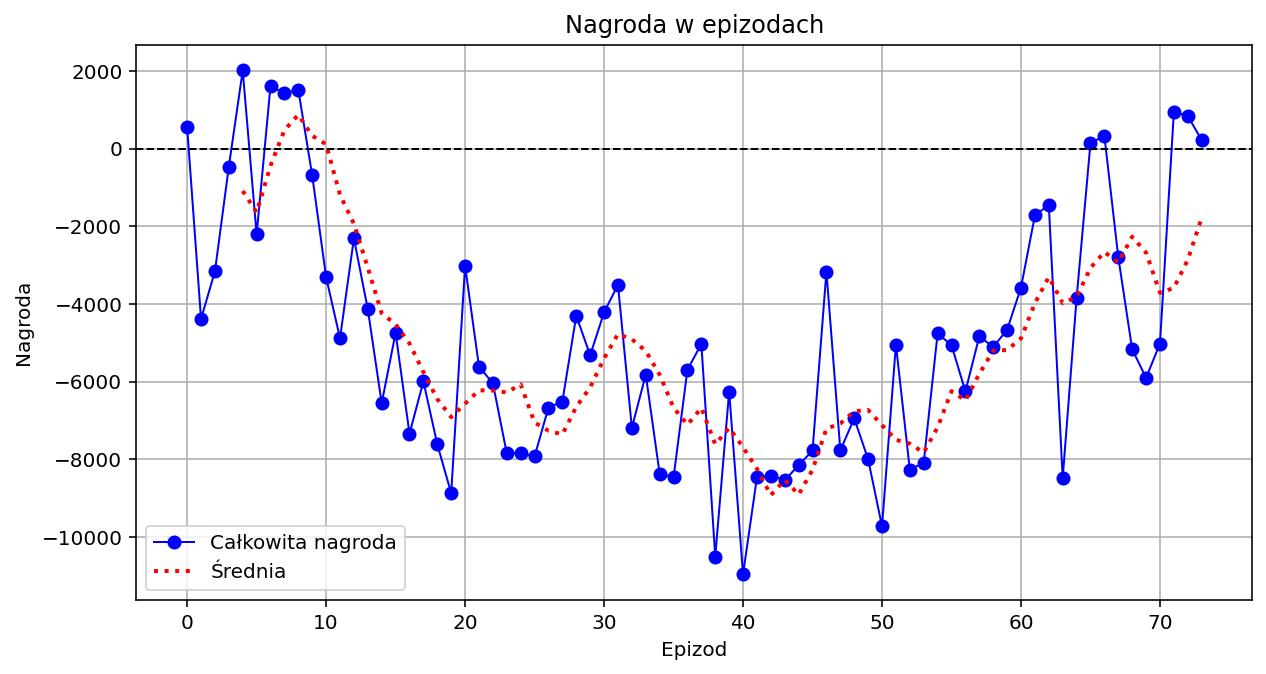
**Sieć 0**

**Siec 1**



**Sieć 2**

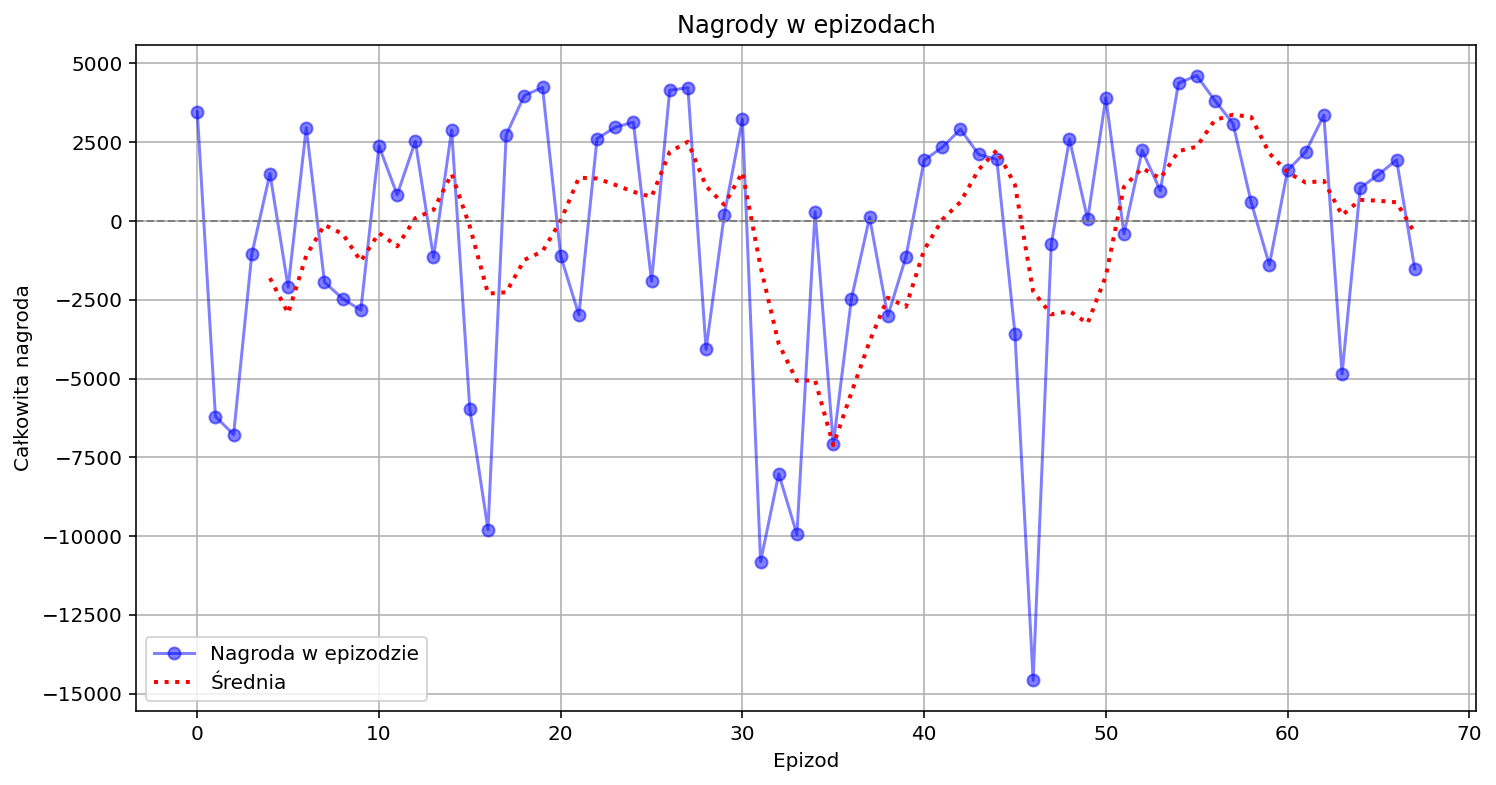
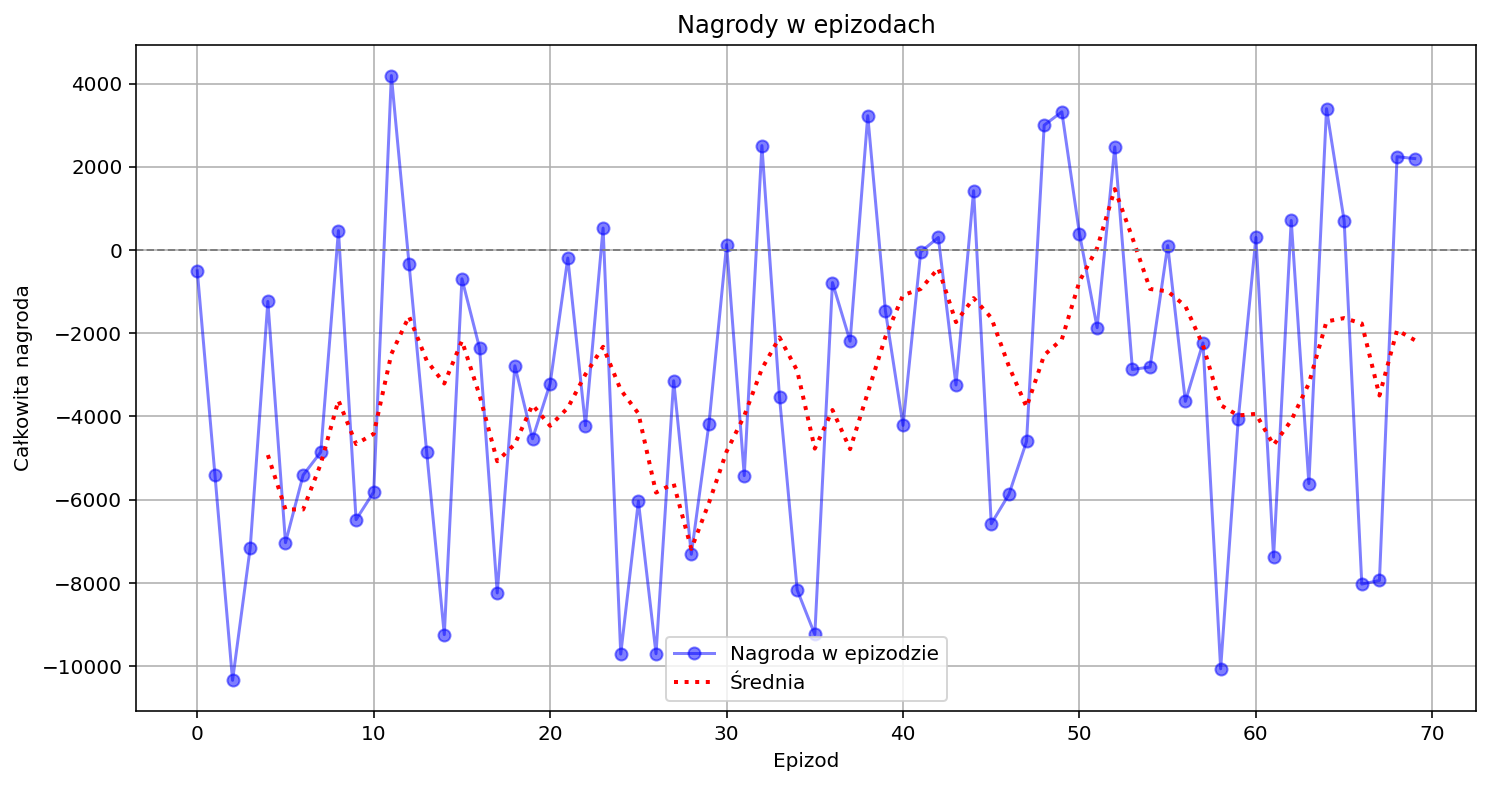


**Sieć nr 3**

epsilon = max(0.01, 0.2 \* (0.89 \*\* episode))

gamma = 0.95

optimizer = tf.keras.optimizers.Adam(learning\_rate=0.0005)



1. Analiza zgromadzonych danych.

*\*\*\*\*\*\*\*\*\*Prace trwają \*\*\*\*\*\*\*\*\**

*Zastosowanie tego kodu do sterowania modelem SUMO pozwala wyciągnąć szereg ciekawych wniosków, zarówno pod kątem efektywności sterowania ruchem, jak i możliwości adaptacyjnego uczenia się. Oto kluczowe wnioski:*

*• Agent uczenia się ze wzmocnieniem, oparty na architekturze Actor-Critic, jest w stanie dynamicznie dostosowywać sekwencje faz sygnalizacyjnych na skrzyżowaniach (P4, P5, P8, P9) na podstawie bieżących danych o kolejkach i czasach oczekiwania. Dzięki temu system reaguje na zmieniające się warunki ruchu i poprawia płynność przejazdu.*

*• Zastosowanie mechanizmu monitorowania stanu (np. analiza długości kolejek i sumarycznego czasu oczekiwania) umożliwia precyzyjną ocenę sytuacji na skrzyżowaniach. To pozwala agentowi na podejmowanie decyzji, które minimalizują opóźnienia i redukują zatłoczenie.*

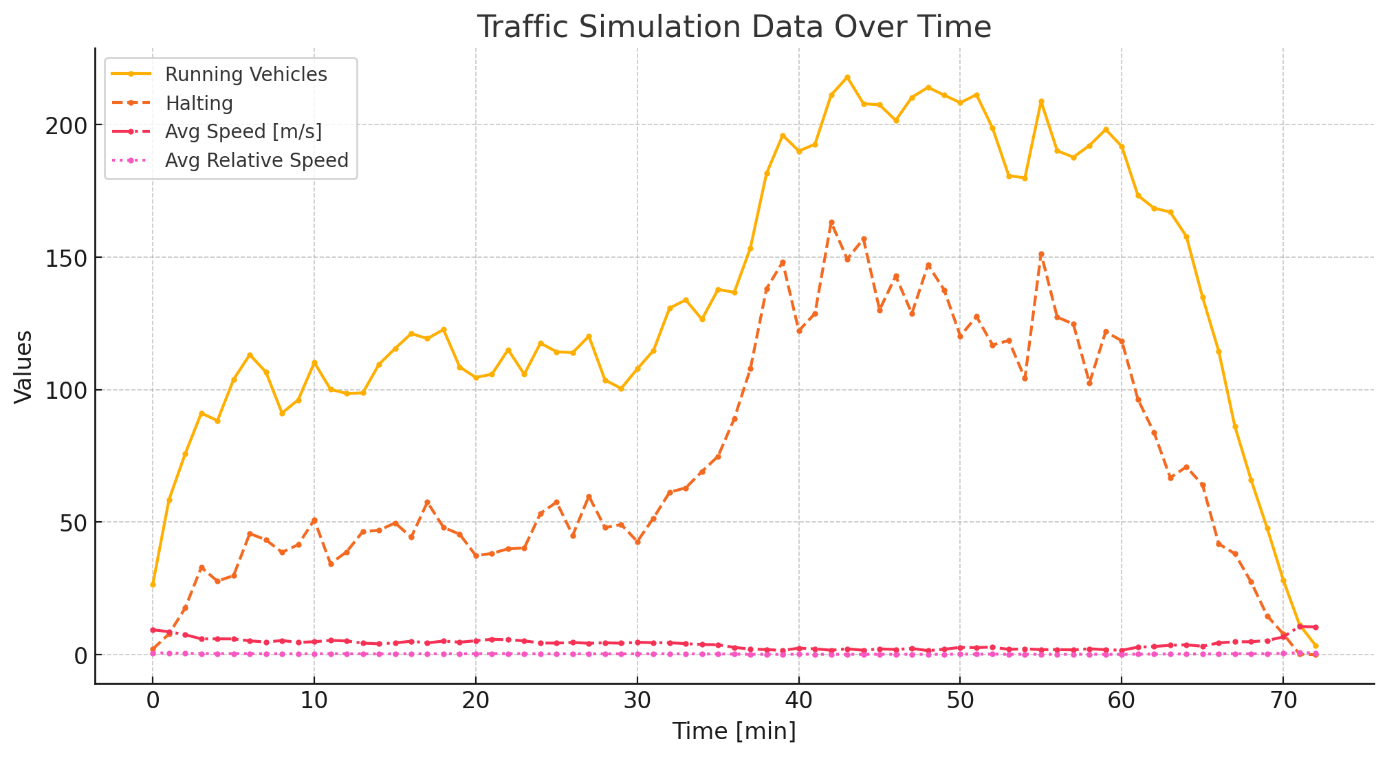
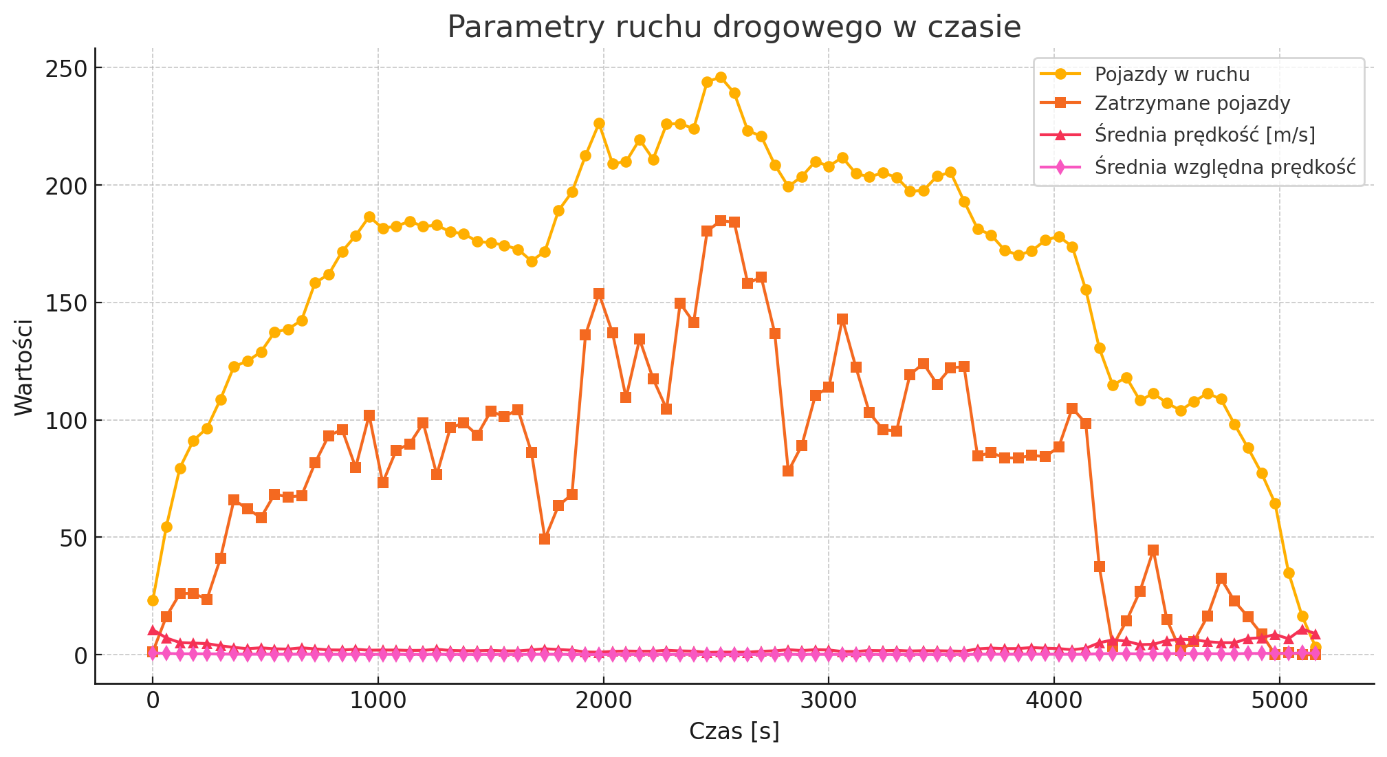
*• Mechanizm wymuszonej losowej zmiany faz, stosowany po wykryciu długiego okresu bez zmian, skutecznie zapobiega utknięciu w suboptymalnych stanach. Choć wymuszenie losowych faz generuje karę, to jednak pomaga systemowi w eksploracji innych rozwiązań, co przekłada się na lepsze dostosowanie do nieprzewidywalnych scenariuszy ruchowych.*

*• Proces treningu, realizowany w określonych przedziałach czasowych, pokazuje, że adaptacyjne sterowanie ruchem może stopniowo poprawiać jakość przepływu na skrzyżowaniach. Agent uczy się, które akcje prowadzą do uzyskania wyższych nagród, co oznacza mniejsze zatłoczenie i krótsze czasy oczekiwania.*

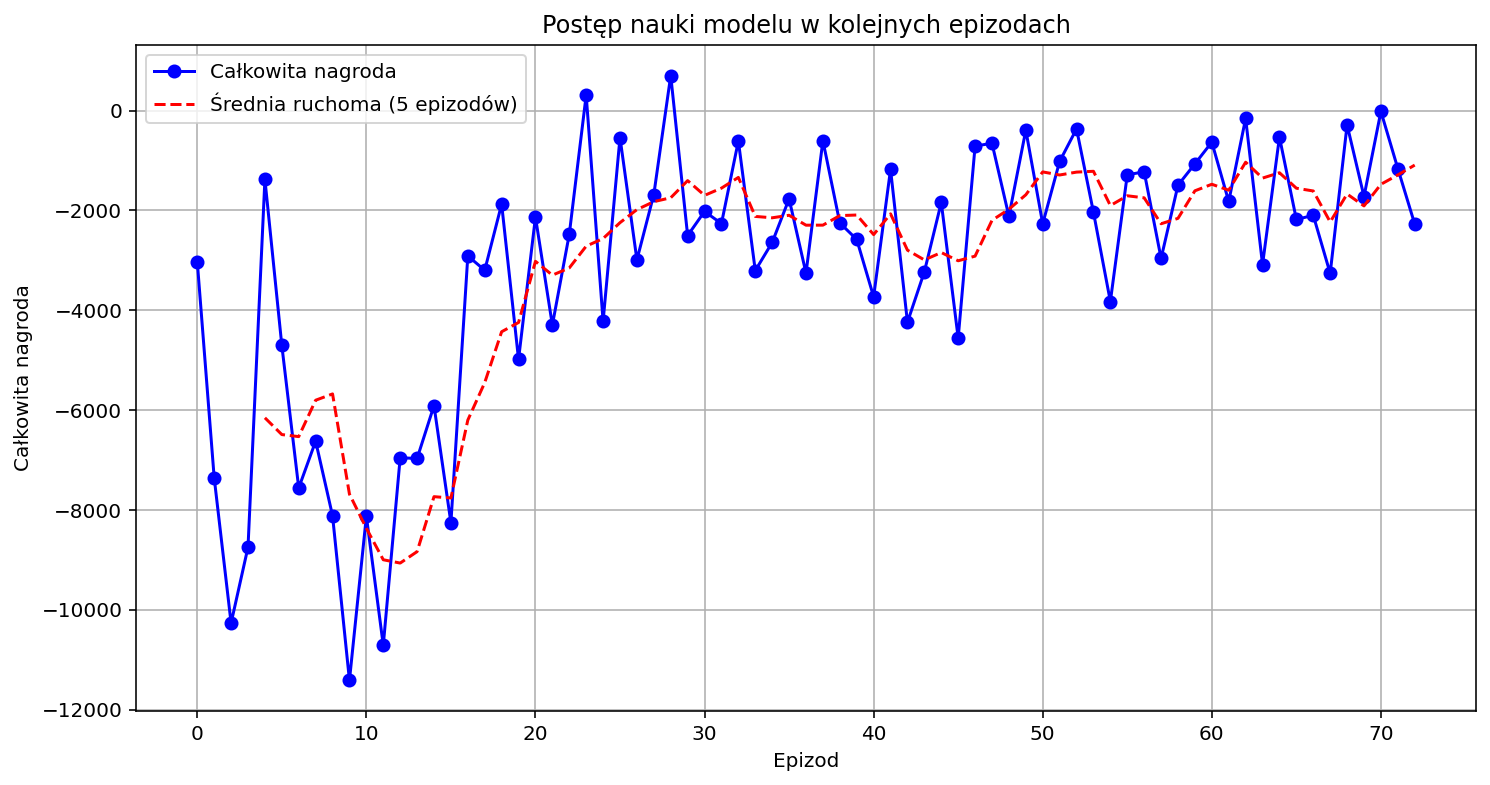
*• Integracja środowiska symulacyjnego SUMO z mechanizmami uczenia głębokiego otwiera możliwości eksperymentowania z różnymi strategiami sterowania ruchem. Wyniki eksperymentu sugerują, że adaptacyjne sterowanie, oparte na analizie stanu systemu, może być skutecznym narzędziem do optymalizacji przepustowości sieci drogowej.*

*Podsumowując, kod nie tylko umożliwia dynamiczne sterowanie sygnalizacją świetlną, ale również ilustruje, jak inteligentny agent może poprawić jakość ruchu drogowego poprzez ciągłą adaptację do zmieniających się warunków na drodze. Takie podejście może być cennym wsparciem w projektowaniu systemów zarządzania ruchem w rzeczywistych miastach.*

[*https://colab.research.google.com/drive/1L9nyLemv8MImeraglxeetxnfKC5QmNDD#scrollTo=NQqDo7vZiKrd*](https://colab.research.google.com/drive/1L9nyLemv8MImeraglxeetxnfKC5QmNDD#scrollTo=NQqDo7vZiKrd)



==========



|  |  |  |
| --- | --- | --- |
| **static** | Stałe czasy faz sygnalizacji, brak reakcji na ruch. | ❌ NIE |

|  |  |  |
| --- | --- | --- |
| **actuated** | Adaptacyjne sterowanie na podstawie liczby pojazdów. | ✅ TAK |

|  |  |  |
| --- | --- | --- |
| **delay\_based** | Adaptacyjne sterowanie na podstawie opóźnienia pojazdów. | ✅ TAK |

|  |  |  |
| --- | --- | --- |
| **traffic-actuated** | Działa podobnie do actuated, ale z bardziej zaawansowanym wykrywaniem pojazdów. | ✅ TAK |

|  |  |  |
| --- | --- | --- |
| **adaptive** | Najbardziej zaawansowane sterowanie - dostosowuje się dynamicznie do natężenia ruchu. | ✅ TAK |

|  |  |  |
| --- | --- | --- |
| **fixedTime** | To samo co static, ale może mieć kilka programów sterowania. | ❌ NIE |

|  |  |  |
| --- | --- | --- |
| **vehicle-actuated** | Bardziej szczegółowe sterowanie oparte na poszczególnych pojazdach. | ✅ TAK |

**ak działa actuated?**

1. **Każda faza ma**:
   * **Minimalny czas trwania (minDur)** → nie może być krótsza niż ten czas.
   * **Maksymalny czas trwania (maxDur)** → nie może być dłuższa niż ten czas.
   * **Czas początkowy (duration)** → standardowy czas fazy, jeśli brak zmian.
   * **Stan (state)** → określenie świateł (G - zielone, r - czerwone, y - żółte).
2. **Detektory ruchu (np. e1Detector)** umieszczone na drogach sprawdzają liczbę pojazdów.
3. **SUMO analizuje ruch i zmienia czas trwania światła zielonego:**
   * Jeśli **dużo pojazdów czeka**, faza zielonego światła się wydłuża (**do maxDur**).
   * Jeśli **ruch jest mały**, światło przechodzi szybciej do następnej fazy (**ale nie krócej niż minDur**).
4. **Dzięki temu zmniejsza się liczba zatrzymanych pojazdów i poprawia płynność ruchu.**

1. National-geographic - https://www.national-geographic.pl/nauka/nagroda-nobla-2024/ [↑](#footnote-ref-1)
2. Obserwator finansowy https://www.obserwatorfinansowy.pl/tematyka/makroekonomia/trendy-gospodarcze/fenomen-chatgpt-i-jego-skutki/ [↑](#footnote-ref-2)
3. Google https://deepmind.google/discover/blog/alphago-zero-starting-from-scratch/ [↑](#footnote-ref-3)
4. By Kara Nelson, CNN - https://edition.cnn.com/2023/11/24/us/garrett-morgan-traffic-signal-100-years-reaj/index.html [↑](#footnote-ref-4)
5. Marcin Ruchaj, „Algorytmy sterowania acykliczną sygnalizacją świetlną w zatłoczonej sieci drogowej”, Rozprawa Doktorska (Marcin\_Ruchaj.pdf) [↑](#footnote-ref-5)
6. Podsystem Sterowania Ruchem, Sprint/ITS/SCATS, Tadeusz Okoń i Daniel Jaros, https://www.itspolska.pl/wp-content/uploads/2022/02/Podsystem-sterowania-ruchem-Sprint-ITS-SCATS-w-Bydgoszczy.pdf [↑](#footnote-ref-6)
7. SCOOT® Version History, Split Cycle and Offset Optimisation Technique, https://trlsoftware.com/software/intelligent-signal-control/scoot/scoot-version-history/ [↑](#footnote-ref-7)
8. Politechnika Opolska Wydział Elektrotechniki, Automatyki i Informatyki Instytut Automatyki i Informatyki, Algorytmy sterowania acykliczną sygnalizacją świetlną w zatłoczonej sieci drogowej [↑](#footnote-ref-8)
9. Miśkiewicz M.: ViaPIACON – polska metoda sterowania ruchem drogowym. Przegląd ITS nr 4, Warszawa 2008. [↑](#footnote-ref-9)
10. Arthur Samuel, Some Studies in Machine Learning Using the Game of Checkers , https://www.cs.virginia.edu/~evans/greatworks/samuel1959.pdf [↑](#footnote-ref-10)
11. Feliks Krup, Sztuczna Inteligencja od Podstaw, (sztuczna-inteligencja-od-podstaw-feliks-kurp-helion-2.pdf) [↑](#footnote-ref-11)
12. Steven L. Brunton, J. Nathan Kutz, Data Driven Science & Engineering Machine Learning, Dynamical Systems, and Control (databookRL.pdf) [↑](#footnote-ref-12)
13. Google CLOUD, https://www.cloudskillsboost.google/focuses/10285?locale=pl&parent=catalog [↑](#footnote-ref-13)
14. Richard S. Sutton and Andrew G. Barto „Reinforcement Learning: An Introduction” - Second edition, in progres ”Complete Draft” November 5, 2017 http://incompleteideas.net/book/bookdraft2017nov5.pdf [↑](#footnote-ref-14)
15. Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto (wzór 3.8), http:, //incompleteideas.net/book/RLbook2020.pdf [↑](#footnote-ref-15)
16. Reinforcement Learning: An Introduction Second edition \*\*\*\*Complete draft\*\*\*\* March 11, 2018 Richard S. Sutton and Andrew G. Barto [↑](#footnote-ref-16)
17. Nature, Human-level control through deep reinforcement learning, https://www.nature.com/articles/nature14236 [↑](#footnote-ref-17)
18. Copyright © 2001-2024 German Aerospace Center (DLR) and others., https://sumo.dlr.de/docs/ [↑](#footnote-ref-18)
19. SUMO TraCI https://sumo.dlr.de/docs/TraCI/Protocol.html [↑](#footnote-ref-19)