Wyższa Szkoła Zarządzania i Bankowości

w Krakowie

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

ITS (Intelligent transportation system)  
RL

Actor-Critic (A2C),

SUMO (Simulation of Urban Mobility)

Deep Neural Network, DNN, sieci głebokie

δ, Δ - delta

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 generują 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) co może mieć tragiczne konsekwencje. 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)[[1]](#footnote-1), 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.

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”[[2]](#footnote-2). 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,[[3]](#footnote-3) 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).[[4]](#footnote-4) Rozwój technologii AI zaczyna być blokowany przez ograniczoną ilość danych niezbędnych do trenowania modeli.   
 Systemy takie jak AlphaGo**,** opracowane przez DeepMind, pokazują że maszyny mogą przekroczyć poziom ludzkich umiejętności. System AlphaGo Zero, [[5]](#footnote-5)osiągnął po 3 godzinach treningu mistrzowski poziom w grze w Go, a po 70 godzinach nauki zaproponował rozwiązania przekraczające dotychczasowe ludzkie doświadczenie.

Tradycyjne metody sterowania ruchem drogowym, mimo swojej skuteczności, nadal mają potencjał do rozwoju i optymalizacji. Nowoczesne systemy drogowe, w połączeniu z technologią autonomicznych pojazdów, mogą znacząco poprawić efektywność i adaptacyjność sterowania. Informacje generowane przez te 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 trendami w dziedzinie AI, potencjałem technologii sieci neuronowych oraz próbą wykorzystania wiedzy z zakresu algorytmów uczenia maszynowego.

.

1. **Cel pracy**

Celem pracy jest zbadanie, w jaki sposób algorytmy RL, takie jak Advantage Actor-Critic (A2C), mogą zostać wykorzystane do sterowania sygnalizacją świetlną na obszarach o dużym natężeniu ruchu. Symulacje przeprowadzone w środowisku SUMO pozwolą na ocenę potencjału oraz efektywności rozwiązania.

Zakres pracy

Zakres pracy obejmuje:

1. Przedmiotowy: Optymalizacja sterowania sygnalizacją świetlną na skrzyżowaniach przy użyciu algorytmu Actor-Critic.   
   Analiza zgromadzonych danych wyników symulacji.
2. Czasowy:   
   - Analizę literatury i istniejących rozwiązań w marca 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 w sposób syntetyczny, 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 w dynamicznych warunkach ruchu.
3. Metody oceny efektywności: Analiza wyników symulacji, w tym pomiar opóźnień, czasu oczekiwania pojazdów oraz przepustowości, ilość zużytego paliwa i wyemitowanego CO2.

Opis zawartości poszczególnych rozdziałów pracy

* Rozdział 4: Sterowanie ruchem świetlnym, omówienie podstaw zarządzania ruchem drogowym oraz metod sterowania sygnalizacją świetlną.
* Rozdział 5 : Uczenie maszynowe; analiza literatury naukowej, opis procesów RL, AC.
* Rozdział 6: Pakiet symulatora SUMO.

Podsumowanie

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

Przeprowadzone prace i symulacje stanowią doświadczenie edukacyjne, wprowadzające w tematykę algorytmów uczenia ze wzmacnianiem, modelowania systemów transportowych oraz wykorzystania intermodalnego symulatora ruchu drogowego SUMO.

Uzyskane wnioski mogą stać się podstawą dla dalszego pogłębiania wiedzy w zakresie ITS.

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.[[6]](#footnote-6) Do roku 1918 sygnalizatory były dwukolorowe, tj. wyposażone w światło czerwone i zielone. Trójkolorową, sygnalizacja zawierającą również światło żółte, zainicjował Londynie.

Sterowanie sygnalizacją ewoluowało od systemów stałoczasowych do systemów zmiennoczasowych. Systemy stałoczasowe działałaką 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 ruchu 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.

Sterowanie Ruchem Drogowym: Szczegółowy Podział Systemów

Zaawansowane technologie i różnorodne podejścia do sterowania sygnalizacją świetlną pozwalają na zwiększenie przepustowości, redukcję opóźnień oraz minimalizację wpływu na środowisko. Poniżej przedstawiono podział systemów sterowania ruchem drogowym.

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 komputer 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.
  + Przykład: Systemy w dużych miastach.

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.

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.
  + Przykład: Proste sygnalizacje świetlne w małych miastach lub na mniej uczęszczanych drogach.
* 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.

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 sztucznej inteligencji:

Wykorzystują metody uczenia maszynowego, takie jak:

* + - Uczenie przez wzmacnianie (Reinforcement Learning, RL):  
      Algorytmy uczą się optymalnych strategii sterowania na podstawie interakcji z rzeczywistym środowiskiem.
    - Sieci neuronowe:  
      Pozwalają na analizę złożonych zależności w danych o ruchu drogowym.
    - Programowanie dynamiczne.

Przykłady:

* + - Systemy adaptacyjne wykorzystujące algorytmy Q-learning lub Advantage Actor-Critic (A2C).
    - Metody multi-agentowe, gdzie różne skrzyżowania zarządzane są przez współpracujących agentów.

Krótki opis działających systemów sterowania ruchem

Urban Traffic Control System (UTCS) to inicjatywa Departamentu Transportu USA, rozwijana od lat 70. XX wieku, obejmująca cztery generacje strategii sterowania ruchem drogowym[[7]](#footnote-7):

* Pierwsza generacja: Oparta na historycznych danych o ruchu, z planami sterowania zmienianymi co 15 minut.
* Druga generacja: Wykorzystuje prognozowane dane ruchowe, aktualizując plany co 5 minut.
* Trzecia generacja: Skraca czas aktualizacji do 3-5 minut i optymalizuje dodatkowo długość cyklu.
* Czwarta generacja: Oparta na aktualizacjach w czasie rzeczywistym, obliczając moment zmiany fazy sygnalizacji w każdym cyklu.

Ewolucja tych 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. System stosuje mechanizmy głosowania do wyboru odpowiednich planów offsetów i koordynacji między podsystemami.

SCATS jest stosowany w wielu miastach, w tym w Polsce, gdzie został wdrożony w Rzeszowie, Łodzi i Olsztynie[[8]](#footnote-8).

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[[9]](#footnote-9), 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**[[10]](#footnote-10) 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 [[11]](#footnote-11)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, które zapewniłyby płynność ruchu uwzględniając nietypowe sytuacje takiej jak kolizje czy remonty.

Przedstawione metody optymalizacji koncentrują się na poprawie efektywności i bezpieczeństwa transportu. Oto korzyści płynące z :

1. **Skrócenie czasu przejazdu** – Optymalizacja ruchu drogowego pozwala zmniejszyć czas potrzebny na pokonanie określonej trasy, co ma istotne znaczenie dla zarówno kierowców, jak i użytkowników komunikacji zbiorowej.
2. **Większa płynność ruchu** – Lepsze zarządzanie ruchem eliminuje korki i zmniejsza liczbę nieprzewidzianych zatrzymań pojazdów.
3. **Zwiększenie bezpieczeństwa** – Poprawa organizacji ruchu zmniejsza ryzyko kolizji i wypadków drogowych.
4. **Monitorowanie ruchu** – Systemy zarządzania ruchem umożliwiają bieżące śledzenie sytuacji na drogach, co wspiera podejmowanie szybkich decyzji w razie utrudnień.
5. **Priorytet dla pojazdów uprzywilejowanych i komunikacji zbiorowej** – Nadanie priorytetu odpowiednim pojazdom skraca ich czas przejazdu, co ma szczególne znaczenie dla służb ratunkowych i komunikacji miejskiej.
6. **Ograniczenie zużycia paliwa i emisji spalin** – Płynniejszy ruch i skrócone postoje zmniejszają zapotrzebowanie na paliwo oraz ograniczają emisję zanieczyszczeń do atmosfery.
7. **Personalizowane planowanie trasy** – Dzięki bieżącym informacjom o ruchu, dostarczanym za pośrednictwem znaków zmiennej treści (VMS) czy aplikacji internetowych, użytkownicy mogą wybierać optymalne trasy.
8. **Dostęp do danych statystycznych** – Systemy te umożliwiają gromadzenie danych o ruchu, co wspiera analizę i dalsze usprawnienia.

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. Jednocześnie wdrażanie takich rozwiązań wpływa pozytywnie na postrzeganie skuteczności działań władz lokalnych w zakresie poprawy bezpieczeństwa ruchu drogowego.

Głównym celem systemów ATMS (Advanced Traffic Management Systems) jest maksymalizacja bezpieczeństwa i efektywności ruchu, co obejmuje zmniejszenie liczby wypadków, czasu podróży oraz poziomu emisji spalin i hałasu.

4o

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.

Rozwój uczenia maszynowego został zainicjowany przez Arthura Samuela w 1959 roku. Samuel wprowadził termin "machine learning" w kontekście programowania komputerów zdolnych do uczenia się na podstawie danych. Jest on również autorem jednego z pierwszych samodzielnie uczących się systemów, programu grającego w warcaby.

Uczenie maszynowe dzieli się na trzy główne typy; uczenie nadzorowane, uczenie bez nadzoru oraz uczenie ze wzmocnieniem[[12]](#footnote-12). 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.

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

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

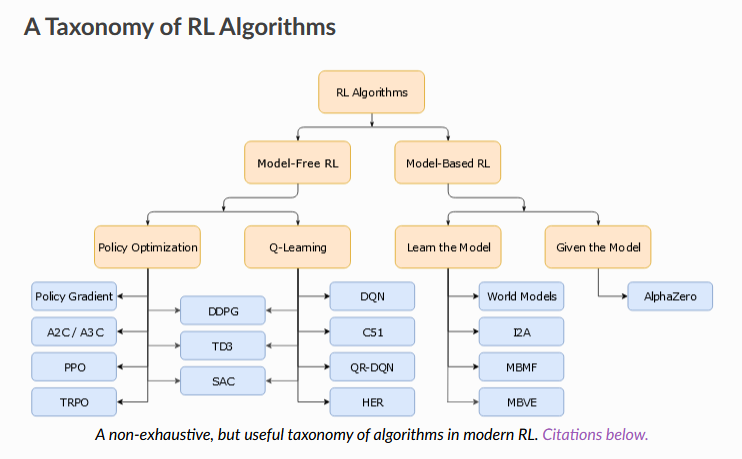
Początki uczenia przez wzmacnianie sięgają lat 50. XX wieku. Są silnie zakorzenione w badaniach nad zachowaniem adaptacyjnym, dynamicznym programowaniem i Procesami Decyzyjnymi Markowa. Istnieje wiele obszarów, które są związane z uczeniem przez wzmacnianie. Najistotniejsze przedstawione są na rysunku 1.3

*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) posiada dużo analogii do modeli psychologicznych z dziedziny warunkowania klasycznego. Eksperymenty Ivana Pawłowa z psami wykazały, że zwierzęta potrafią kojarzyć sygnały środowiskowe, takie jak dźwięk dzwonka, z nagrodami, np. jedzeniem. Pawłow nazwał to terminem „wzmocnienie” które odnosi się do nagrody, która wzmacniała pożądane działania psa (agenta). [[13]](#footnote-13)

Współcześnie algorytmy RL to wariacie na temat sieci deep-Q (deep Q-network, DQN) oraz deep deterministic policy gradient (DDPG). Oba algorytmy są łatwe do wdrożenia, a jednocześnie oferują bardzo duże możliwości adaptacjo do środowiska. [[14]](#footnote-14)

Na rysunku poniżej znajduje się taksometria współczesnych algorytmów RL, korzystających z różnych technik predykcji podejmowanych akcji.

(rysunek)[[15]](#footnote-15)

1. 1 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

Żródło: Schemat pochodzi z ksiązki „Reinforcement Learning: An Introduction” Second edition, in progress

November 5, 2017, stron 38 [[16]](#footnote-16)

**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.

1.2 **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 akcje i nagrody.

MDP można zdefiniować jako 5-eleentową krotkę:

*MDP = (S,A,P,R,γ) (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:

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

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

(2)

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

**1.4 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.

(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.

**1.5 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:  
[[17]](#footnote-17) (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.

**1.6 Polityka.**

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

(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).

1.7 Równania Bellmana

Równania Bellmana są wykorzystywane do **rekurencyjnego wyznaczania wartości stanu** (V(s)) lub **optymalnej polityki** (π∗(s)) w danym stanie. 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)):**

**Wyjaśnienie składników:**

1. **Vπ(s)V^\pi(s)Vπ(s):**
   * Wartość stanu sss przy danej polityce π\piπ.
   * Oznacza oczekiwaną sumę zdyskontowanych nagród, zaczynając od stanu sss i postępując zgodnie z polityką π\piπ.
2. **Ea∼π,s′∼P\mathbb{E}\_{a \sim \pi, s' \sim P}Ea∼π,s′∼P​:**
   * Oczekiwanie (średnia wartość) po losowych zmiennych:
     + a∼πa \sim \pia∼π: Akcja aaa jest wybierana zgodnie z polityką π(a∣s)\pi(a|s)π(a∣s), czyli prawdopodobieństwem wybrania akcji aaa w stanie sss.
     + s′∼Ps' \sim Ps′∼P: Nowy stan s′s's′ jest losowany z rozkładu P(s′∣s,a)P(s' | s, a)P(s′∣s,a), który opisuje przejścia między stanami w środowisku.
3. **r(s,a)r(s, a)r(s,a):**
   * Nagroda natychmiastowa za wykonanie akcji a w stanie sss.
4. **γ\gammaγ:**
   * Współczynnik dyskontowania (0≤γ≤10 \leq \gamma \leq 10≤γ≤1).
   * Określa, jak bardzo przyszłe nagrody są uwzględniane w obecnych decyzjach. Im mniejsze γ, tym mniej ważne są przyszłe nagrody.
5. **Vπ(s′)V^\pi(s')Vπ(s′):**
   * Wartość stanu s′, do którego przechodzi system po wykonaniu akcji aaa.

Określa wartość stanu s, gdy agent działa zgodnie z polityką π.

Nagroda (r(s,a)) za wykonanie akcji a w stanie s jest sumowana z wartością zdyskontowaną przyszłych stanów (γVπ(s′)).

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

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 aaa 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)\pi^\*(s)π∗(s).

****Iteracja polityki:****

Naprzemienne kroki oceny polityki (Vπ(s)V^\pi(s)Vπ(s)) i jej ulepszania (π′(s)\pi'(s)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

1.9 Algorytm Aktor-Krytyk (Actor-Critic)



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

**Algorytm actor–critic** jest połączeniem algorytmów aproksymacji zarówno funkcji polityki (policy function), jak i funkcji wartości (value function). W algorytmach opartych na polityce, funkcja polityki jest aktualizowana na końcu epizodu, co bywa mało efektywne. Wysoka wariancja gradientu (rezultat sumowania wszystkich zdarzeń z epizodu) powoduje, że potrzeba więcej próbek (epizodów) celem stabilizacji modelu.  
Algorytm actor–critic 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 , 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 dwóch elementów: (rysunek przedstawia dynamikę procesu)

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

**Critic** - 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.

,

gdzie:

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

rt​ to nagroda natychmiastowa,

V(s) to funkcja wartości stanu,

γ to współczynnik dyskontowania.

Metody aktor-krytyk łączą więc zalety:

* Metod opartych na wartościach (redukcja wariancji dzięki krytykowi),
* Metod opartych na politykach (elastyczność w modelowaniu przestrzeni ciągłych).



Rysunek: asdas[[18]](#footnote-18)

2. Dlaczego Actor-Critic?

00:02 - 00:42  
Algorytm REINFORCE jest dobrym wprowadzeniem do metod gradientu polityki, ale ma kilka ograniczeń. REINFORCE charakteryzuje się wysoką wariancją. W najprostszym przypadku używa jednej trajektorii do obliczenia gradientu polityki, co sprawia, że proces uczenia jest bardzo niestabilny. Jako metoda Monte Carlo, REINFORCE uczy się dopiero na końcu każdego epizodu. Z kolei metody różnicy czasowej (Temporal Difference) uczą się w każdym kroku, co pozwala na bardziej efektywne uczenie. Metody Actor-Critic rozwiązują te problemy dzięki sieci Critic, która wprowadza uczenie oparte na różnicy czasowej. Omówimy algorytm Advantage Actor-Critic (A2C), który wykorzystuje błąd TD jako funkcję przewagi (advantage).

3. Intuicja stojąca za metodami Actor-Critic

00:42 - 01:25  
Wyobraź sobie studentkę przygotowującą się do egzaminu. Może uczyć się sama i otrzymać informację zwrotną dopiero po uzyskaniu wyników egzaminu albo dołączyć do grupy naukowej, gdzie regularnie testuje swoją wiedzę z pomocą innych. Studentka jest jak sieć aktora – decyduje, czego się uczyć i odpowiada na pytania testowe, ale sama nie potrafi dobrze ocenić swoich postępów. Grupa naukowa jest jak sieć krytyka – dostarcza regularną informację zwrotną, aby pomóc studentce uczyć się lepiej i szybciej. Formalnie rzecz biorąc, sieć krytyka to sieć wartości. Jej zadaniem jest ocena funkcji wartości w każdym kroku, aby ocenić jakość ostatniej akcji i dostarczyć tę informację zwrotną aktorowi.

4. Sieć Critic

01:25 - 01:57  
Podobnie jak sieci Q, które poznaliśmy w Deep Q-Learning, sieć Critic jest aproksymatorem funkcji wartości. W tym przypadku aproksymuje funkcję wartości stanu VVV, a nie funkcję wartości akcji QQQ. Zadaniem Critic jest ocena ostatniej akcji ata\_tat​ aktora za pomocą funkcji przewagi (advantage) lub błędu TD. Architektura sieci Critic jest podobna do sieci Q, ale ma pojedynczy węzeł wyjściowy, który reprezentuje wartość stanu.

5. Dynamika Actor-Critic

01:57 - 02:10  
Przyjrzyjmy się dynamice algorytmów Actor-Critic. Sieć aktora przypomina sieć polityki w algorytmie REINFORCE. Reprezentuje stochastyczną politykę agenta i wybiera akcję w każdym kroku.

6. Dynamika Actor-Critic

Sieć Critic obserwuje nagrodę i zaktualizowany stan pochodzący ze środowiska.

Critic oblicza błąd TD. Jest to kluczowe dla obliczania funkcji straty.

Zaktualizowana sieć Actor również obserwuje nowy stan.

Proces zaczyna się od nowa, gdy aktor wybiera swoją akcję na następny krok.

1.10 Deep Learning w kontekście RL

**Uczenie głębokie (Deep Learning, DL)** to dziedzina sztucznej inteligencji, która korzysta z sieci neuronowych o wielu warstwach, pozwalających na efektywne przetwarzanie i reprezentowanie złożonych danych. W uczeniu przez wzmacnianie (Reinforcement Learning, RL), metody DL odgrywają kluczową rolę w rozwiązywaniu problemów związanych z dużymi i złożonymi przestrzeniami stanów oraz akcji. Dzięki DL możliwe jest wykorzystanie modeli zdolnych do aproksymacji funkcji i optymalizacji polityki agenta.

**Aproksymacja funkcji wartości i polityki:** W klasycznym RL metody, takie jak iteracja wartości lub Q-learning, wymagają obliczeń na tablicach (tabelaryczny RL). W praktyce jest to ograniczone do małych przestrzeni stanów i akcji. DL pozwala na użycie sieci neuronowych jako funkcji aproksymujących, np. wartości stanów V(s), wartości Q-funkcji Q(s,a) lub bezpośrednio polityki π(a∣s).

**Zastępowanie rekurencyjnych obliczeń:** Tradycyjne algorytmy RL, jak Q-learning, polegają na wielokrotnym wykonywaniu równań Bellmana w celu propagacji nagród w czasie. Dzięki wykorzystaniu sieci neuronowych, takie obliczenia mogą zostać „nauczone”, co redukuje koszt obliczeniowy poprzez jednorazowe trenowanie modelu.

Współdziałanie RL i DL: Główne korzyści

**Deep Q-Learning (DQN):** Pierwszym przełomem było wprowadzenie przez Mnih i wsp. (2015) metody Deep Q-Network, w której głęboka sieć neuronowa aproksymuje wartość Q-funkcji. DQN umożliwiło rozwiązanie problemów takich jak gry Atari, gdzie przestrzeń stanów (obrazy z gry) jest bardzo duża.

Sieć neuronowa uczy się, przypisując wartości Q(s,a), co eliminuje potrzebę stosowania tabelarycznej reprezentacji dla każdej pary stan-akcja.

Zamiast iteracyjnego stosowania równań Bellmana, model uczy się przewidywania wartości zdyskontowanych nagród.

**Polityki stochastyczne i deterministyczne:** W algorytmach opartych na politykach (np. REINFORCE, PPO, Actor-Critic) sieci neuronowe są używane do modelowania samej polityki π(a∣s)\pi(a|s)π(a∣s), co pozwala na eksplorację bardziej złożonych środowisk i lepsze strategie decyzyjne.

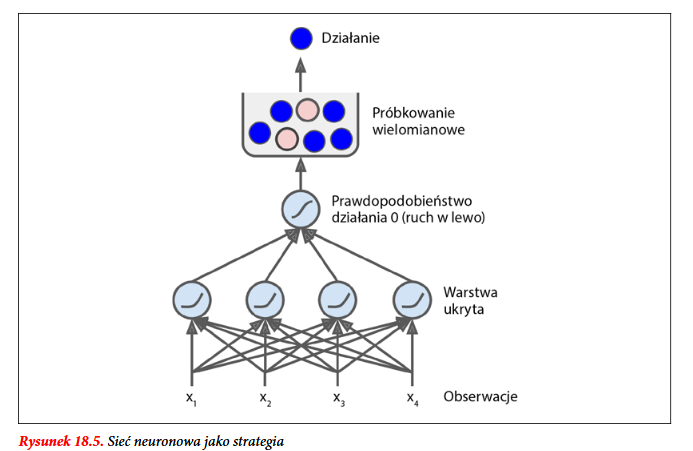
**Efektywność obliczeniowa:**

Sieci neuronowe są w stanie uogólniać wiedzę na niespotkane wcześniej stany dzięki wykorzystaniu gradientowego uczenia.

Współczesne akceleratory sprzętowe (np. GPU) umożliwiają szybkie uczenie bardzo złożonych modeli.

Podsumowanie: RL wspierane przez Deep Learning

**Uczenie głębokie** umożliwia zastosowanie algorytmów RL w rzeczywistych problemach, takich jak sterowanie robotami, planowanie w środowiskach symulowanych czy podejmowanie decyzji w grach. Kluczowym elementem jest zdolność sieci neuronowych do przechwytywania i uczenia się wzorców, które wcześniej wymagały mozolnych obliczeń rekurencyjnych lub ręcznego modelowania. Tym samym Deep Learning zrewolucjonizował RL, czyniąc go narzędziem praktycznym i skalowalnym do rzeczywistych zastosowań.



1. Pakiet SUMO:

SUMO jest zestawem aplikacji wspierających symulację scenariuszy ruchu drogowego. Obejmuje narzędzia do generowania sieci drogowej, tworzenia popytu transportowego oraz przeprowadzania symulacji​

Narzędzia takie jak „netgenerate” i „netconvert” umożliwiają generowanie lub importowanie sieci drogowych z różnych formatów. Dodatkowo SUMO obsługuje tworzenie scenariuszy o dużej skali, takich jak symulacje ruchu w całych miastach​

Zastosowania SUMO:

SUMO jest używany w badaniach nad komunikacją V2X (pojazd-pojazd i pojazd-infrastruktura), wspierając generowanie realistycznych ścieżek pojazdów na potrzeby symulacji komunikacyjnych​

Narzędzie pozwala na badanie algorytmów wyboru tras, dynamicznej nawigacji i optymalizacji sygnalizacji świetlnej​

​ .

Rozwój i nowe funkcje[[19]](#footnote-19):

SUMO zostało rozszerzone o modele emisji hałasu oraz zanieczyszczeń powietrza, co pozwala ocenić ekologiczne skutki różnych scenariuszy ruchu​

Dodano funkcjonalności umożliwiające symulację ruchu intermodalnego oraz wsparcie dla pojazdów autonomicznych, co znajduje zastosowanie w projektach takich jak CityMobil​

.

Plany na przyszłość:

Planowane są dalsze ulepszenia modeli ruchu, integracja z innymi symulatorami oraz rozwój interfejsu TraCI, który umożliwia sterowanie symulacją w czasie rzeczywistym​

Artykuł zachęca również społeczność do korzystania z SUMO w swoich badaniach i rozwoju nowych algorytmów. Platforma zyskała popularność dzięki swojej wszechstronności, otwartemu kodowi źródłowemu oraz wsparciu dla dużych symulacji.

***Interfejs do SUMO***

* *Najczęściej wykorzystuje się TraCI (Traffic Control Interface) do komunikacji z SUMO w czasie rzeczywistym.*
* *Z poziomu TraCI można odczytywać statystyki (np. prędkości pojazdów, czasy zatrzymania, długość kolejki) oraz ustawiać stan sygnalizacji świetlnych.*

***Implementacja AC***

*Metody aktor-krytyk to technika uczenia ze wzmocnieniem, która łączy dwa elementy:*

* *Aktor: odpowiada za podejmowanie decyzji, czyli wybór akcji (polityka).*
* *Krytyk: ocenia jakość tych decyzji, przewidując wartość stanów lub akcji.*

*Jak działa metoda aktor-krytyk?[[20]](#footnote-20)*

1. *Co robi aktor?*
   * *Aktor wybiera akcje na podstawie polityki (strategie), którą próbuje się zoptymalizować.*
2. *Co robi krytyk?*
   * *Krytyk oblicza funkcję wartości, która ocenia, jak dobre są stany lub akcje. To pozwala na szybsze uczenie, bo krytyk dostarcza wskazówek aktorowi.*
3. *Bootstrapping:*
   * *Krytyk wykorzystuje częściowe informacje o przyszłych stanach (czyli przewiduje wartości kolejnych kroków), aby przyspieszyć uczenie.*
   * *To wprowadza pewne "obciążenie" (niewielki błąd w obliczeniach), ale zmniejsza wariancję wyników, co czyni algorytm bardziej efektywnym.*

*Dlaczego warto korzystać z aktor-krytyk?*

* *Szybsze uczenie: Dzięki krytykowi algorytm lepiej wykorzystuje informacje z każdej interakcji ze środowiskiem.*
* *Mniej chaotyczne wyniki: Metoda redukuje przypadkowość (wariancję), co daje stabilniejsze efekty.*
* *Działa w czasie rzeczywistym: Nadaje się do problemów, które wymagają ciągłego uczenia i szybkiego podejmowania decyzji.*

*ALGORYTM: ACTOR-CRITIC Z DYSKRETNYMI AKCJAMI {2,3,5,7,10,15}*

*INICJALIZACJA:*

*1. Wylosuj/ustaw parametry sieci aktora θ*

*2. Wylosuj/ustaw parametry sieci krytyka φ*

*3. Zdefiniuj zbiór akcji: actions = [2, 3, 5, 7, 10, 15]*

*4. Ustal współczynniki: α\_θ, α\_φ, γ*

*5. Ustal liczbę epizodów: MAX\_EPISODES*

*DLA epizod = 1 DO MAX\_EPISODES:*

*# 1. RESET środowiska i pobierz stan początkowy*

*s\_t = Environment.reset()*

*done = false*

*DOPÓKI done == false:*

*# 2. Aktor -> wyznacz rozkład P(a|s\_t)*

*prob = policy(s\_t; θ) # prob jest np. wektorem 6 liczb (softmax)*

*# 3. Wylosuj indeks akcji z prob*

*a\_idx = sample\_multinomial(prob) # a\_idx ∈ {0,1,2,3,4,5}*

*a\_t = actions[a\_idx] # a\_t ∈ {2,3,5,7,10,15}*

*# 4. Wykonaj akcję w środowisku*

*(s\_t\_plus\_1, r\_t\_plus\_1, done) = Environment.step(a\_t)*

*# 5. Krytyk -> oceń stany*

*V\_st = V(s\_t; φ)*

*V\_st\_plus\_1 = 0*

*JEŻELI done == false:*

*V\_st\_plus\_1 = V(s\_t\_plus\_1; φ)*

*# 6. Oblicz TD-error: δ\_t = r + γ V(s\_{t+1}) - V(s\_t)*

*delta\_t = r\_t\_plus\_1 + γ \* V\_st\_plus\_1 - V\_st*

*# 7. AKTUALIZACJA AKTORA:*

*# L\_actor = - δ\_t \* log π\_θ(a\_t | s\_t)*

*# θ ← θ - α\_θ \* ∂L\_actor/∂θ*

*# Pseudokodowo (rozpisane kroki gradientu pomijamy -*

*# w praktyce wywołujesz "backprop" w bibliotece NN).*

*theta\_grad = - delta\_t \* grad( log π\_θ(a\_t | s\_t) )*

*θ = θ - α\_θ \* theta\_grad*

*# 8. AKTUALIZACJA KRYTYKA:*

*# L\_critic = ( V(s\_t) - [ r\_t\_plus\_1 + γ V(s\_{t+1}) ] )^2*

*# φ ← φ - α\_φ \* ∂L\_critic/∂φ*

*target = r\_t\_plus\_1 + γ \* V\_st\_plus\_1*

*critic\_grad = grad( (V\_st - target)^2 w.r.t φ )*

*φ = φ - α\_φ \* critic\_grad*

*# 9. Przejdź do kolejnego kroku*

*s\_t = s\_t\_plus\_1*

*KONIEC\_DOPÓKI*

*KONIEC\_DLA*

1. Generalna Dyrekcja Dróg Krajowych i Autostrad, https://drogi.gddkia.gov.pl/serwisy-gddkia/krajowy-system-zarzadzania-ruchem/zalozenia-kszr/co-to-jest-its [↑](#footnote-ref-1)
2. National-geographic - https://www.national-geographic.pl/nauka/nagroda-nobla-2024/ [↑](#footnote-ref-2)
3. Obserwator finansowy https://www.obserwatorfinansowy.pl/tematyka/makroekonomia/trendy-gospodarcze/fenomen-chatgpt-i-jego-skutki/ [↑](#footnote-ref-3)
4. Wikipedia https://en.wikipedia.org/wiki/Tensor\_Processing\_Unit [↑](#footnote-ref-4)
5. Google https://deepmind.google/discover/blog/alphago-zero-starting-from-scratch/ [↑](#footnote-ref-5)
6. By Kara Nelson, CNN - https://edition.cnn.com/2023/11/24/us/garrett-morgan-traffic-signal-100-years-reaj/index.html [↑](#footnote-ref-6)
7. Gaca S., Suchorzewski W., Tracz M., Inżynieria ruchu drogowego. Teoria i praktyka, W.K.Ł., Warszawa, 2008 [↑](#footnote-ref-7)
8. 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-8)
9. SCOOT® Version History, Split Cycle and Offset Optimisation Technique, https://trlsoftware.com/software/intelligent-signal-control/scoot/scoot-version-history/ [↑](#footnote-ref-9)
10. Politechnika Opolska Wydział Elektrotechniki, Automatyki i Informatyki Instytut Automatyki i Informatyki, Algorytmy sterowania acykliczną sygnalizacją świetlną w zatłoczonej sieci drogowej [↑](#footnote-ref-10)
11. MiĞkiewicz M.: ViaPIACON – polska metoda sterowania ruchem drogowym. Przegląd ITS nr 4, Warszawa 2008. [↑](#footnote-ref-11)
12. Sztuczna Inteligencja od Podstaw, sztuczna-inteligencja-od-podstaw-feliks-kurp-helion-2.pdf [↑](#footnote-ref-12)
13. Steven L. Brunton, J. Nathan Kutz, Data Driven Science & Engineering Machine Learning, Dynamical Systems, and Control (databookRL.pdf) [↑](#footnote-ref-13)
14. Google CLOUD, https://www.cloudskillsboost.google/focuses/10285?locale=pl&parent=catalog [↑](#footnote-ref-14)
15. Open AI Spinning UP https://spinningup.openai.com/en/latest/spinningup/rl\_intro2.html [↑](#footnote-ref-15)
16. „Reinforcement Learning: An Introduction” - Second edition, in progress

    ”Complete Draft” November 5, 2017 Richard S. Sutton and Andrew G. Barto http://incompleteideas.net/book/bookdraft2017nov5.pdf [↑](#footnote-ref-16)
17. I Reinforcement Learning: An Introduction (wzór 3.8) [↑](#footnote-ref-17)
18. An Analysis of Actor/Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imp erfect Value Functions: Ha jime Kimura# Shigenobu Kobayashi <http://users.umiacs.umd.edu/~hal/courses/2016F_RL/Kimura98.pdf> [↑](#footnote-ref-18)
19. The International Journal on Advances in Systems and Measurements is published by IARIA [↑](#footnote-ref-19)
20. <http://incompleteideas.net/book/bookdraft2017nov5.pdf> s.254 [↑](#footnote-ref-20)