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

Glosariusz terminów technicznych

ITS (Intelligent Transportation System) – Inteligentne systemy transportowe wspomagające zarządzanie ruchem drogowym.
RL (Reinforcement Learning) – Uczenie ze wzmocnieniem; agent uczy się przez interakcję ze środowiskiem, dążąc do maksymalizacji nagrody.
Aktor–Krytyk (Actor–Critic, AC) – Algorytm RL łączący wybór akcji (aktor) i ocenę stanu (krytyk).
SUMO – Symulator ruchu drogowego używany do testowania strategii sterowania ruchem.
TraCI – Interfejs umożliwiający sterowanie SUMO z poziomu kodu (np. Python).
DNN (Deep Neural Network) – Głęboka sieć neuronowa z wieloma warstwami ukrytymi.
Uczenie głębokie (DL) – Uczenie oparte na głębokich sieciach neuronowych.
MDP (Markov Decision Process) – Model RL opisujący stany, akcje, nagrody i przejścia.
δ, Δ (delta) – Symbol różnicy; np. błąd predykcji nagrody.
θ, w – Parametry (wagi) sieci neuronowej.
Softmax – Funkcja aktywacji przekształcająca dane wyjściowe na rozkład prawdopodobieństwa.
Epsilon-greedy – Strategia eksploracji polegająca na losowym wyborze akcji z prawdopodobieństwem ε.
Klipowanie gradientów – Technika ograniczania wartości gradientów w celu stabilizacji uczenia.
Reward shaping – Definiowanie funkcji nagrody, by lepiej kierować procesem uczenia.

1. Wstęp

Ruch drogowy odgrywa fundamentalną rolę w funkcjonowaniu współczesnych, zurbanizowanych społeczności. Ma on bezpośredni wpływ na ich rozwój gospodarczy, oraz na jakość życia. Dynamiczny wzrost liczby pojazdów generuje coraz większe obciążenie infrastruktury transportowej, której możliwości przepustowe stają się niewystarczające. Skutkuje to kongestią drogową, powodującą znaczące straty ekonomiczne oraz obniżaniem poziom życia mieszkańców. Problemy związane z zatłoczeniem infrastruktury drogowej wpływają negatywnie na czas reakcji służb ratunkowych.

Ponieważ rozbudowa sieci drogowej wiąże się z dużymi kosztami, coraz większe znaczenie mają nowoczesne metody zarządzania ruchem. Dynamiczne dostosowanie sygnalizacji świetlnej do aktualnych warunków drogowych staje się możliwe dzięki zastosowaniu zaawansowanych metod optymalizacji sterowania. Współczesne inteligentne systemy transportowe (Intelligent Transportation Systems – ITS), takie jak SCATS, SCOOT czy RHODES, pozwalają na bieżąco dostosowywać długość cykli oraz kolejność faz świateł, skutecznie redukując zatory i poprawiając płynność ruchu. Pomimo ich skuteczności, wciąż istnieje potencjał dalszego udoskonalenia tych systemów.

Obecnie rośnie zainteresowanie wykorzystaniem algorytmów uczenia maszynowego w wielu dziedzinach nauki i gospodarki. Potwierdzeniem znaczenia tej technologii było przyznanie Nagrody Nobla z dziedziny fizyki w 2024 roku Johnowi J. Hopfieldowi oraz Geoffreyowi E. Hintonowi za pionierskie prace nad sztucznymi sieciami neuronowymi.[1] Ich badania, w tym mechanizm wstecznej propagacji błędów, przyczyniły się do rozwoju głębokich sieci neuronowych – podstawy współczesnych rozwiązań sztucznej inteligencji.

Przykłady sukcesów takich technologii jak modele językowe (ChatGPT), systemy autonomiczne (Tesla Autopilot) oraz zastosowania w biologii obliczeniowej (AlphaFold) wskazują, że rozwiązania oparte na sztucznej inteligencji skutecznie rozwiązują złożone problemy współczesnego świata.[2]

W tym kontekście wykorzystanie metod AI w sterowaniu ruchem drogowym staje się naturalnym kierunkiem dalszego rozwoju ITS.

---

[1] National-geographic - https://www.national-geographic.pl/nauka/nagroda-nobla-2024/

[2] Obserwator finansowy - https://www.obserwatorfinansowy.pl/tematyka/makroekonomia/trendy-gospodarcze/fenomen-chatgpt-i-jego-skutki/

1.1. Uzasadnienie wyboru tematu

Obecnie jesteśmy świadkami rewolucji w dziedzinie sztucznej inteligencji. Powstają nowe typy jednostek obliczeniowych, takie jak TPU v6, osiągające wydajność na poziomie 1836 TOPS (Tera Operations Per Second). Dynamiczny rozwój technologii AI zaczyna być ograniczany jedynie przez niedostateczną ilość sklasyfikowanych danych niezbędnych do skutecznego trenowania modeli

Systemy takie jak AlphaGo, opracowane przez DeepMind, uświadamiają nam, że maszyny mogą przekroczyć poziom ludzkich umiejętności. System AlphaGo Zero,[3] osiągnął po 3 godzinach treningu mistrzowski poziom w grze w Go, a po 70 godzinach nauki zaproponował rozwiązania przekraczające dotychczasowe ludzkie doświadczenie.

Wykorzystanie technologii AI w zarządzaniu ruchem drogowym, zwłaszcza w połączeniu z rozwojem pojazdów autonomicznych, może znacząco poprawić efektywność systemów transportowych. Dane generowane przez pojazdy autonomiczne, infrastrukturę drogową oraz inne źródła (np. urządzenia mobilne, systemy monitorujące przepływy pieszych) mogą stanowić istotną podstawę do tworzenia zaawansowanych strategii sterowania ruchem.

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.

[3] Google https://deepmind.google/discover/blog/alphago-zero-starting-from-scratch/

1.2. Cel pracy

Celem pracy jest zbadanie możliwości wykorzystania algorytmów uczenia ze wzmocnieniem (RL), takich jak algorytm Aktor-Krytyk, do optymalizacji sterowania sygnalizacją świetlną na obszarach charakteryzujących się dużym natężeniem ruchu. Przeprowadzone symulacje w środowisku SUMO pozwolą ocenić potencjał, efektywność oraz praktyczne aspekty zastosowania proponowanego rozwiązania.

1.2.1. Zakres pracy obejmuje:

- Przedmiotowy: Optymalizację sterowania sygnalizacją świetlną na skrzyżowaniach z wykorzystaniem algorytmu Aktor-Krytyk.
  Analizę oraz interpretację wyników przeprowadzonych symulacji komputerowych.
- Czasowy:
  - Analizę literatury i istniejących rozwiązań semestr 5.
  - Projektowanie i implementację algorytmu semestr 6.
  - Testowanie i analizę wyników w środowisku symulacyjnym SUMO semestr 7.
- Przestrzenny: Symulacje zostanie przeprowadzona w wirtualnym środowisku SUMO. Ruch drogowy będzie generowany syntetycznie, z uwzględnieniem scenariusza, który koncentruje się na tworzeniu zatorów drogowych.

1.2.2. Stosowane metody

- Analiza źródeł: Przegląd istniejących systemów sterowania ruchem oraz prac naukowych związanych z zastosowaniem sztucznej inteligencji w tej dziedzinie.
- Modelowanie i symulacja: Implementacja algorytmu Aktor-Krytyk w środowisku SUMO, pozwalająca na symulację sterowania sygnalizacją świetlną.
- Metody oceny efektywności: Analiza wyników symulacji, w tym pomiar opóźnień, czasu oczekiwania pojazdów, przepustowości, prędkości pojazdów na skrzyżowaniach.

1.3. Opis zawartości poszczególnych rozdziałów

Rozdział 1: Wstęp – wprowadzenie do tematu pracy, opis problemu zarządzania ruchem drogowym, cel i zakres prac, uzasadnienie wyboru tematu, cel i zakres pracy.
Rozdział 2: Sterowanie sygnalizacją świetlną w systemach zarządzania ruchem drogowym – metody sterowania sygnalizacją świetlną. Przedstawiono podstawowe strategie zarządzania ruchem, ich zalety oraz ograniczenia.
Rozdział 3: Uczenie maszynowe – podstawy uczenia ze wzmocnieniem (RL), Procesy Decyzyjne Markowa (MDP), równania Bellmana, algorytm Aktor-Krytyk (AC). W rozdziale omówiono teoretyczne podstawy niezbędne do zrozumienia zagadnień RL.
Rozdział 4: Środowisko symulacyjne – opis modelu sieci drogowej, konfiguracji ruchu i scenariusza symulacji. Zawarto szczegółowe informacje dotyczące struktury modelowanej sieci drogowej oraz sposobu generowania ruchu.
Rozdział 5: Implementacja algorytmu AC w środowisku testowym – architektura sieci neuronowej, trening modelu. Rozdział zawiera opis zastosowanych technik programistycznych oraz szczegóły procesu uczenia modelu.
Rozdział 6: Analiza działania wytrenowanego modelu sterowania ruchem – analiza wyników symulacji, porównanie z tradycyjnymi metodami sterowania. Przedstawiono również wnioski dotyczące efektywności zaproponowanego rozwiązania oraz rekomendacje dotyczące praktycznego wdrożenia.
Rozdział 7: Podsumowanie – wnioski końcowe, osiągnięcia pracy, możliwe kierunki dalszych badań. Wskazano potencjalne obszary rozwoju tematu oraz możliwości dalszych badań naukowych.

1.4. Podsumowanie

Praca stanowi połączenie teorii algorytmów sztucznej inteligencji z praktycznym ich zastosowaniem. Celem jest implementacja algorytmu Aktor-Krytyk do sterowania sygnalizacją świetlną w modelowanym środowisku SUMO.

Przeprowadzone symulacje będą stanowić cenne doświadczenie edukacyjne, umożliwiające zgłębienie złożonej tematyki algorytmów uczenia ze wzmacnianiem, sieci neuronowych oraz modelowania systemów transportowych. Projekt umożliwia 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.

2. Sterowanie sygnalizacją świetlną w systemach zarządzania ruchem drogowym

Pierwsze zastosowanie sygnalizacji świetlnej w sterowaniu ruchem drogowym miało miejsce w 1868 roku w Londynie – były to latarnie zasilane gazem. Elektryczna sygnalizacja świetlna została wprowadzona po raz pierwszy w 1914 roku w Cleveland. Do 1918 roku stosowano sygnalizatory dwukolorowe (czerwone i zielone), natomiast trójkolorowy system, zawierający również światło żółte, po raz pierwszy zastosowano 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 w zależności od panujących 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 dostosowują sterowanie do aktualnych warunków ruchu, lecz także przewidują możliwe przyszłe scenariusze, co umożliwia bardziej efektywne planowanie i podejmowanie decyzji.

2.1. Sterowanie ruchem drogowym: szczegółowy podział systemów

Poniżej przedstawiono klasyfikację systemów sterowania ruchem drogowym[4], uwzględniającą różne podejścia strukturalne, technologiczne i funkcjonalne stosowane w zarządzaniu sygnalizacją świetlną.

2.1.1. 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] Marcin Ruchaj, *Algorytmy sterowania acykliczną sygnalizacją świetlną w zatłoczonej sieci drogowej*

2.1.2. 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.

2.1.3. 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:
Wykorzystują zaawansowane modele matematyczne i statystyczne, które uczą się optymalnych strategii sterowania ruchem na podstawie danych. Do najważniejszych podejść należą:

- Uczenie przez wzmacnianie (Reinforcement Learning, RL);
- Sieci neuronowe

2.2. Krótki opis działających systemów sterowania ruchem

W rozdziale skrótowo przedstawiono najważniejsze elementy systemów sterowania ruchem drogowym – od klasycznych rozwiązań po nowoczesne, inteligentne i adaptacyjne podejścia. Skupiono się na konkretnych przykładach wdrożonych systemów.

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.

2. **SCATS** (Sydney Coordinated Adaptive Traffic System):

   SCATS - opracowany przez australijskich naukowców, 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 adaptację do warunków ruchu.

SCATS jest stosowany w wielu miastach, w tym w Polsce, gdzie został wdrożony w Rzeszowie, Łodzi i Olsztynie[5].

3. **SCOOT** (Split Cycle Offset Optimization Technique):
   SCOOT - 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 podobszary, a sterowniki w każdym podobszarze 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 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 nadal ewoluuje. Najnowsza wersja, SCOOT MC3[6], wprowadza priorytety dla autobusów i pojazdów uprzywilejowanych .

[5] 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

[6] SCOOT® Version History, Split Cycle and Offset Optimisation Technique, https://trlsoftware.com/software/intelligent-signal-control/scoot/scoot-version-history/

4. **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.

5. **GASCAP, SPPORT**[7] 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, takie jak przedłużenie fazy zielonej, są podejmowane na podstawie reguł rozmytych, które oceniają stopień dopasowania do aktualnej sytuacji. 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 10 może należeć jednocześnie do zbiorów Średnia i Długa z przynależnością 0,3. Dzięki zdolności do generalizacji logika rozmyta jest skuteczną i elastyczną metodą sterowania ruchem drogowym.

6. **PIACON 8** to metoda inteligentnego sterowania ruchem drogowym, opracowana w 2008 roku przez AGH i holenderskiego producenta automatyki, wdrożona w Lubinie. Bazuje na systemach ekspertowych oraz algorytmach optymalizacyjnych i działa na trzech poziomach: lokalnym, arterialnym i sieciowym. Wykorzystuje dane z detektorów ruchu, szacując liczba pojazdów i długości kolejek. Uwzględnia wielokryterialne podejście, analizując m.in. straty czasu i zatory, by dynamicznie dostosowywać sygnalizację do aktualnych warunków drogowych.

[7] Marcin Ruchaj, Politechnika Opolska Wydział Elektrotechniki, Automatyki i Informatyki Instytut Automatyki i Informatyki, *Algorytmy sterowania acykliczną sygnalizacją świetlną w zatłoczonej sieci drogowej*
[8] Miśkiewicz M.: ViaPIACON – polska metoda sterowania ruchem drogowym. Przegląd ITS nr 4, Warszawa 2008.

2.3. Wnioski z analizy istniejących systemów

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. Rosnąca dostępność danych sprzyja dalszej automatyzacji i optymalizacji polityk sterujących. W nadchodzących latach można oczekiwać jeszcze większego nacisku na integrację tych systemów z inteligentną infrastrukturą miejską.

Nowoczesny i wydajny system sterowania ruchem to dziś:

- większa płynność ruchu,
- skrócenie czasu przejazdu,
- zwiększenie bezpieczeństwa,
- rejestracja i analiza ruchu,
- priorytety dla komunikacji zbiorowej,
- personalizowane tras.

3. Uczenie maszynowe

Sztuczna inteligencja (AI) oraz uczenie maszynowe (ML) to dynamicznie rozwijające się dziedziny, które odgrywają kluczową rolę we współczesnych technologiach informatycznych. Za ojca sztucznej inteligencji i informatyki uznaje się Alana Turinga, który już w 1943 roku postawił fundamentalne pytanie: „Czy maszyny mogą myśleć?". Jego pionierskie prace nad maszynami obliczeniowymi dały początek koncepcji tworzenia inteligentnych systemów informatycznych.
Kilka lat później, w 1956 roku, John McCarthy wprowadził termin „sztuczna inteligencja" podczas przełomowej konferencji w Dartmouth College, uznawanej za symboliczny początek badań 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.[9]

Uczenie maszynowe dzieli się obecnie na trzy główne typy: uczenie nadzorowane, uczenie bez nadzoru oraz uczenie ze wzmocnieniem.
W uczeniu nadzorowanym model trenuje się na danych zawierających etykiety, co umożliwia rozwiązywanie zadań klasyfikacji czy regresja.
Uczenie bez nadzoru analizuje nieoznakowane dane, aby znaleźć ukryte zależności – najczęściej przez klasteryzację lub zmniejszanie liczby cech.
Z kolei uczenie ze wzmocnieniem opiera się na interakcji agenta z otoczeniem, w której model uczy się podejmować decyzje prowadzące do maksymalizacji długoterminowej nagrody.

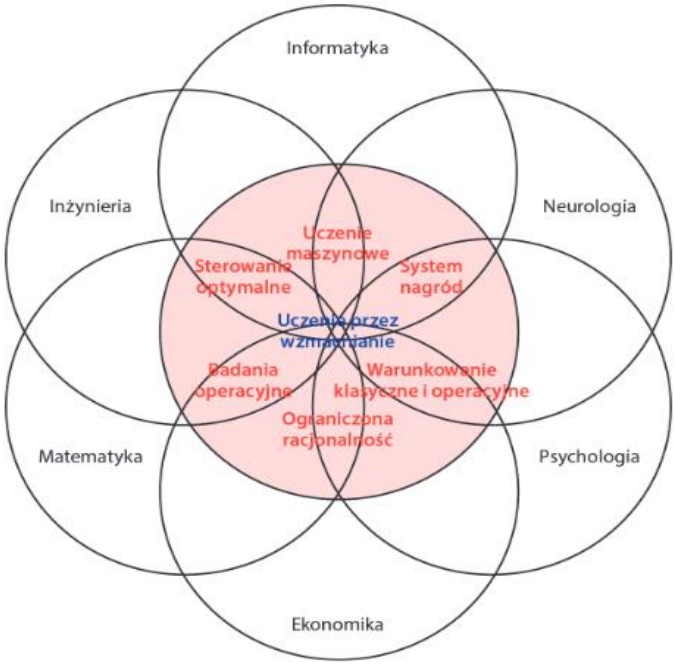
[9] Arthur Samuel, Some Studies in Machine Learning Using the Game of Checkers , https://www.cs.virginia.edu/~evans/greatworks/samuel1959.pdf

3.1. 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. Obszary nauki związane z uczeniem maszynowym



Źródło: Maxim Lapan, *Głębokie uczenie przez wzmacnianie. Praca z chatbotami oraz robotyka, optymalizacja dyskretna i automatyzacja sieciowa w praktyce*, S.31.

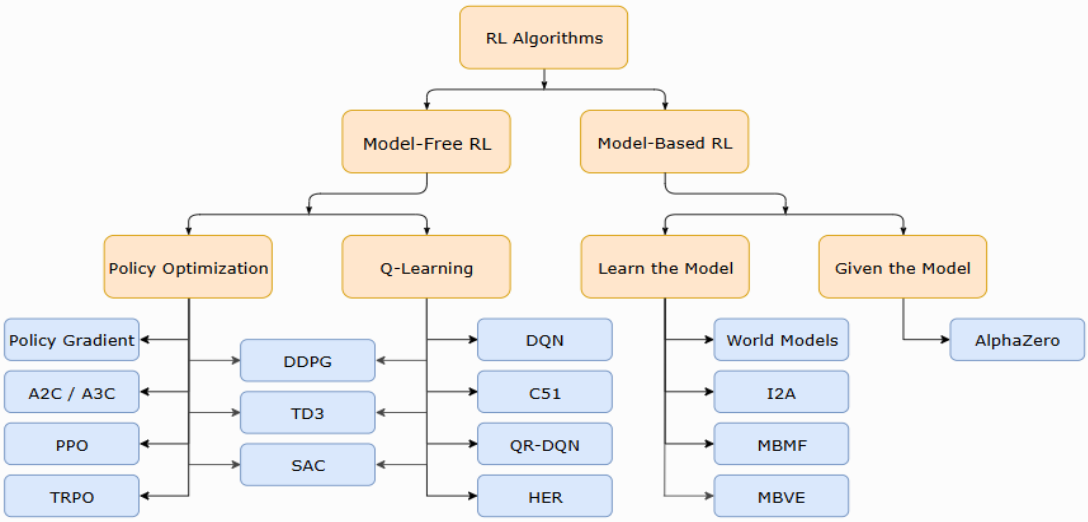
Podstawowy model uczenia przez wzmocnienie wykazuje liczne analogie do modeli psychologicznych z dziedziny warunkowania klasycznego. Eksperymenty przeprowadzone przez Iwana Pawłowa z psami demonstrują zdolność zwierząt do kojarzenia sygnałów środowiskowych, takich jak dźwięk dzwonka, z bodźcami nagradzającymi, np. jedzeniem. Pawłow określił ten mechanizm jako 'wzmocnienie', odnosząc się do bodźca nagradzającego, który wzmacniał pożądane zachowania psa (agenta). [10]

[10] Steven L. Brunton, J. Nathan Kutz, Data Driven Science & Engineering Machine Learning, Dynamical Systems, and Control (databookRL.pdf)

W ramach uczenia ze wzmocnieniem opracowano wiele algorytmów, spośród których szczególną popularność zdobyły dwa: Deep Q-Network (DQN) oraz Deep Deterministic Policy Gradient (DDPG). Oba podejścia cechują się stosunkowo prostą implementacją oraz dużą elastycznością w adaptacji do złożonych środowisk symulacyjnych. [11]

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

Rysunek 2. Taksonomia algorytmów we współczesnym uczeniu przez wzmacnianie.



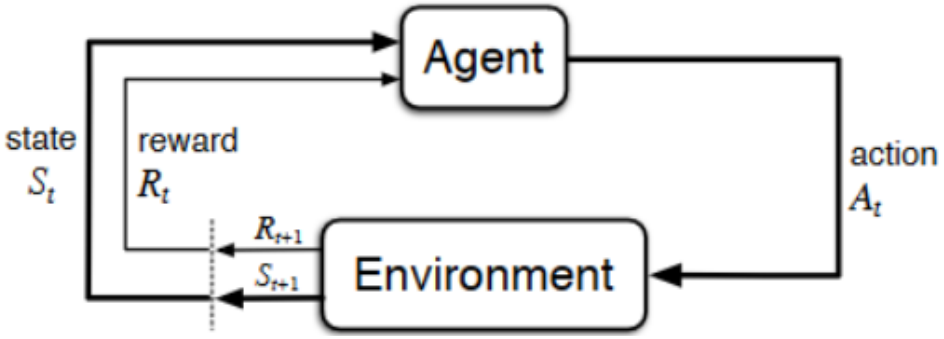
Źródło: Josha Achiama,OpenAI*; https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html*

[11] Google CLOUD, https://www.cloudskillsboost.google/focuses/10285?locale=pl&parent=catalog

3.2. Formalne podstawy i terminologia

Główne elementy uczenia przez wzmocnienie to: agent i środowisko (environment) oraz kanały interakcji między nimi, takie jak: akcje (action), nagrody (reward) i stany (state). Rysunek 3 ilustruje strukturę modelowego środowiska oraz zależności pomiędzy poszczególnymi komponentami.

Rysunek 3 Schemat blokowy algorytmu RL



Źródło: Richard S. Sutton, Andrew G. Barto, *Reinforcement Learning: An Introduction* Second edition, in progres November 5, 2017, stron 38

3.2.1. Agent i środowisko

Agent to podmiot, który wchodzi w interakcję ze środowiskiem w dyskretnych krokach czasowych t, agent znajduje się w stanie $s_t \in S$, gdzie S jest zbiorem wszystkich możliwych stanów środowiska. W każdym kroku t agent wykonuje akcję $a_t \in A$, odbiera obserwację stanu s_{t+1} oraz otrzymuje nagrodę $r_{t+1} \in R$, gdzie A jest zbiorem dostępnych akcji, a R zbiorem możliwych nagród.

Środowisko reprezentuje wszystko, co otacza agenta, dostarcza mu informacji s_{t+1} i reagując na jego działania a_t.

3.2.2. 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.

3.2.3. Obserwacje

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

3.2.4. 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.

3.3. Procesy Decyzyjne Markowa (MDP)

Procesy Decyzyjne Markowa (MDP) to matematyczny model wykorzystywany w uczeniu przez wzmocnienie. Umożliwia formalne opisanie środowiska oraz interakcji pomiędzy agentem a środowiskiem. MDP stanowi rozszerzenie klasycznego procesu Markowa poprzez wprowadzenie dodatkowych elementów, takich jak akcja i nagroda.

MDP można zdefiniować jako 5-eleentową krotkę:

$$MDP = (S,A,P,R,\gamma) \quad \text{(wzór 1)}$$

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,
$\gamma \in [0,1)$: 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 P() i R().

3.4. Funkcja przejścia P(s′|s, a)

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

$$P(s'|s,a) = P(S_{t+1} = s'|S_t = s, A_t = a) \quad \text{(wzór 2)}$$

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

3.4. Funkcja nagrody R(s,a)

Funkcja nagrody R(s,a) określa oczekiwaną wartość nagrody r_{t+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.

$$R(s,a) = \mathbb{E}(r_{t+1}|S_t = s, A_t = a) \quad \text{(wzór 3)}$$

gdzie
$\mathbb{E}[\cdot]$: Operator wartości oczekiwanej, obliczający średnią ważoną wszystkich możliwych wyników.

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

3.5. 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:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$ [12] (wzór 4)

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

3.6. Polityka

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

$$\pi(a|s) = \Pr(A_t = a|S_t = s)$$ (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).

3.7. 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.

[12] Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto (wzór 3.8), http:, //incompleteideas.net/book/RLbook2020.pdf

Równanie Bellmana dla wartości stanu (Vπ(s))

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

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi,s'\sim P}[r(s,a) + \gamma V^{\pi}(s')] \quad \text{(wzór 6)}$$

gdzie

$V^{\pi}(s)$ - wartość stanu s przy danej polityce π.
$E_{a\sim\pi,s'\sim P}[.]$ oczekiwanie (średnia wartość) po losowych zmiennych:
$a\sim\pi$ Akcjaa jest wybierana zgodnie z polityką π(a|s), czyli prawdopodobieństwem wybrania akcji a w stanie s.
$s'\sim P$: 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 ($0\leq\gamma\leq1$).
$V^{\pi}(s')$ - Wartość stanu s′, do którego przechodzi system po wykonaniu akcji a.

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

$$V^{*}(s) = \max_a \mathbb{E}_{s',\sim P}[r(s,a) + \gamma V^{*}(s')] \quad \text{(wzór 7)}$$

Określa maksymalną możliwą wartość stanu s, gdy agent działa w sposób optymalny.
W przeciwieństwie do wersji on-policy, tu dodany jest operator $\max_a$, który reprezentuje wybór akcji a maksymalizującej wartość.

Techniki wykorzystujące równania Bellmana

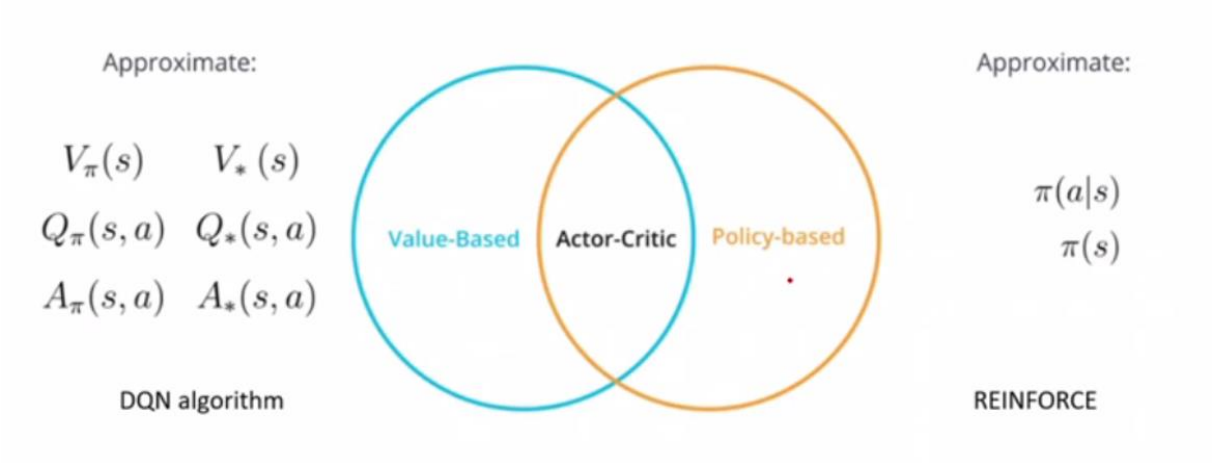
Iteracja wartości: Rekurencyjnie oblicza V(s) dla wszystkich stanów, aż do zbieżności.
Po zakończeniu procesu wyznacza optymalną politykę $\pi^{*}(s)$.
Iteracja polityki: Naprzemienne kroki oceny polityki (Vπ(s)) i jej ulepszania (π′(s)).
Równania Bellmana są używane w obu etapach.

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

3.8. Algorytm Aktor-Krytyk (Actor-Critic)

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.

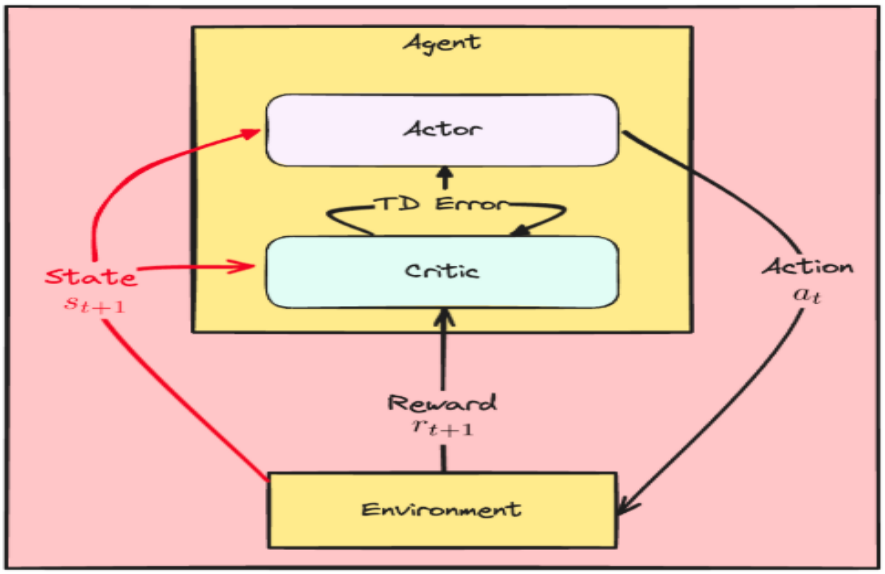
Rysunek 4. Połączenie podejść Value-Based i Policy-Based w algorytmie Actor-Critic.



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

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 Schemat działania algorytmu Aktor Krytyk



Źródło: 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 δ_t , który służy jako sygnał wzmocnienia do ulepszania polityki w aktorze.

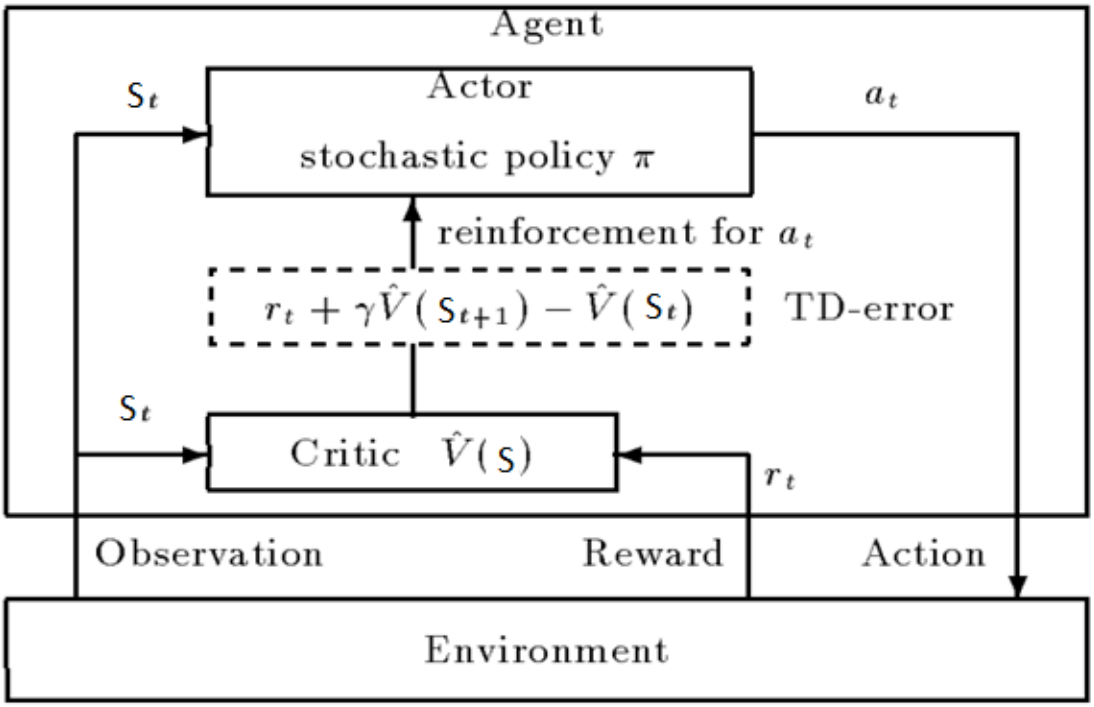
$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \qquad \text{(wzór 8)}$$

gdzie:

δ_t to błąd różnicy czasowej (TD-error),
r_t 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. Dynamika algorytmu Aktor-Krytyk



Źródło: Kimura H., Kobayashi S., *An Analysis of Actor/Critic Algorithms Using Eligibility Traces: Reinforcement Learning with Imperfect Value Function*, ACM Digital Library, 1998.

Opis formalny algorytmu uwzględniającego wykorzystanie sieci neuronowych zaczerpnięty z „Reinforcement Learning: An Introduction”[13]

Wejście:

$\pi(a|s,\theta)$, różniczkowalna funkcja prawdopodobieństwa wyboru akcji a w stanie s.

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

Współczynniki uczenia: $\alpha_\theta>0$, $\alpha_w>0$.

Inicjalizacja:

Parametry polityki: $\theta \in R$.

Wagi funkcji wartości: $w \in R$.

ALGORYTM:

Pętla nieskończona (dla każdego epizodu):

1. Inicjalizuj s pierwszy stan epizodu.
2. $I \leftarrow 1$

Pętla czasowa (dopóki s nie jest terminalny):

3. Wybierz akcję $a \sim \pi(\cdot|s,\theta)$.
4. Wykonaj akcję a, zaobserwuj nowy stan s' i nagrodę r.
5. Oblicz błąd TD (δ):

$$\delta \leftarrow r + \gamma V(s_{t+1}, w) - V(s_t, w)$$ (nawiązanie do wzoru 8)

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

6. Zaktualizuj wagi funkcji wartości:

$$w \leftarrow w + \alpha_w I \delta \nabla V(s, w)$$

7. Zaktualizuj parametry polityki:

$$\theta \leftarrow \theta + I \delta \nabla \ln \pi(a \mid s, \theta)$$

8. Zaktualizuj współczynnik wpływu I:

$$I \leftarrow \gamma I$$

9. Przejdź do następnego stanu:

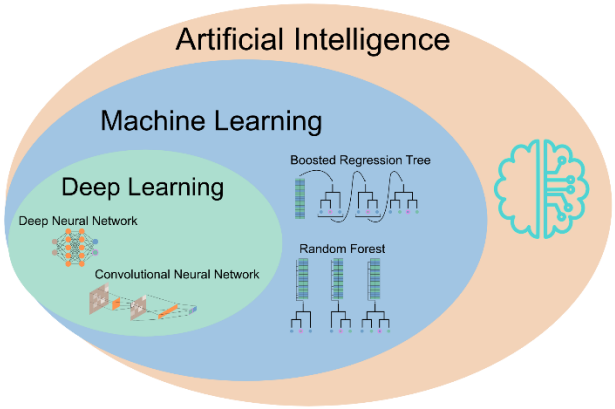
$$s \leftarrow s_{t+1}$$

[13] Reinforcement Learning: An Introduction Second edition ****Complete draft**** March 11, 2018 Richard S. Sutton and Andrew G. Barto

3.9. Deep Learning w kontekście RL

Uczenie głębokie (Deep Learning, DL) to dziedzina sztucznej inteligencji, która korzysta z wielowarstwowych sieci neuronowych (rysunek 7), 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 7. Schemat głębokiej sieci neuronowej wykorzystywanej w uczeniu maszynowym.



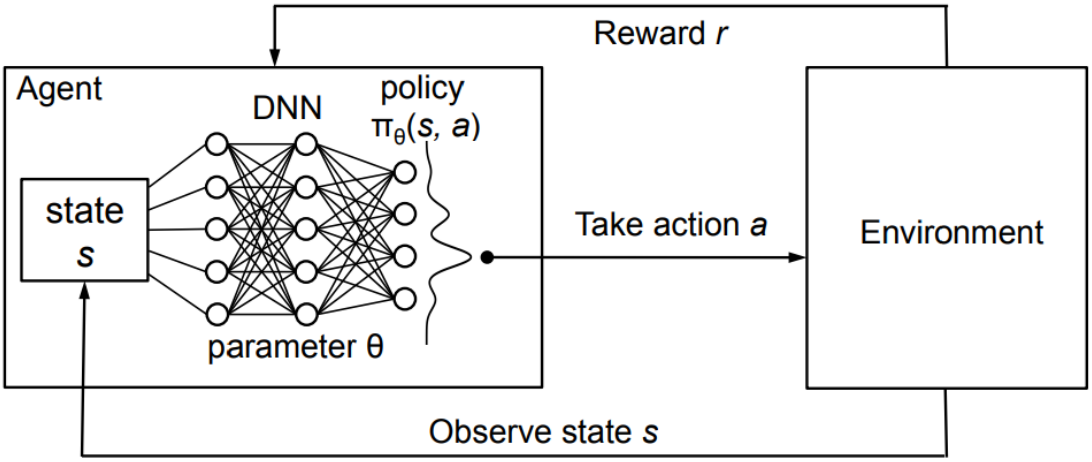
Źródło: Pichler M., Hartig F., *Machine Learning and Deep Learning with R*, 2023.

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.[14] 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.

[14] Nature, Human-level control through deep reinforcement learning, https://www.nature.com/articles/nature14236

Na rysunku 8, pokazano ogólny schemat zastosowania sieci neuronowej (DNN) do predykcji polityki π_θ, (algorytm Policy Gradient) gdzie agent wchodzi w interakcję ze środowiskiem. Należy zwrócić uwagę na symbol θ będący parametrem sieci.

Rysunek 8. schemat zastosowania sieci neuronowej (DNN) do predykcji polityki



Źródło: 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

4. Środowisko symulacyjne

Rozdział przedstawia sposób przygotowania środowiska testowego w symulatorze SUMO. Opisano strukturę sieci drogowej, parametry ruchu pojazdów, zastosowany scenariusz oraz konfigurację plików niezbędnych do przeprowadzenia symulacji.

4.1. Pakiet SUMO

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. Scenariusze 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) [15], 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++, JavaMATLAB

[15] SUMO TraCI https://sumo.dlr.de/docs/TraCI/Protocol.html

4.2. Pliki konfiguracyjne testowanego modelu

Najważniejsze elementy modelu, do którego zostanie zaimplementowane sterowanie oświetleniem drogowym, zostały określone w kilku kluczowych plikach konfiguracyjnych.

4.2.1. Plik 2x2.net.xml

Ten plik jest rdzeniem modelu, stanowi mapę drogową dla symulacji ruchu.
Główny element <net> definiuje całą sieć drogową. Znajdują się w nim atrybuty takie jak opisy 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.

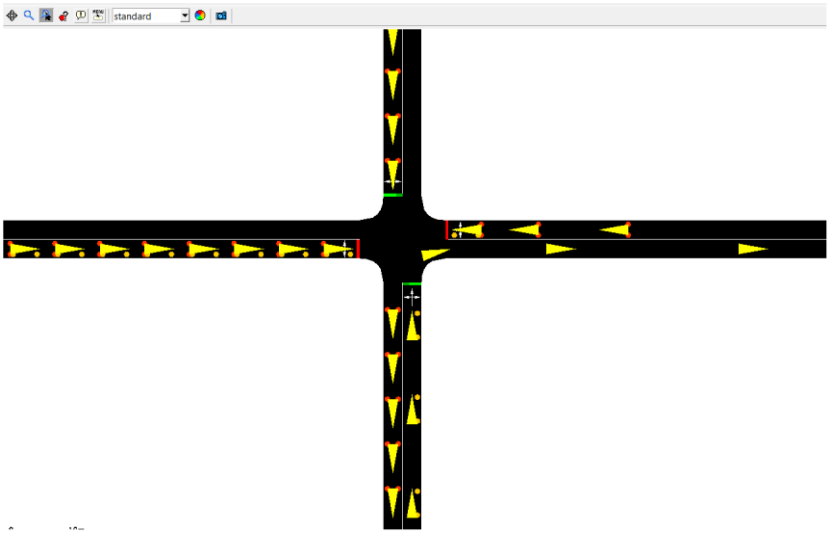
Logika sterowania ruchem na skrzyżowaniach:

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.

Rysunek 9. Przykładowe skrzyżowanie z sygnalizatorami w symulatorze SUMO



Źródło: Opracowanie własne

4.2.2. Plik 2x2.rou.xml

W tym pliku określono parametry generowania i przepływu pojazdów, takie jak:

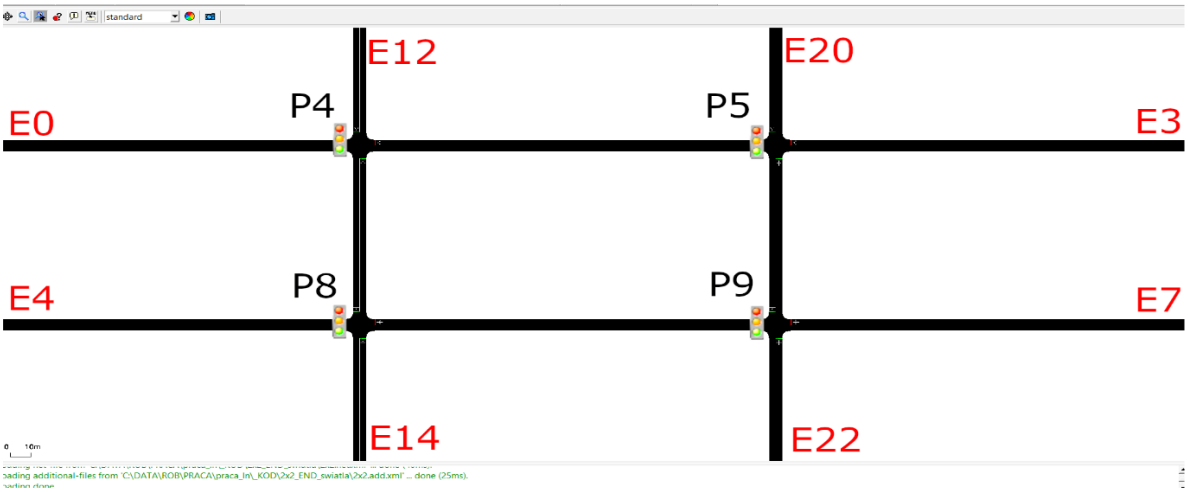
- Prawdopodobieństwo pojawienia się pojazdu na określonej trasie,
- Parametry takie jak „departLane” (wartość „free”, co oznacza dowolny pas startowy) oraz „departSpeed” ustawione na „random” (co odzwierciedla naturalne różnice w prędkościach pojazdów),
- Okres symulacji (od 0 do 3600 kroków).

Dzięki temu model odzwierciedla zmienność i losowość pojawiających się pojazdów, co odzwierciedla realistyczną dynamiki ruchu drogowego.

4.3. Szczegółowy opis modelu

Model testowy symuluje ruch pojazdów w sieci drogowej, obejmując węzły, drogi, ograniczenia prędkości, trajektorie pojazdów oraz sposób ich generowania (Rysunek 10). Struktura sieci składa się z węzłów wylotowych, skrzyżowań sterowanych sygnalizacją oraz węzłów wewnętrznych. Drogi podzielono na zewnętrzne 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. Schemat połaczeń drogowych w modelu



Źródło: opracowanie i obliczenia własne

4.3.1. Węzły

Zdefiniowano 3 rodzaje węzłów:

a) 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)
b) Skrzyżowań sterowanych sygnalizacją (typ „traffic_light”):
   Występują w punktach: P4, P5, P8, P9 (łącznie 4 skrzyżowania)
c) 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.

4.3.2. Drogi

Drogi zostały podzielone według dwóch kryteriów.

a) 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.
b) Drogi wewnętrzne (definiujące szczegółowy przebieg ruchu wewnątrz skrzyżowań):
   Dla każdego skrzyżowania sterowanego stworzono 16 dróg wewnętrznych w celu określanie obciążenia występującego na skrzyżowaniu.

4.3.3. Prędkości na drogach

W celu zwiększenia realizmu na drogach występują różne ograniczenia prędkości

a) 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.
b) 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

4.3.4. Trajektorie ruchu

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

4.3.5. Parametry generowania pojazdów:

Atrybut probability określa szansę pojawienia się pojazdu na danym przepływie w każdej jednostce czasu.
W modelu występują dwa poziomy intensywności:

a) Probability 0.1 – oznacza wyższą częstotliwość generowania pojazdów (około 0.1 pojazdu na sekundę, co daje średnio około 360 pojazdów na godzinę),
b) 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.

4.4. 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.

5. Implementacja algorytmu AC w środowisku testowym

Rozdział przedstawia implementację algorytmu Aktor-Krytyk w symulatorze SUMO z wykorzystaniem interfejsu TraCI. Opisano strukturę sieci neuronowej, sposób integracji modelu z symulacją oraz mechanizmy podejmowania decyzji i uczenia. Zaprezentowano również przebieg procesu trenowania modelu oraz analizę uzyskanych wyników.

5.1. TraCI

W implementowanym modelu symulacji ruchu drogowego wykorzystano interfejs TraCI (Traffic Control Interface), który umożliwia komunikację pomiędzy kodem sterującym a symulatorem SUMO. Dzięki TraCI możliwe jest dynamiczne sterowanie sygnalizacją świetlną oraz pobieranie danych z symulowanego środowiska.

W skrypcie (generowanie_modeluA7_01.py) algorytm Aktor-Krytyk, poprzez interfejs TraCI łączy się z SUMO.
SUMO jest uruchamiane w trybie serwera za pomocą funkcji:

```
traci.start([SUMO_BINARY, "-c", CONFIG_FILE])
```

gdzie SUMO_BINARY określa tryb pracy, a CONFIG_FILE wskazuje plik konfiguracyjny symulacji (2x2.sumocfg).
Skrypt modyfikuje poprzez interfejs fazy świateł na skrzyżowaniach (P4, P5, P8, P9).
Fazy są wybierane na podstawie polityki, 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, dane przekazywane są do modelu uczenia maszynowego jako wejściowy stan środowiska.
W procesie uczenia modelu Aktor-Krytyk, wybór akcji odbywa się na podstawie prawdopodobieństwa wyznaczonego przez warstwę aktora. Pomimo zastosowania warstwy softmax generującej rozkład prawdopodobieństwa, dodatkowo zaimplementowano mechanizm epsilon-greedy, który pozwala na losowy wybór akcji z ustalonym prawdopodobieństwem w początkowej fazie uczenia. Zabieg ten zwiększa eksplorację przestrzeni akcji i przyczynia się do uniknięcia lokalnych minimów w procesie uczenia.

Podczas symulacji skrypt wykonuje kroki symulacji SUMO, przechodząc do następnych iteracji modelu:

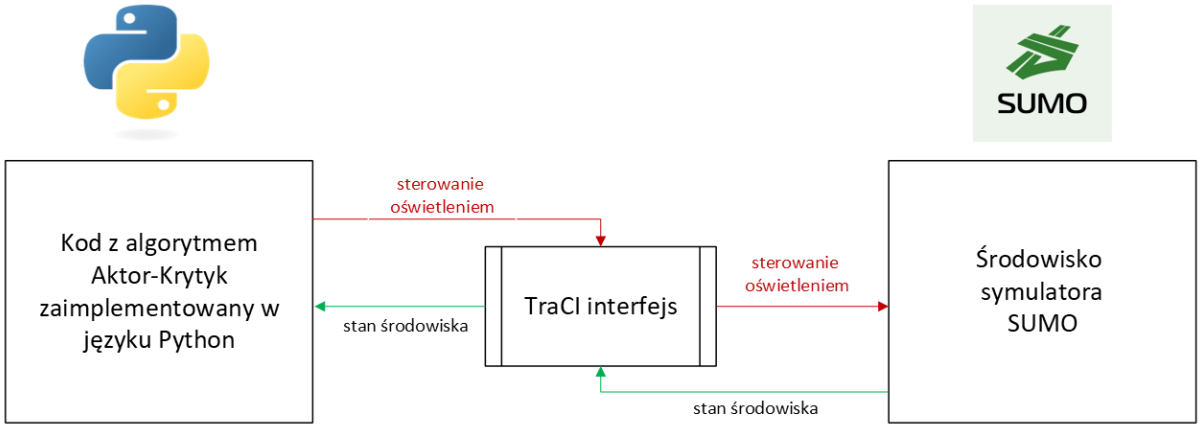
```
traci.simulationStep()
```

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

```
traci.close()
```

Symulator SUMO jako środowisko interaktywne (Rysunek12), które w czasie rzeczywistym reaguje na decyzje podejmowane przez model uczący się, pozwala to na badanie efektywności różnych strategii sterowania ruchem drogowym.

Rysunek 11. Schemat blokowy przepływu komunikacji Skrypt-TraCI-SUMO

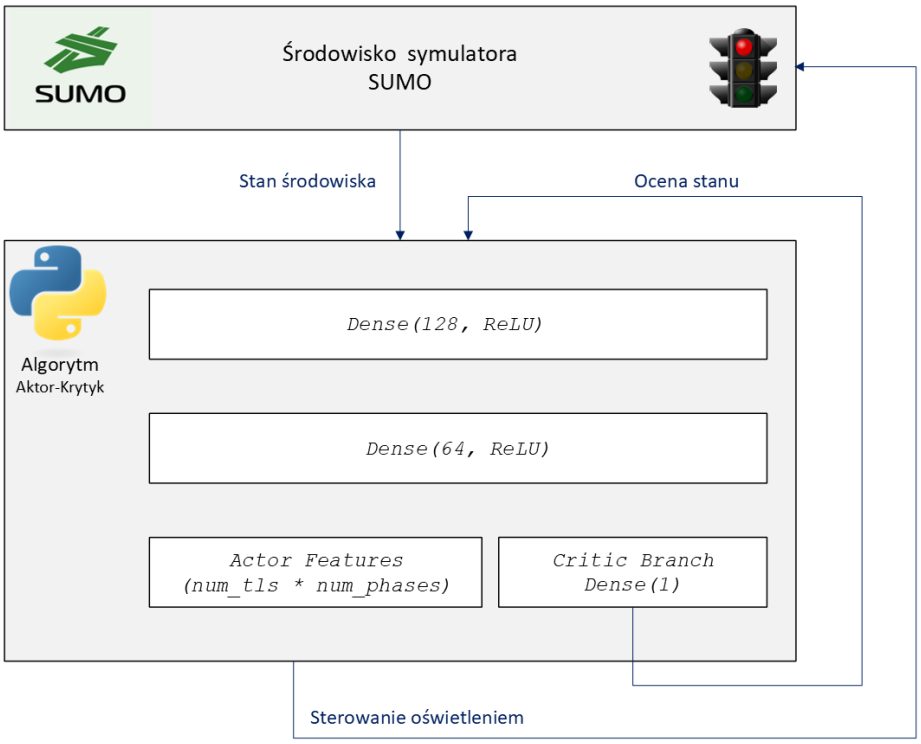


Źródło: Opracowanie i obliczenia własne.

5.2. Sieć neuronowa

Zastosowana sieć neuronowa składa się z trzech głównych komponentów: warstwy wspólnej odpowiedzialnej za ekstrakcję cech ze stanu środowiska, warstwy aktora generującej decyzje sterujące oraz warstwy krytyka oceniającej jakość danego stanu. Takie rozwiązanie pozwala na skuteczne łączenie percepcji sytuacji drogowej z podejmowaniem decyzji w czasie rzeczywistym.

Rysunek 12. Schemat warstw wykorzystanej sieci neuronowej.



Źródło: Opracowanie i obliczenia własne.

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 neuronów, 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.

```
self.common = tf.keras.Sequential([
layers.Dense(128, activation="relu", kernel_initializer="he_normal"),
layers.Dense(64, activation="relu", kernel_initializer="he_normal")
        ])
```

2. 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 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")
```

3. 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.
    - Wartość wyjściowa reprezentuje estymację funkcji wartości V(s), czyli oczekiwanej skumulowanej nagrody z aktualnego stanu przy założeniu stosowania bieżącej polityki.

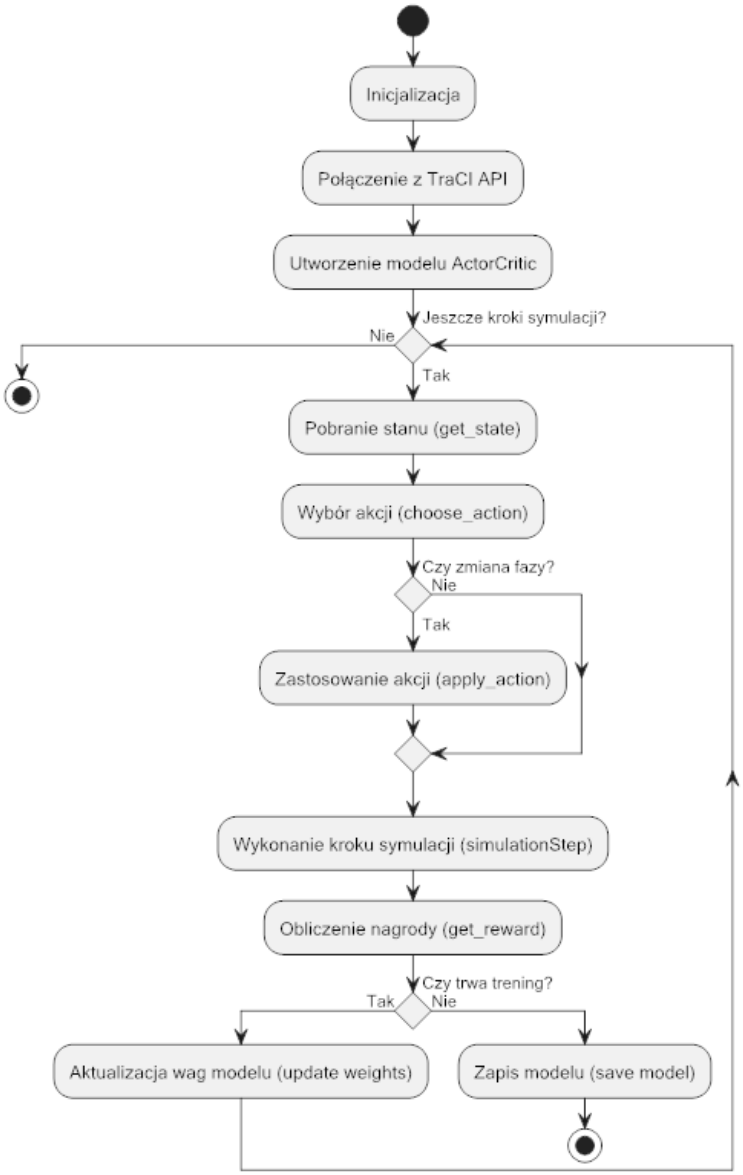
```
self.critic = layers.Dense(1, name="critic")
```

Zaproponowana architektura sieci neuronowej, obejmująca warstwę wspólną, aktora oraz krytyka, charakteryzuje się wysoką efektywnością przetwarzania dynamicznych danych wejściowych. Umożliwia elastyczne podejmowanie decyzji sterujących na podstawie generowanego rozkładu prawdopodobieństwa oraz zapewnia stabilną ocenę wartości stanów. Synergia tych komponentów sprzyja szybkiemu i trafnemu reagowaniu modelu na zmieniające się warunki ruchu drogowego.

5.3. Implementacja algorytmu systemu sterowania
(skrypt generowanie_modeluA7_01.py)

Przedstawiony kod realizuje adaptacyjne sterowanie ruchem drogowym z wykorzystaniem symulacji w środowisku SUMO oraz agenta uczenia ze wzmocnieniem opartego na architekturze Aktor-Krytyk. Integracja modelu symulacyjnego z algorytmem uczenia pozwala na iteracyjną optymalizację ustawień faz sygnalizacyjnych oraz przeprowadzenia procesu uczenia. Rysunek 13 przedstawia blokowy schemat programu.

Rysunek 13. Schemat blokowy programu uczącego model AI



Źródło: Opracowanie i obliczenia własne.

5.3.1. Konfiguracja i inicjalizacja

Definicja stałych konfiguracyjnych:

1. SUMO_BINARY: Ustawione na "sumo". Wyłączona wizualizacja.
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 3)

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

5.3.2. Architektura Modelu Aktor-Krytyk

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 ($4 \times 3 = 12$). Generuje rozkład prawdopodobieństwa wyboru poszczególnych faz dla każdego skrzyżowania.
3. Warstwy krytyka: Pojedyncza warstwa Dense, która ocenia jakość danego stanu (przewidując jego wartość).

Taka architektura umożliwia jednoczesne podejmowanie decyzji sterujących i ocenę ich jakości, co pozwala modelowi efektywnie uczyć się optymalnych strategii.

5.3.3. Pozyskiwanie Stanu Symulacji

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 = 400, max_waiting_time = 10000), a następnie łączone w jeden wektor stanu.

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

5.3.4. 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.

Wybrane akcje decydują o tym, która z trzech możliwych sekwencji świateł zostanie zastosowana na danym skrzyżowaniu.

5.3.5. Aplikacja Akcji w Symulacji

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.

5.3.6. Obliczanie Nagrody

Funkcja get_reward():

1. Oblicza nagrodę na podstawie całkowitej długości kolejek (suma pojazdów zatrzymanych) oraz łącznego czasu oczekiwania.
2. 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.

5.3.7. Proces Treningu

Funkcja train_actor_critic() realizuje główną pętlę treningową, obejmującą:

1. Uruchomienie symulacji przez interfejs traci. Dla każdego z 300 epizodów wykonywanych jest 6000 kroków symulacji.
2. Co 10 kroków symulacji podejmowaną jest decyzję o zmianie faz.

3. Po każdej zmianie faz symulacja wykonuje krok (traci.simulationStep()), pobierany jest nowy stan, a nagroda jest ponownie obliczana.
4. Aktualizacja modelu obejmuje obliczenie wartości docelowej przy użyciu bieżącej nagrody oraz przewidywanej wartości stanu następnego (z dyskontowaniem przy gamma = 0.95). Dzięki mechanizmowi GradientTape obliczane są straty aktora i krytyka, które następnie są minimalizowane przy użyciu optymalizatora Adam. Gradienty są przycinane dla stabilności procesu uczenia.
5. Po zakończeniu każdego epizodu, wyświetlana jest całkowita uzyskana nagroda, a wagi modelu zapisywane są na dysku.

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

5.3.8. Integracja z Symulacją SUMO

Cały kod opiera się na interakcji z symulacją SUMO poprzez moduł TraCI, który umożliwia:

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

Szczegóły tego procesu zostały opisane w paragrafie 5.1

Podsumowanie

Kod (generowanie_modeluA7_01.py) integruje symulator SUMO z agentem uczenia ze wzmocnieniem opartym na architekturze Aktor-Krytyk. System ten:

1. Pobiera dane dotyczące długości kolejek oraz czasu oczekiwania na skrzyżowaniach,
2. Wykorzystuje model Actor-Critic (TensorFlow) do wyboru optymalnych faz sygnalizacji świetlnej,
3. Aktualizuje strategię sterowania ruchem na podstawie uzyskiwanych nagród,
4. Wykorzystuje mechanizm epsilon-greedy, który wspomaga eksplorację przestrzeni decyzji.

Dzięki takiej integracji możliwe jest dynamiczne i adaptacyjne sterowanie ruchem drogowym oraz przeprowadzanie procesu uczenia modelu w środowisku symulacyjnym.

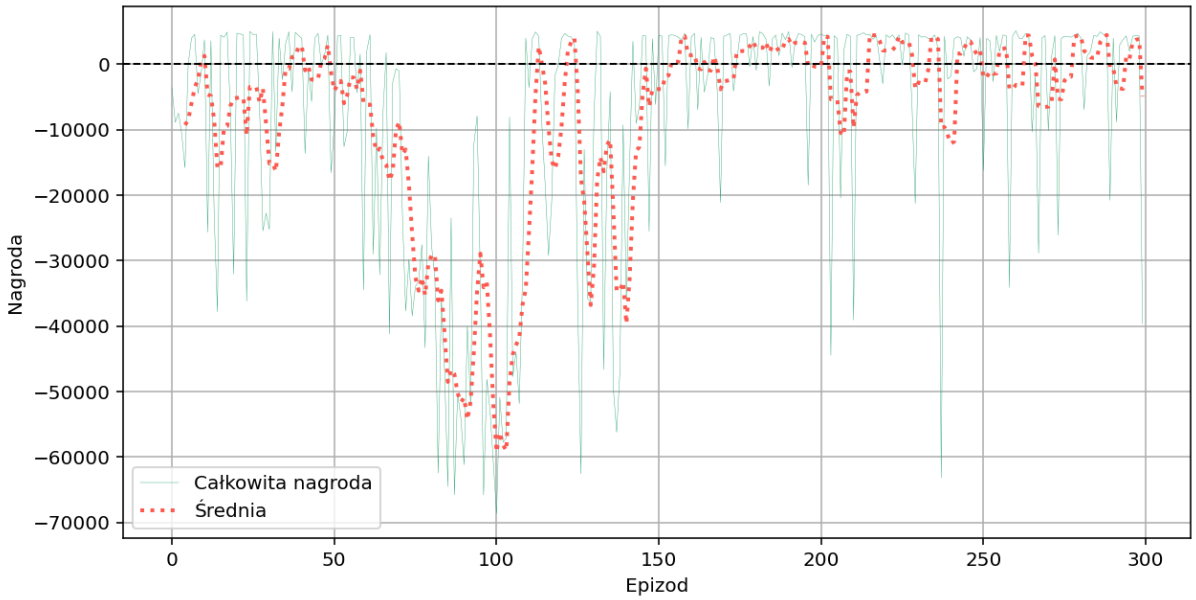
5.4. Trenowanie modelu (generowanie_modeluA7_01.py)

Model AI został trenowany w chmurze przy użyciu aplikacji Google COLAB, dzięki czemu czas nauki algorytmu został skrócony do około pięciu godzin. Podczas wielokrotnych epizodów agent uczył się coraz skuteczniej zarządzać ruchem drogowym w środowisku SUMO, poprawiając politykę działania.

W kodzie zastosowano mechanizm malejącego współczynnika ε (epsilon-greedy), który początkowo miał wartość 0.1, a następnie wraz z kolejnymi epizodami stopniowo spadał, co zapewniło płynne przechodzenie od eksploracji (losowy wybór akcji) do eksploatacji (wybór akcji zgodnie z wyuczoną polityką). Dodatkowo wykorzystano metodę klipowania gradientów co usprawniło proces uczenia.

Pomimo niestabilności w środkowej fazie treningu, agent wykazał zdolność do poprawy swojej polityki. Zwiększająca się częstotliwość epizodów z dodatnimi nagrodami.

Rysunek 14. Przebieg procesu uczenia modelu



Źródło: Opracowanie i obliczenia własne

Wykres (Rysunek 14) przedstawia całkowitą nagrodę uzyskiwaną przez agenta w kolejnych epizodach treningu. Dane obejmują łącznie 299 epizodów. Dla czytelności, oprócz surowych wartości, zastosowano również średnią. Takie wygładzenie pozwala wychwycić ogólny trend procesu.

Analiza przebiegu uczenia

Na początku treningu agent otrzymywał głównie bardzo niskie nagrody, co wskazuje na losowe, nieoptymalne działanie. W pierwszych ~20 epizodach wystąpiły także pojedyncze przypadki wysokich nagród. Między 20 a 100 epizodem agent naprzemiennie osiągał wysokie i bardzo niskie nagrody. Średnia krocząca również wykazywała w tym zakresie silne fluktuacje, co świadczy o słabej polityce która w wyniku eksploracji ulegała ciągłym zmianom i celowym zakłóceniom.

Po około 150 epizodzie nastąpiła zauważalna poprawa. Więcej epizodów kończyło się pozytywnymi nagrodami, a średnia krocząca stopniowo rosła. Od tego momentu nastąpiła powolna stabilizacja co wskazuje, że algorytm wypracował efektywne polityki działania.

Tabela 1. Wartości nagród zebrane podczas trenowania modelu.

Średnia nagroda (dla wszystkich epizodów):	≈ -9 360.95
Maksymalna nagroda:	5090.55
Minimalna nagroda:	-68 651.33
Odchylenie standardowe nagród:	≈ 19 495.57

Źródło: Opracowanie i obliczenia własne

Dane z tabeli 1 potwierdzają, że proces uczenia rozpoczął się od chaotycznego eksplorowania przestrzeni strategii, ale z czasem agent nauczył się podejmować coraz bardziej efektywne decyzje, co przełożyło się na stabilniejsze, wysokie wartości nagród.

6. Analiza działania wytrenowanego modelu sterowania ruchem

Wytrenowany model został poddany testowi (test_modeluA7_01_pluswykresy.py) podczas którego zebrano dane na temat środowiska. Zgromadzone wyniki symulacji pozwoliły ocenić wydajność modelu AI. Celem tych czynności było określenie, w jakim stopniu model AI spełnia założenia dotyczące zapewnienia płynności ruchu w zmieniających się warunkach.

Do porównania działania wygenerowanego modelu z innymi strategiami sterowania (systemem stałoczasowy oraz system zoptymalizowany przez SUMO), wykorzystano trzy kluczowe wskaźniki:

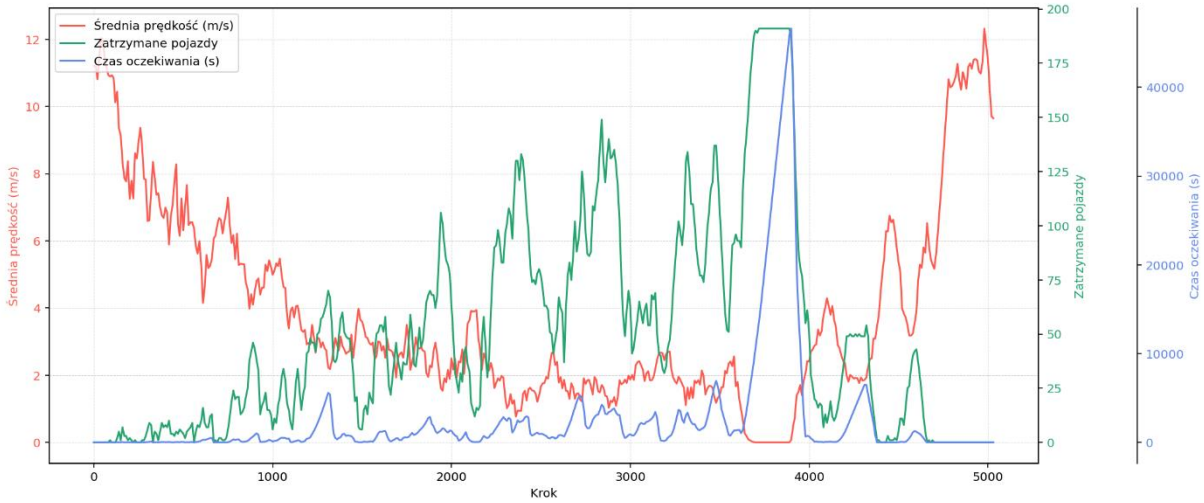
- średnią prędkość pojazdów,
- liczbę zatrzymań
- średni czas oczekiwania.

Wskaźniki te umożliwiły kompleksową ocenę zachowania systemu zarówno w krótkim, jak i dłuższym horyzoncie czasowym.

6.1. Analiza wskaźników jakości sterowania ruchem (wysztkowJednymAI.py)

Wykres (Rys. 15) przedstawia przebieg badanych wskaźników w przedziałach czasu. Możemy tu prześledzić, dynamikę działania modelu w trakcie całej symulacji.

Rysunek 15. Wykres wybranych parametrów symulacji SUMO sterowanej modelem AI



Źródło: Opracowanie i obliczenia własne

Średnia prędkość pojazdów jest dość niska oscyluje wokół wartości 3,86 m/s, co wynika z celowego symulowania warunków przeciążenia - środowisko testowe zostało zaprojektowane tak, by model był poddawany ekstremalnym przeciążeniom ruchu drogowego.
Liczba zatrzymanych pojazdów osiągnęła średnio 50,4 a czas oczekiwania to średnio 2536,3 s.

Tabela 2. Statystyki wskaźników uzyskanych w trakcie symulacji.

Wskaźnik	Średnia	Max. wartość	Odchylenie standardowe
Średnia prędkość (m/s)	3,86	12,80	2,94
Zatrzymane pojazdy	50,40	191,00	49,35
Czas oczekiwania (s)	2536,35	47 445,00	6594,54

Źródło: Opracowanie i obliczenia własne

Analiza danych z tabeli 2 wykazuje dużą zmienność oraz wysokie wartości krańcowych parametrów. Należy podkreślić, że są one wynikiem środowiska testowego, mającego na celu sprawdzenie elastyczności i efektywności modelu AI.

Wytrenowany model wykazuje:

- zdolność adaptacji do ekstremalnych warunków,
- dynamiczną reakcję na duże zatory i przestoje.

Wyżej wymienione właściwości świadczą o potencjale wytrenowanego modelu który radzi sobie z ekstremalnymi warunkami środowiska z uwzględnieniem niestabilnych i trudnych warunków ruchu.

6.2. Porównanie wytrenowanego modelu AI z innymi systemami sterowania

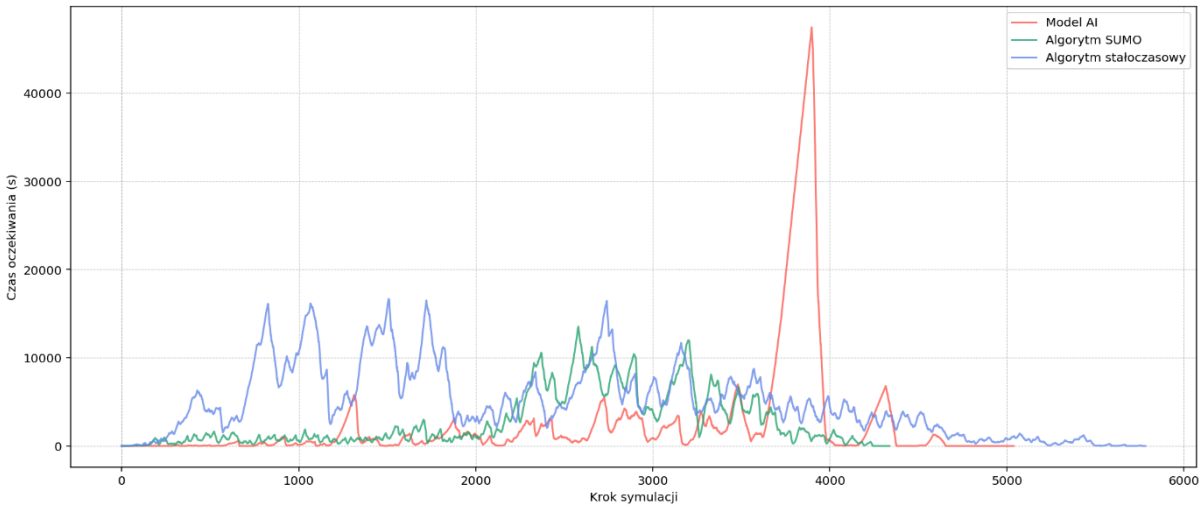
W celu obiektywnej oceny skuteczności wytrenowanego modelu przeprowadzono porównanie z dwoma alternatywnymi strategiami:

1. Optymalizator SUMO - system dynamiczny, będący domyślnym algorytmem sterowania ruchem w oprogramowaniu SUMO. Oparty o dynamiczną zmianę zarówno czasów jak i sekwencji świetlnych.
2. Sterowanie stałoczasowe, w którym każda faza sygnalizacji świetlnej ma ustaloną, równą długość, niezależnie od natężenia ruchu.

6.2.1. Czas zatrzymania pojazdów w czasie symulacji (oczekiwanie w czasie.py)

Jednym z kluczowych wskaźników oceny efektywności systemu sterowania jest czas w którym pojazdy pozostają nieruchome w wyniku zatorów i przestojów.

Rysunek 16. Porównanie algorytmów sterowania pod względem czasu oczekiwania pojazdów.



Źródło: Opracowanie i obliczenia własne

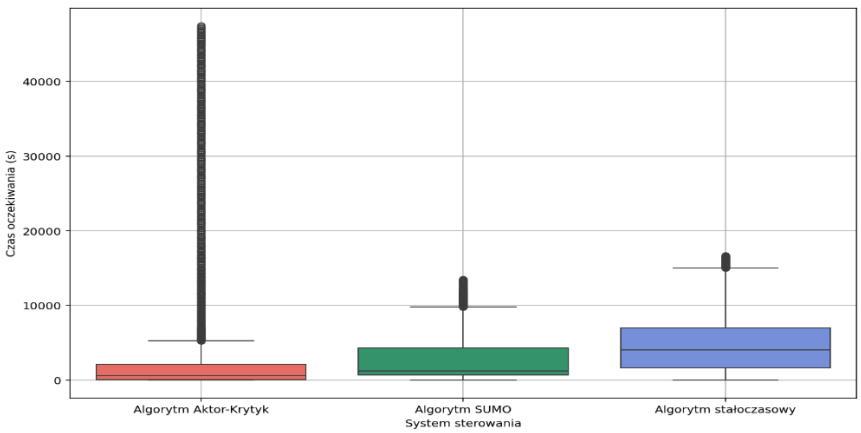
Analizując przebieg wykresu przedstawionego na rys. 16, można zauważyć istotne różnice w zachowaniu poszczególnych systemów sterowania ruchem.

Model AI utrzymuje zrównoważony poziom czasu oczekiwania przez większość trwania symulacji. Mimo że średnia wartość jest wysoka (głównie z powodu ekstremalnych wartości w kilku krokach), wykres ukazuje zdolność modelu do szybkiej reakcji i adaptacji. System „optymalizator SUMO'" osiąga dobre wyniki – w wielu przypadkach porównywalne z wytrenowanym modelem. Jednak w sytuacji wysokim natężeniu ruchu obserwuje się wyraźne wydłużenie czasu oczekiwania, co może świadczyć o ograniczonej elastyczności. System stałoczasowy wypada zdecydowanie najsłabiej – charakteryzuje się wyższymi i niestabilnymi wartościami czasu oczekiwania, co wynika z jego sztywnej, nieadaptacyjnej logiki działania.

W analizowanych danych zaobserwowano pojedynczy pik wartości czasu oczekiwania na poziomie około 47 000 sekund w jednym z kroków symulacji. Jest to wyraźna anomalia. Ważne jest jednak, że model AI wykazał zdolność samoregulacji i po tak ekstremalnym piku szybko przywrócił wartości do stabilnego poziomu.

Porównanie algorytmów wskazuje, że model AI osiąga najniższe wartości czasu oczekiwania. Tendencję tą wyraźnie widać na wykresie skrzynkowym rysunek 17. Mimo pojedynczej bardzo wysokie wartości, większość przypadków ma krótszy czas oczekiwania niż w pozostałych badanych systemach.

Rysunek 17. Porównanie algorytmów; rozkład czasu oczekiwania pojazdów

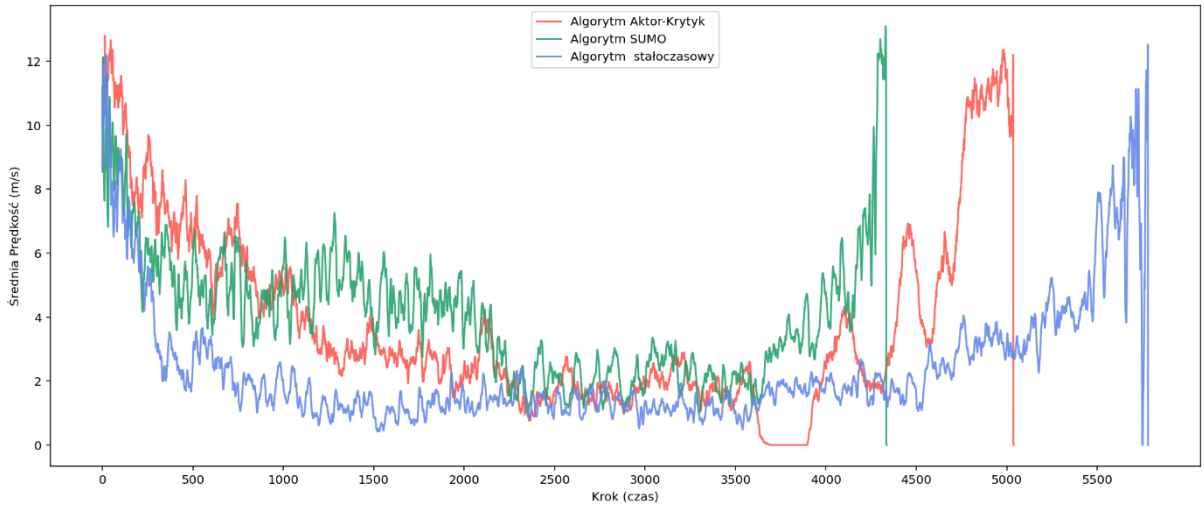


Źródło: Opracowanie i obliczenia własne

6.2.2. Średnią prędkość pojazdów podczas symulacji (wykres_predkosci.py)

Średnia prędkość ruchu drogowego stanowi kolejny istotny wskaźnik płynności oraz efektywności systemu transportowego.

Rysunek 18. Porównanie algorytmów sterowania, średnia prędkość pojazdów



Źródło: Opracowanie i obliczenia własne

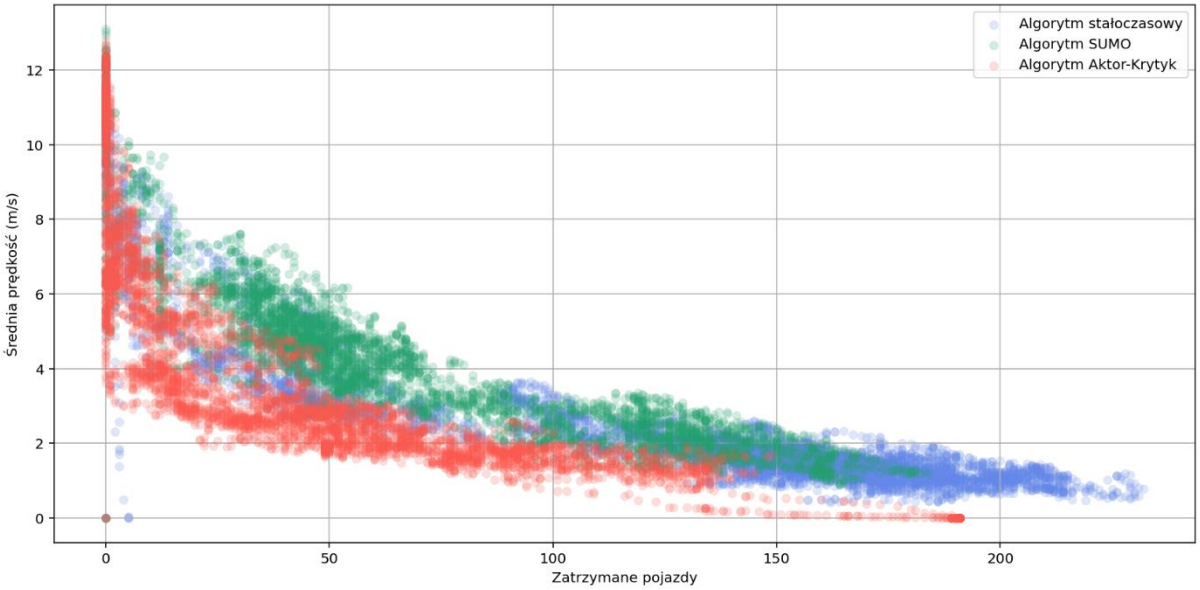
.

Z analizy danych (Rysunek 19) wynika, że najwyższą średnią prędkość osiąga system „optymalizator SUMO", utrzymując wartość na poziomie około 4,06 m/s.
Model AI zapewnia niższą prędkość 3,86 m/s.
Sterowanie stałoczasowe charakteryzuje się najniższą średnią prędkością (2,42 m/s) co potwierdza jego ograniczoną efektywność w dynamicznym środowisku miejskim.

6.2.3. Analiza zależności między liczbą zatrzymanych pojazdów a średnią prędkością

Zestawienie liczby zatrzymanych pojazdów ze średnią prędkością jazdy pozwala na kompleksową ocenę wydajności i płynności działania systemu sterowania ruchem.

Rysunek 20. Porównanie algorytmów, zależności między liczbą zatrzymanych pojazdów a średnią prędkością



Zródło: Opracowanie i obliczenia własne

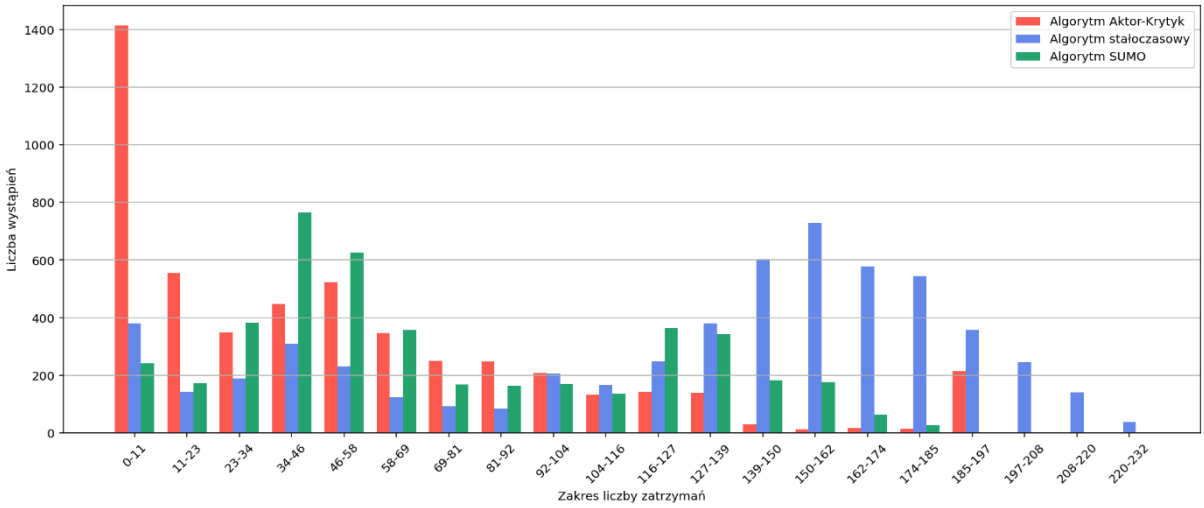
Patrząc na wykres punktowy (Rysunek 19) widzimy, że Model AI koncentruje większość swoich punktów w lewym górnym rogu wykresu - czyli tam, gdzie liczba zatrzymanych pojazdów jest niska, a średnia prędkość wysoka.
Algorytm stałoczasowy wykazuje dużą koncentrację punktów w prawym dolnym obszarze, gdzie zatrzymanych pojazdów jest wiele, a prędkość niska. Taki rozrzut wskazuje na nieefektywne zarządzanie ruchem.
Algorytm SUMO zajmuje obszar pośredni, przy czym jego punkty są nieco bardziej rozproszone niż w przypadku AI. Oznacza to, że system ten działa lepiej niż podejście stałoczasowe, ale nie osiąga takiej stabilności i efektywności jak wyuczony.

Z wykresu wyraźnie wynika, że model AI skutecznie utrzymuje optymalną równowagę między niską liczbą zatrzymanych pojazdów a wysoką średnią prędkością. Analiza rozrzutu potwierdza, że adaptacyjność algorytmu przekładają się nie na lepsze wartości średnie w dynamicznym środowisku.

6.2.4. Rozkład liczby zatrzymanych pojazdów w różnych systemach sterowania

Analiza rozkładu liczby zatrzymań (Rys.20) pozwala szczegółowo ocenić, jak często i w jakim zakresie dochodzi do przestojów w ruchu drogowym w poszczególnych systemach sterowania. Zamiast skupiać się na wartościach średnich, podejście to uwzględnia całą strukturę danych, umożliwiając identyfikację niekorzystnych fragmentów symulacji.

Rysunek 21. Porównanie algorytmów, rozkład częstości zatrzymań pojazdów



Źródło: Opracowanie i obliczenia własne

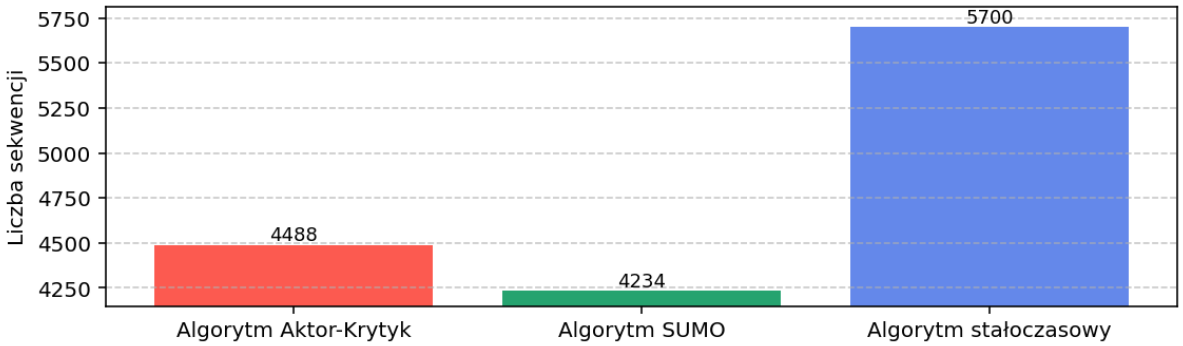
Model AI charakteryzuje się najkorzystniejszym rozkładem – dominują przedziały z niską liczbą zatrzymań (0–10, 10–20). Oznacza to, że model skutecznie minimalizuje tworzenie się kolejek, utrzymując wysoką płynność ruchu.
System stałoczasowy posiada nieefektywny rozkład – znaczna liczba przypadków zatrzymań znajduje się w środkowych i wyższych przedziałach (140–180 i więcej). Wskazuje to na częste tworzenie się zatorów i długotrwałe zatrzymywanie pojazdów.
Optymalizator SUMO prezentuje rozsądny kompromis – jego rozkład jest przesunięty w kierunku większej liczby zatrzymań niż w modelu AI, lecz znacznie korzystniejszy niż w podejściu sekwencyjnym.

Wyniki analizy stanowią kolejne potwierdzenie, że testowany modelu nie tylko ograniczają liczbę zatrzymań, ale również zapewniają większą stabilność działania.

6.2.5. Liczba sekwencji symulacji potrzebna do całkowitego opróżnienia skrzyżowań z pojazdów.

Całkowita liczba kroków symulacji, po których ostatni pojazd opuszcza skrzyżowania, stanowi kolejny wskaźnik efektywności systemu. Parametr ten pozwala ocenić, jak skutecznie dana polityka radzi sobie z rozładowaniem ruchu w skali całego środowiska - od momentu rozpoczęcia symulacji aż do całkowitego wyczyszczenia sieci z pojazdów.

Rysunek 22. Porównanie algorytmów sterowania, liczba kroków symulacji wymaganych do całkowitego opuszczenia skrzyżowań przez wszystkie pojazdy



Źródło: Opracowanie i obliczenia własne

Na wykresie (Rys.21) widać, że optymizator SUMO najszybciej rozładował natłok pojazdów, co oznacza, że system funkcjonuje bardzo efektywnie.
System AI potrzebował zauważalnie większej liczby sekwencji.
System stałoczasowy osiągnął najgorszy wynik, wynika to z baraku adaptacji do zmieniającej się sytuacji na skrzyżowaniach.

Analiza liczby sekwencji potwierdza, że choć „optymalizator SUMO" działa najefektywniej w idealnych warunkach, to model AI wykazuje akceptowalne właściwości adaptacyjne.

6.3. Analiza zalet i wad zastosowanego modelu AI

Zalety modelu AI (Aktor-Krytyk)

1. Wysoka adaptacyjność do dynamicznych warunków ruchu. Model prawidłowo reaguje na zmieniające się warunki drogowe, wykazując zdolność do adaptacji w sytuacjach przeciążenia lub kryzysu.
2. Stabilne i przewidywalne zachowanie. Pomimo pojedynczych anomalii, model szybko stabilizuje działanie, co świadczy o dobrym mechanizmach samoregulacji.
3. Minimalizacja liczby zatrzymań pojazdów. Histogramy i rozkłady potwierdzają, że model generuje najwięcej przypadków z niską liczbą zatrzymań – dominują przedziały 0–10 i 10–20. To dowód na utrzymywanie płynności ruchu.
4. Lepsze rezultaty w rozproszeniu przestojów. W analizie zależności między zatrzymaniami a prędkością, model AI najczęściej pojawia się w strefie „niskie zatrzymania - wysoka prędkość.

Wady modelu AI (Aktor-Krytyk)

1. Wysokie wartości skrajne. Model wykazywał ekstremalny pik oczekiwania rzędu 47 445 s, co wpływa na ogólną średnią. Takie zachowanie może mieć znaczenie w kontekście komfortu użytkownika.
2. Większa liczba sekwencji do pełnego rozładowania. W porównaniu z optymalizatorem SUMO, wytrenowany model potrzebuje większej liczby cykli, by całkowicie rozładować skrzyżowania - może to wskazywać na mniej agresywną politykę sterowania.
3. Niższa średnia prędkość względem algorytmu SUMO. Model osiąga średnio 3,86 m/s, w porównaniu do 4,06 m/s w systemie optymalnym. To oznacza, że prędkość pojazdów nie jest maksymalizowana.

7. Podsumowanie

W przedstawionej pracy zaprezentowano koncepcję wykorzystania algorytmu Aktor-Krytyk do adaptacyjnego sterowania sygnalizacją świetlną w ruchu drogowym. W tym celu przygotowano model sieci drogowej w symulatorze SUMO, obejmujący cztery kluczowe skrzyżowania wyposażone w sygnalizację świetlną. Następnie, poprzez interfejs TraCI, zaimplementowano komunikację między SUMO a modułem uczenia maszynowego w języku Python. W efekcie pozwoliło to na wytrenowanie modelu sterującego zmianami faz sygnalizacji.

7.1. Główne elementy pracy:

Część teoretyczna.
Przedstawiono podstawowe i zaawansowane systemy sterowania ruchem drogowym, od rozwiązań stałoczasowych, przez adaptacyjne (SCATS, SCOOT), aż po najnowsze podejścia wykorzystujące logikę rozmytą i elementy sztucznej inteligencji. Zaprezentowano korzyści płynące ze stosowania sterowania scentralizowanego i hierarchicznego w dużych aglomeracjach miejskich.
Przedstawiono również formalne podstawy algorytmów uczenia ze wzmocnieniem – w tym Procesy Decyzyjne Markowa (MDP) oraz równania Bellmana – wskazując, w jaki sposób algorytm Aktor–Krytyk łączy zalety metod opartych na wartościach z metodami opartymi na polityce.

Budowa środowiska testowego w SUMO.
Przygotowano model sieci drogowej z ośmioma węzłami wylotowymi i czterema skrzyżowaniami sterowanymi sygnalizacją. Sieć uwzględnia różnorodne drogi, różne ograniczenia prędkości oraz przepływy ruchu, w tym losowe generowanie pojazdów z ustalonym rozkładem prawdopodobieństwa. W efekcie uzyskano realistyczne warunki symulacyjne, pozwalające ocenić działanie algorytmu.

Implementacja algorytmu aktor–krytyk

Zaimplementowano model oparty na architekturze sieci neuronowej, składającej się z warstw wspólnych oraz dwóch warstw wyjściowych: aktora, odpowiedzialnego za wybór akcji, oraz krytyka, oceniającego wartość stanu.

Zintegrowano model z symulatorem SUMO przy użyciu interfejsu TraCI, umożliwiając analizę bieżących obserwacji środowiska (takich jak długości kolejek i czasy oczekiwania pojazdów) oraz wybór ustawień faz sygnalizacji świetlnej, które są natychmiast wprowadzane do symulacji.

W ramach treningu dążono do maksymalizacji skumulowanej nagrody, definiowanej m.in. przez skrócenie średniego czasu postoju oraz zmniejszenie łącznej długości kolejek pojazdów na skrzyżowaniach.

7.2. Rezultaty i wnioski

Wyniki przeprowadzonych eksperymentów potwierdzają, że zastosowanie algorytmu Aktor–Krytyk do adaptacyjnego sterowania sygnalizacją świetlną przyniosło wymierne korzyści w porównaniu z rozwiązaniami stałoczasowymi. W czterowęzłowej sieci drogowej symulowanej w środowisku SUMO zaobserwowano istotne skrócenie czasu oczekiwania pojazdów oraz ograniczenie powstawania zatorów.

Zaproponowana architektura łączy część oceniającą (krytyk) z częścią decyzyjną (aktor) i wykorzystuje błąd różnicy czasowej (TD-error), by szybciej korygować politykę sterowania. Dzięki temu zmniejsza się ryzyko nadmiernego faworyzowania poszczególnych wlotów skrzyżowań, a ruch drogowy rozkłada się bardziej równomiernie.

W miarę rozbudowy sieci drogowej i zwiększania liczby faz sygnalizacji zauważalnie rosną jednak wymagania obliczeniowe. W takich warunkach niezbędne może być optymalizowanie architektury sieci neuronowej lub skorzystanie z bardziej wydajnych wariantów algorytmu (A2C, A3C). Dodatkowo kluczowe znaczenie ma właściwe zbalansowanie systemu kar i nagród, tak aby uwzględniał długość kolejek, czas oczekiwania oraz wskaźniki świadczące o płynności ruchu. Starannie dobrany model przyznawania nagród pozwala uniknąć częstych lokalnych minimów, zapewniając stabilniejszy i skuteczniejszy proces uczenia.

7.2. Ograniczenia i wyzwania

- *Skalowalność:* Model został przetestowany w środowisku symulacyjnym obejmującym cztery skrzyżowania. Przeniesienie rozwiązania na skalę rzeczywistego miasta, gdzie sieci drogowe są znacznie większe, stanowi istotne wyzwanie obliczeniowe i organizacyjne.
- *Założenia o ruchu:* Ruch pojazdów generowany jest losowo na podstawie ustalonego rozkładu prawdopodobieństwa. Choć takie podejście pozwala na testowanie różnych scenariuszy, nie odzwierciedla w pełni złożonych wzorców ruchu drogowego występujących w rzeczywistości.
- *Brak mechanizmów stabilizujących eksplorację:* W obecnej wersji modelu nie zaimplementowano mechanizmów zapobiegających zbyt długiemu utrzymywaniu jednej fazy sygnalizacji. Może to prowadzić do utknięcia agenta w lokalnym optimum i ograniczenia eksploracji przestrzeni decyzyjnej.

7.3. Możliwe kierunki dalszych prac

- *Priorytetyzacja pojazdów uprzywilejowanych i komunikacji zbiorowej*.
  Rozszerzenie modelu o mechanizmy nadawania priorytetu pojazdom uprzywilejowanym (np. karetką, straży pożarnej) oraz transportowi publicznemu.
- *Zastosowanie metod wieloagentowych (multi-agent systems).*
  Traktowanie każdego skrzyżowania jako autonomicznego agenta uczącego się współpracy z sąsiednimi węzłami. Podejście to może przyczynić się do poprawy globalnej koordynacji sygnalizacji świetlnej w rozległych sieciach miejskich.
- *Wielokryterialna optymalizacja.*
  Uwzględnienie dodatkowych kryteriów podczas procesu optymalizacji, takich jak: emisja zanieczyszczeń, zużycie paliwa, koszty ekonomiczne,
- *Ujęcie ruchu pieszych i rowerzystów.*
  Integracja modelu z ruchem pieszym i rowerowym, który ma istotny wpływ na dynamikę ruchu oraz bezpieczeństwo w obszarach miejskich.
- *Wykorzystanie danych z systemów zewnętrznych*.
  Integracja danych pochodzących z: pojazdów autonomicznych, sieci telefonii komórkowej umożliwi bardziej precyzyjne monitorowanie poziomu nasycenia ruchem – zarówno drogowym, jak i pieszym.

Znaczenie praktyczne

Adaptacyjne algorytmy sterowania sygnalizacją świetlną mają szansę poprawić płynność ruchu, skrócić czas podróży i zredukować emisję spalin. W dłuższej perspektywie rozwój takich systemów może wpisywać się w ideę inteligentnych miast (smart cities), w których infrastruktura drogowa dynamicznie dostosowuje się do zapotrzebowania.

Praca ta łączy teorię algorytmów uczenia ze wzmocnieniem z praktycznym zastosowaniem w dziedzinie sterowania ruchem drogowym. Wykorzystanie symulatora SUMO umożliwiło wielokrotne testy i zbieranie danych w różnych warunkach. Uzyskane wyniki sygnalizują istotny potencjał metod opartych na sztucznej inteligencji w optymalizacji ruchu drogowego, jednocześnie zaznaczając konieczność dalszych badań w zakresie skalowalności i uwzględniania bardziej złożonych czynników, aby takie doświadczenia przenieść na poziom rzeczywistych sieci miejskich.

8. Bibliografia

Książki

1 Brunton S.L., Kutz J.N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge: Cambridge University Press, 2019

2 Kimura H., Kobayashi S. *An Analysis of Actor/Critic Algorithms Using Eligibility Traces: Reinforcement Learning with Imperfect Value Functions*. W: *Proceedings of the 15th International Conference on Machine Learning*, 1998, s. 278–286.

3 Lapan M. *Głębokie uczenie przez wzmacnianie: praca z chatbotami oraz robotyka, optymalizacja dyskretna i automatyzacja sieciowa w praktyce*. Wydanie II. Gliwice: Helion, 2022.

4 Sutton R.S., Barto A.G. *Reinforcement Learning: An Introduction*. Cambridge (Mass.); London: The MIT Press, 2015

Czasopisma

1 Mao H., Alizadeh M., Menache I., Kandula S. *Reinforcement learning with policy represented via DNN*. *Proceedings of the 15th ACM Workshop on Hot Topics in Networks „*HotNets '16*"*. ACM, 2016. [Online]. https://people.csail.mit.edu/hongzi/content/publications/DeepRM-HotNets16.pdf

2 Miśkiewicz M. *ViaPIACON – polska metoda sterowania ruchem drogowym*. „Przegląd ITS", nr 4, 2008.

3 Mnih V., Kavukcuoglu K., Silver D. et al. *Human-level control through deep reinforcement learning*., „Nature", vol. 518, no. 7540, 2015, pp. 529–533.

4 Samuel A.L. *Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress*., „IBM Journal of Research and Development, vol. 11, no. 6", 1967

Netografia

1 “A Brief Review of Actor Critic Methods.” YouTube, 2022. [Film]. https://www.youtube.com/watch?v=aODdNpihRwM, (data odczytu: 5 kwietnia 2025)

2 Achiam J. *Spinning Up in Deep Reinforcement Learning*. OpenAI, 2018. https://spinningup.openai.com/, (data odczytu: 5 kwietnia 2025)

3 Carayol T. *Deep Reinforcement Learning in Python*. DataCamp, 2025. https://campus.datacamp.com/, (data odczytu: 5 kwietnia 2025)

4 Ciesielski M. *Fenomen ChatGPT i jego skutki*. *Obserwator Finansowy*, 08.03.2023. [Online]. https://www.obserwatorfinansowy.pl/tematyka/makroekonomia/trendy-gospodarcze/fenomen-chatgpt-i-jego-skutki/, (data odczytu: 5 kwietnia 2025)

5 Google Cloud Skills Boost. *Reinforcement learning, czyli uczenie się przez wzmacnianie: Qwik Start*. 27.09.2023., https://www.cloudskillsboost.google/, (data odczytu: 5 kwietnia 2025)

6 Okoń T. *Podsystem Sterowania Ruchem Sprint/ITS/SCATS*. ITS Polska, 2022. [Online]. https://www.itspolska.pl/, (data odczytu: 5 kwietnia 2025)

7 Ruchaj M. *Algorytmy sterowania acykliczną sygnalizacją świetlną w zatłoczonej sieci drogowej*. Praca doktorska. Politechnika Wrocławska, 2012. Dolnośląska Biblioteka Cyfrowa, https://www.dbc.wroc.pl/ , (data odczytu: 5 kwietnia 2025)

8 Silver D. *AlphaGo Zero: Starting from scratch*. DeepMind Blog, 18.10.2017, https://deepmind.google/discover/blog/alphago-zero-starting-from-scratch/, (data odczytu: 5 kwietnia 2025)

9 STRL Software. *SCOOT® Version History – Split Cycle and Offset Optimisation Technique*., https://trlsoftware.com/products/traffic-control/scoot/scoot-version-history/, (data odczytu: 5 kwietnia 2025)

10 Stradowski J. *Nagroda Nobla 2024: pełna lista laureatów*. *National-Geographic.pl*, 14.10.2024. https://www.national-geographic.pl/nauka/nagroda-nobla-2024/, (data odczytu: 5 kwietnia 2025)

11 *SUMO. SUMO TraCI Protocol Documentation.*, *https://sumo.dlr.de/docs/TraCI/Protocol.html*, (data odczytu: 5 kwietnia 2025)

Spis Rysunków