VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Sustiprinto mašininio mokymo metodų taikymas kompiuteriniuose galvosūkių žaidimuose

Application of reinforced machine learning methods in logic-based computer games

Magistro baigiamasis darbas

Atliko: Jonas Mažeika

Darbo vadovas: dr. Andrius Vytautas Misiukas Misiūnas

Recenzentas: doc. Valentas Gružauskas

Vilnius – 2025

Santrauka

Šiame darbe yra tiriama ir vertinama įvairių sustiprinto mokymosi metodų, skirtų agento mokymui imituotoje aplinkoje, veikimas. Pagrindinis dėmesys buvo skiriamas Proksimalinės politikos optimizavimo (PPO) ir Švelnaus aktoriaus-kritiko (SAC) metodams, pritaikant ir lyginant tokias patobulintas strategijas, kaip PPO su prisitaikančia KL bauda, PPO su pasitikėjimo regiono optimizavimu, SAC su fiksuota temperatūra ir SAC su automatiniu temperatūros reguliavimu. Šiame tyrime nagrinėjama problema yra susijusi su politikos mokymosi optimizavimu dinamiškoje aplinkoje, užtikrinant stabilią konvergenciją ir maksimaliai padidinant kumuliacinį atlygį, taip pat – tinkamiausio metodo parinkimas, priklausomai nuo scenarijaus.

Eksperimentai buvo atlikti dviejose skirtingose aplinkose, kur kiekvienas metodas buvo išbandytas ir analizuojamas lygiagrečiai remiantis tokiais metrikų rodikliais, kaip atlygis už epizodą, politikos praradimas, vertės praradimas ir kitomis metrikomis. Rezultatai parodė, kad SAC su automatiniu temperatūros reguliavimu veikė geriausiai pirmoje (lengvesnėje) aplinkoje dėl prisitaikančio entropijos tvarkymo, kuris palengvino veiksmingą tyrinėjimą ir naudojimą. Priešingai, PPO su patikimumo regiono optimizavimu puikiai pasirodė antroje (sudėtingesnėje) aplinkoje, pademonstruodamas didesnį stabilumą ir atlygio kaupimą. Pagrindiniai hiperparametrai, įskaitant gama (nuolaidos koeficientą), mokymosi greitį, entropijos koeficientus, KL baudas ir partijos dydžius, buvo sistemingai derinami, siekiant optimalaus našumo.

Šio tyrimo išvados rodo, kad SAC pagrįsti metodai suteikia geresnį pritaikomumą, ypač aplinkoje, kuriai reikalingas dinaminis tyrinėjimas, o PPO pagrįsti metodai užtikrina stabilumą kontroliuojamuose nustatymuose su apribotais politikos atnaujinimais. Šie tyrimai prisideda prie nuolatinio sustiprinimo mokymosi tobulinimo žaidimuose, kurie gali būti pritaikyti ir realaus pasaulio sprendimų priėmimo užduotims atlikti.

Raktiniai žodžiai: Sustiprintas mokymasis, PPO, SAC, politikos optimizavimas, KL bauda, agentų mokymas.

Summary

This work explores and evaluates the performance of different reinforcement learning methods for training an agent in a simulated environment. The primary focus was on Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) algorithms, comparing variations such as PPO with Adaptive KL Penalty, PPO with Trust Region Optimization, SAC with Fixed Temperature, and SAC with Automatic Temperature Adjustment. The problem addressed in this study involves optimizing policy learning in dynamic environments, ensuring stable convergence and maximizing cumulative rewards as well as choosing the right method according to the scenario.

The experiments were conducted in two different environments, where each method was tested and analyzed based on metrics such as reward per episode, policy loss, value loss, and KL divergence. Results demonstrated that SAC with Automatic Temperature Adjustment performed best in the first environment due to its adaptability in regulating exploration and exploitation. In contrast, PPO with Trust Region Optimization excelled in the second environment, showcasing improved stability and reward accumulation. Key hyperparameters, including gamma (discount factor), learning rate, entropy coefficients, KL penalties, and batch sizes, were systematically tuned to achieve optimal performance.

The findings of this study suggest that SAC-based methods offer better adaptability, especially in environments requiring dynamic exploration, while PPO-based approaches ensure stability in controlled settings with constrained policy updates. These insights contribute to the ongoing improvements in deep reinforcement learning applications for real-world decision-making tasks.

Keywords: Reinforcement learning, PPO, SAC, Policy optimization, KL penalty, Agent training.

Turinys

ĮVADAS 6

1. LITERATŪROS ANALIZĖ 8
   1.1. Unity žaidimų variklis 8
   1.2. Skirtingų metodų testavimo įrankis 9
   1.3. Skirtingi sustiprinti mokymosi metodai 11
      1.3.1. Proximal policy optimisation. Nr. 1 11
      1.3.2. Proximal policy optimisation. Nr. 2 13
      1.3.3. Proximal policy optimisation. Nr. 3 16
      1.3.4. Proximal policy optimisation. Nr. 4 19
      1.3.5. Proximal policy optimisation. Nr. 5 21
      1.3.6. Generative Adversarial Imitation Learning 23
      1.3.7. Actor-Critic Nr. 1 25
      1.3.8. Actor-Critic Nr. 2 27
      1.3.9. Actor-Critic Nr. 3 28
      1.3.10. Actor-Critic Nr. 4 30
   1.4. Pagalbinis metodas 31
      1.4.1. Behavioural Cloning 31
   1.5. Aktyvacijos funkcija 33
      1.5.1. RMAF: ReluMemristor Like Activation 33
   1.6. Išanalizuotos literatūros apibendrinimas 34

2. TECHNOLOGIJOS IR TESTAVIMO APLINKA 36
   2.1. Unity Game Engine 36
   2.2. Unity ML-Agents Toolkit 36
   2.3. Python API 37
   2.4. TensorFlowSharp integracija 37
   2.5. Mokymo ir testavimo aplinka 37

3. METODIKA 39
   3.1. Eksperimentinė sąranka 39
   3.2. Metodų patobulinimai 40
      3.2.1. PPO su adaptyviąja KL bauda 40
      3.2.2. PPO su pasitikėjimo regiono optimizavimu 41
      3.2.3. SAC su fiksuota temperatūra 43
      3.2.4. SAC su automatiniu temperatūros reguliavimu 44
   3.3. Palyginimas tarp keturių metodų 45

4. TYRIMO REZULTATAI 47
   4.1. Parametrų radimas metodams 47
      4.1.1. PPO su adaptyviąja KL bauda parametrai 47
      4.1.2. PPO su pasitikėjimo regiono optimizavimu parametrai 48
      4.1.3. SAC su fiksuota temperatūra parametrai 48
      4.1.4. SAC su automatiniu temperatūros reguliavimu 49
   4.2. Metodų palyginimas aplinkose 50
      4.2.1. Mokymosi procesas pirmojoje aplinkoje 50
      4.2.2. Mokymosi procesas antrojoje aplinkoje 51

4.3. Rezultatų validavimas 52
4.3.1. Rezultatų validavimas pirmojoje aplinkoje 52
4.3.2. Rezultatų validavimas antrojoje aplinkoje 54
5. DISKUSIJA 57
REZULTATAI 58
IŠVADOS 59
ŠALTINIAI 60
PRIEDAI 63

Įvadas

Šiuolaikinėje technologinėje aplinkoje mašininis mokymasis (ML) tapo esminiu įvairių pramonės šakų ir taikomųjų programų įrankiu, leidžiančiu iš didelio duomenų kiekio gauti reikšmingų įžvalgų, modelių ir prognozių. Vienas reikšmingų dirbtinio intelekto (AI) pasiekimų yra didėjantis modeliuojamų aplinkų, naudojamų mašininio mokymosi modeliams lavinti, tikroviškumas ir sudėtingumas [JBT+20]. Tačiau daugelis esamų platformų duoda nerealius rezultatus, kenčia nuo netikslios fizikos, mažo užduočių sudėtingumo ir ribotų agentų sąveikos galimybių [Ma+19]. Be to, daugeliui platformų (pvz. Unity Engine) trūksta lankstumo konfigūruojant modeliavimo aplinką norint efektyviai mokytis (iš sistemos perspektyvos). Remiantis [Res23] portalu, galima matyti, kad žaidimų rinka yra labai didelė (2022 metais jų buvo parduota už 29,16 milijardų dolerių) ir nemaža jų dalis naudojo mašininį mokymąsi. Problema yra ta, kad nors ir yra nemažai sukurta įvairių sustiprinto mokymosi metodų, jie dažnai nėra palyginami su kitais arba tai nėra įvertinama akademiškai (pvz. [AIc19] puslapyje galima rasti Unity sukurtos aplinkos iššūkį, kuriame dalyvavo programuotojai, tačiau nėra paminėta nei kokius metodus jie naudojo nei mokymosi eigos). Šiuo darbu struktūriškai ir lygiagrečiai yra vertinami keturi metodai, nurodant jų parametrus, mokymosi eigą ir jų rezultatus.

Darbo tikslas:

Šio darbo tikslas yra ištirti, įvertinti ir palyginti patobulintus PPO ir SAC sustiprinto mašininio mokymosi metodus skirtingo sudėtingumo scenarijuose, randant tuos metodus, kurie demonstruotų geresnį veikimą pagal vieną ar kelias metrikas, ir būtų pritaikomi kompiuterinių žaidimų kūrime.

Darbo tikslui pasiekti yra keliami šie uždaviniai:

1. Atlikti sustiprinto mašininio mokymosi metodų literatūros apžvalgą, taip pasirenkant metodus tolesniems tyrimams.
2. Skirtingo sudėtingumo aplinkų ir scenarijų kūrimas ML agentams, vertinimas (su TensorFlow įrankiu).
3. Gautų rezultatų analizė ir vertinimas (metodų metrikų radimas, pagal kurias, metodai galėtų būti vertinami, taip pat eksperimento sukūrimas, kuris šiuos metodai lygiaverčiai ištestuotų).
4. Pagal gautus tyrimų rezultatus geriausių sprendimų radimas, nustatant, kuris metodas būtų labiausiai tinkamas tam tikro sudėtingumo aplinkoje.

Aktualumas ir sprendžiama problema:

Žaidimų pramonė vis labiau pasikliauna sudėtingu DI, gerinant žaidėjų patirtį žaidimuose. Tačiau sukurti „protingus“ žaidimų agentus, gebančius mokytis, pritaikyti ir optimizuoti strategijas sudėtingoje aplinkoje, yra nemažas iššūkis. Šiame darbe yra giliau gilinamasi į šiuos

iššūkius, toliau nagrinėjant ir modifikuojant proksimalinės politikos optimizavimo (PPO) ir švelnaus veikėjo kritiko (SAC) metodus.

Vienas iš šių problemų sprendimo būdų yra sustiprinto mokymosi (SM) pasitelkimas. Pastaraisiais metais SM išpopuliarėjo dėl savo gebėjimo gerai veikiant dirbtinai sukurtoje aplinkoje, ypač naudojant tokius žaidimų variklius kaip Unity [JKB+19]. Šis žaidimo variklis yra naudojamas šiame darbe kuriant sudėtingus į galvosūkius panašius scenarijus, kuriuos mašininio mokymosi agentai turės išspręsti priimdami sprendimus laikui bėgant. Pagrindiniai SM metodai, kurie buvo tiriami „Magistrinio darbo literatūros apžvalgos“ darbe , yra proksimalinės politikos optimizavimas (PPO), švelnus veikėjas-kritikas (SAC), generatyvus priešiškas mokymasis (GAIL) ir elgesio klonavimas (BC) [JBV+18]. Šie metodai nebuvo iki galo ištirti, o jų veikimas greičio ir efektyvumo požiūriu tam tikruose scenarijuose, pavyzdžiui, vienodom sąlygom sukurtose aplinkose, lieka neįvertintas. Išanalizavus šiuos SM metodus ir suradus du geriausius (žr. „1.6 Išanalizuotos literatūros apibendrinimas“ poskyrį atrinktiems metodams), siekiama pasiūlyti šių metodų patobulintas versijas, kurie parodytų geresnį vienos ar kelių metrikų našumą ir galėtų būti pritaikyti kuriant kompiuterinius žaidimus.

1. Literatūros analizė

Šiame skyriuje yra apžvelgiami skirtingi literatūros šaltiniai, į kuriuos įeina aplinkos ir scenarijų kūrimo įrankis - Unity žaidimų variklis, skirtingų metodų testavimo įrankis ir pačių metodų nagrinėjimas bei jų surasti trūkumai ir privalumai.

1.1. Unity žaidimų variklis

Straipsnio [JBT+20] santraukoje pabrėžiami naujausi dirbtinio intelekto (AI) pasiekimai, kuriuos lemia tikroviškos imituojamos aplinkos, ir nurodomi esamų platformų apribojimai. Šie apribojimai apima netikrovišką vaizdą, netikslią fiziką, mažą užduočių sudėtingumą, ribotą agento perspektyvą ir ribotą konfigūraciją. Siekiant išspręsti šias problemas, darbe siūloma nauja modeliavimo platformų taksonomija, daugiausia dėmesio skiriant bendroms platformoms, palaikančioms platų vizualinį, fizinį, užduočių sudėtingumą. Darbe labiausiai yra koncentruojamasi į Unity žaidimo variklį ir Unity ML-Agents Toolkit. Įvade pabrėžiamas svarbus modeliavimo platformų vaidmuo gilaus mokymosi (DRL) tyrimuose. Minimos platformas kaip „Arcade Learning Environment“ (ALE), „VizDoom“, „MuJoCo“ ir kitos, skirtos lyginamajai analizei ir algoritmams tobulinti. Straipsnyje gilinamasi į aplinkos ir simuliatorių charakteristikas, pristatoma jutiminė, fizinė, užduočių logika ir socialinis sudėtingumas pagrindžiant taksonomija. Eksperimentiniams tyrimams aptariamos esminės modeliavimo savybės, tokios kaip greitas ir paskirstytas modeliavimas bei lankstus valdymas.

Simuliatorių apklausos rezultatai buvo suskirstyti į keturis tipus: viena aplinka, aplinkos rinkinys, konkrečios srities platforma ir bendroji platforma. Analizuojami pavyzdžiai, tokie kaip ALE, DeepMind Lab, Project Malmo, MuJoCo ir VizDoom, pabrėžiant jų stipriąsias ir ribotas puses.

Unity ML-Agents Toolkit aprašoma kaip galinga platforma intelektualiems agentams kurti ir mokyti. Šis įrankis pasižymi didesniu modeliavimo greičiu nei realiuoju. Taip pat leidžia nepriklausomai vykdyti žaidimo logiką, o tai naudinga aplinkai be vizualinių stebėjimų. Našumo metrika, aprašyta 2 lentelėje, parodo įrankių rinkinio galimybes valdant aplinkas naudojant Python API. Pavyzdinės aplinkos, išsamiai aprašytos 3 paveiksle, apima įvairius scenarijus, stebėjimus, veiksmus ir apdovanojimus. Pradiniai rezultatai naudojant PPO ir SAC algoritmus pateikti 3 lentelėje. 6 skyriuje pabrėžiami tyrimai, atlikti naudojant Unity ir Unity ML-Agents Toolkit, pabrėžiant jo lankstumą. Kliūčių bokšto etalonas yra demonstruojamas kaip sudėtingumo, pasiekiamo Unity aplinkoje, pavyzdys.

Privalumai:

1. Unity siūlo lankstumą kuriant aplinką, leidžiančią kurti sudėtingas vaizdines scenas, sudėtingus fizikos modeliavimus ir interaktyvius scenarijus. Tvirtos platformos funkcijos suteikia tyrėjams ir programuotojams galimybę sukurti įvairius ir sudėtingus DI eksperimentavimo nustatymus (arba juos koreguoti).

2. Unity suteikia intuityvią ir patogią sąsają, supaprastinančią aplinkos projektavimo ir eksperimentavimo procesą. Prieinama sąsaja pagerina bendrą vartotojo patirtį, todėl ji tinka tiek ekspertams, tiek tiems, kurie pradeda dirbti su dirbtinio intelekto tyrimais.
3. Patogus vartotojui platformos dizainas apima bendrą naudojimo paprastumą, sumažinant tyrėjų ir programuotojų mokymosi laiką. Tai leidžia vartotojams daugiau dėmesio skirti AI algoritmų kūrimui ir testavimui.
4. „Unity“ palaiko kelių platformų diegimą, leidžiantį sklandžiai perkelti ir vykdyti sukurtas aplinkas įvairiuose įrenginiuose ir operacinėse sistemose. Ši galimybė pagerina prieinamumą ir palengvina bendradarbiavimą tarp programuotojų, naudojančių skirtingas aparatūros sąrankas.
5. „Unity“ ML-Agents Toolkit įtraukimas praplečia platformos galimybes, nes leidžia diegti mokymosi algoritmus tiesiogiai Unity aplinkose. Programuotojai gali pasinaudoti šiuo įrankiu, norint efektyviai įgyvendinti sustiprinimo mokymosi ir imitavimo mokymosi algoritmus.

Trūkumai:

1. „Unity“ gali turėti vaizdinio ir fizinio tikslumo apribojimų, palyginti su specializuotomis platformomis, sukurtomis išskirtinai tikroviškam modeliavimui. Aplinkos, sukurtos naudojant „Unity“, gali nepasiekti tokio paties vaizdų ar fizikos tikroviškumo lygio, kurios yra labiau specializuotuose varikliuose.
2. „Unity ML-Agents“ įrankių rinkinys, nors ir yra pakankamai galingas, suteikia platformai specifiškumo. Priklausomybė nuo šio įrankių rinkinio gali apriboti „Unity“ pritaikymą tam tikriems mokslinių tyrimų poreikiams, ypač lyginant su bendresnės paskirties varikliais, kurie palaiko platesnį įrankių rinkinių ir sistemų spektrą.
3. „Unity“ našumas gali skirtis atsižvelgiant į modeliuojamos aplinkos sudėtingumą. Aplinkose, kuriose yra sudėtingi vaizdai, sudėtinga fizika ar dideli skaičiavimo reikalavimai - gali kilti našumo problemų, kas gali turėti įtakos Unity mastelio keitimui tam tikrose programose.

1.2. Skirtingų metodų testavimo įrankis

Autorius siūlo integruoti mašininio mokymosi bibliotekas į Unity žaidimų variklį, naudojant TensorflowSharp ir KerasSharp, o tai leistų kūrėjams įdiegti mašininio mokymosi metodus, koncentruojantis į neuroninius tinklus su mokymo galimybe, kurie gali veikti tiek Unity redaktoriuje, tiek atskiruose žaidimuose. Darbe taip pat aptariama galimybė ateityje tobulinti biblioteką, pavyzdžiui, pridėti daugiau neuroninių tinklų architektūrų ir mokymo algoritmų. Autorius pažymi, kad Unity ML-Agents biblioteka turi dvi funkcijas, kurios šiuo metu nepalaikomos šio darbo bibliotekoje: pasikartojančius neuroninius tinklus ir smalsumo modulį. Pasikartojantis neuroninis tinklas yra labai svarbus dirbtiniam intelektui, kuriam reikia atminties, o smalsumo modulis pagerina mokymąsi, jei iš aplinkos gaunami atlygiai yra menki. Taip pat yra pabrėžiami darbo apribojimai, tokie kaip kai kurių populiarių neuroninių

tinklų architektūrų, tokių kaip pasikartojantys neuroniniai tinklai, palaikymo trūkumas ir tai, kad kai kurios funkcijos, tokios kaip neuroninio tinklo mokymas, nepalaikomos kai kuriose platformose, pvz. Android ir iOS. Ateityje autorius planuoja pašalinti šiuos apribojimus, ir daro išvadą, kad biblioteka gali būti naudojama kaip paprastas įrankių rinkinys mašininiam mokymuisi tirti ir mašinų mokymuisi žaidimuose diegti.

Formulės, paminėtos darbe. Šios formulės naudojamos kaip pradiniai taškai kuriant naujas aplinkas arba kaip mokymosi algoritmų etalonas:

1. Formulė nurodo vieną neuroną grįžtamojo ryšio neuroniniame tinkle, kur įvestis x transformuojama svorio vektoriumi w ir poslinkiu b, ir perduodama per aktyvinimo funkciją σ, kad būtų gauta išvestis y: $y = f(x) = \sigma(wx + b)$
2. Ši lygtis parodo visų neuronų operacijas viename grįžtamojo ryšio neuroninio tinklo sluoksnyje. Įvestis x ir poslinkis b vaizduojami kaip vektoriai, o svorio matrica W reiškia ryšius tarp sluoksnio neuronų. Aktyvinimo funkcija σi taikoma kiekvieno neurono svertinai įvesčiai, kad būtų gauta išvestis $f_i(x)$.: $f_i(x) = \sigma_i\left(W_i x + b_i\right)$
3. Stochastic Gradient Descent (SGD) darbe naudojama modelio parametrams atnaujinti treniruočių metu. θ - modelio parametrai, α reiškia mokymosi greitį, o ∇L(θ) - praradimo funkcijos gradientas, atsižvelgiant į parametrus. Praradimo funkcijos gradientas yra θ judėjimo kryptis, kur L didėja greičiausiai. Todėl galima iteratyviai perkelti θ į priešingą šiai krypčiai, kol jis pasiekia tašką (arba pakankamai arti), kuriame gradientas yra nulis, vietinis minimumas. Mokymosi greitis α yra maža konstanta, kuri parodo kiekvienos iteracijos žingsnio dydį. Paprastai kuo mažesnis mokymosi tempas, tuo stabilesni atnaujinimai. Tačiau jei α yra per mažas, vietiniam minimumui pasiekti gali prireikti daug laiko. Formulė: $\theta_{n+1} = \theta_n - \alpha \nabla L(\theta)$
4. Taip pat darbe naudojama politikos optimizavimo (PPO) formulė stiprinant mokymąsi naudojant neuroninius tinklus. Šioje formulėje θ nurodo politikos tinklą, $\hat{A}_t$ - pranašumo įvertinimą, r(θ) - naujosios ir senosios politikos santykį, o ε nurodo hiperparametrą, kuris valdo iškirpimo diapazono dydį. Algoritmas veikia maksimaliai padidindamas pakaitalą L parametrų atžvilgiu naudojant K epochas. Tai nurodo, kiek atrinktų veiksmų vertė ankstesniame modeliavime yra geresnė už apskaičiuotą vertę iš verčių tinklo. Dokumente teigiama, kad ši algoritmas yra efektyvus ir stabilus, ir įrodyta, kad jis veiksmingas gerinant našumą įvairiuose scenarijuose. Formulė:

$$L_t^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{1}$$

Vienas iš šių formulių pranašumas yra tas, kad jos pateikia matematinį neuroninio tinklo operacijų vaizdą, kurį galima naudoti norint suprasti, kaip tinklas veikia ir kaip jis gali būti optimizuotas. Be to, universaliosios aproksimacijos teorema teigia, kad grįžtamojo ryšio tinklas su linijiniu išvesties sluoksniu ir bent vienu paslėptu sluoksniu su tam tikra nelinijine aktyvinimo funkcija gali apytiksliai su tinkamais parametrais suderinti bet kurią funkciją.

Tai reiškia, kad šios formulės gali būti naudojamos kuriant neuroninius tinklus, galinčius apytiksliai suderinti bet kokią funkciją, o tai yra galingas mašininio mokymosi įrankis. Tačiau vienas galimas šių formulių trūkumas yra tas, kad formulės gali būti taikomos ne visų tipų mašininio mokymosi problemoms spręsti, nes skirtingoms problemoms gali prireikti skirtingų tipų neuroninių tinklų arba algoritmų.

Galimi tobulinimai ir apsvarstymai:

1. Hiperparametrų derinimas eksperimentuojant su skirtingais mokymosi tempais, nuolaidų veiksniais ir pranašumo veiksniais. Pavyzdžiui, išbandyti mokymosi greičius (0,001, 0,01 ir 0,1) randant optimalią vertę konkrečiai aplinkai.
2. Tyrinėjimo strategijoje keisti entropijos premiją (S) nuostolių funkcijoje. Didinant arba mažinant jos svorį, norint paskatinti daugiau ar mažiau tyrinėjimo treniruotės metu.
3. Koreguojant gradiento iškirpimą (angl. gradient clipping), pvz. įdiegiant gradiento iškirpimą norint apriboti gradientus dauginimo atgal metu (angl. backpropagation). Nustatant slenkstį, pavyzdžiui, 0,5, kurį viršijus gradientai bus sumažinti.

Unity žaidimo variklio privalumai:

C# biblioteka sumažina priklausomybę nuo programinės įrangos ir leidžia treniruoti neuroninius tinklus žaidimo metu. Kitose bibliotekose, pvz., „Unity ML-Agents“, reikia įdiegti „Python“ ir kai kuriuos mašininio mokymosi įrankius, o tai kai kuriems vartotojams gali būti nepraktiška. Kitas privalumas yra tai, kad bibliotekoje yra tinkinami mokymo proceso vizualizavimo įrankiai. Taip pat tai yra programavimo kalba, kuri yra naudojama Unity (su C#) - yra geresnė programinės įrangos kūrimui nei Python, nes kūrėjai turi daugiau prieigos prie žemo lygio duomenų struktūrų, kad galėtų optimizuoti veikimo laiką. C# taip pat yra stipriai įvesta programavimo kalba (angl. strongly typed programming language), kurią lengviau derinti nei Python.

1.3. Skirtingi sustiprinti mokymosi metodai

Šiame skyriuje yra nagrinėjami skirtingi mokymosi metodai ir jų pritaikymas užduočių sprendimams sukurtoje Unity aplinkoje.

1.3.1. Proximal policy optimisation. Nr. 1

Šaltinyje [KB20] autoriai pateikia išsamią savo tyrimų apie proksimalinės politikos optimizavimo (PPO) taikymą mobiliajame galvosūkių žaidime „Lily's Garden“ analizę. Jie tiria įvairius metodus, įvertina rezultatus ir aptaria savo išvadų pasekmes. Autoriai pradeda pabrėždami, kaip svarbu suprasti skirtingų PPO variantų įtaką mokymosi elgsenai. Jie pabrėžia, kad reikia strategijų, užtikrinančių stabilų algoritmo veikimą mobiliojo galvosūkio žaidimo žanre. Norėdami tai išspręsti, jie siūlo dviejų žingsnių seką treniruojant agentą savo tyrime.

Pirma, autoriai tiria įvairias sąrankas ir hiperparametrus, skirtus mokyti agentą Lily's

Garden žaidime. Jie atlieka eksperimentus naudodami tris bazinius modelius su skirtingais entropijos koeficientais. Kiekvieno modelio mokymosi kreivės analizuojamos siekiant įvertinti agento mokymosi pažangą. Rezultatai rodo, kad agentas greitai mokosi ir pagerina savo veiklą, o tai rodo epizodų atlygio padidėjimas. Antra, autoriai vertina apmokyto agento apibendrinimą, išbandydami jį tiek matytu, tiek nematytu lygiu - matuodami galiojančius judesių procentus ir užbaigimo rodiklius, įvertinant agento kompetenciją suprasti žaidimo mechaniką ir sėkmingai užbaigti lygius. Vertinimai po treniruotės rodo, kad agentas kai kuriais lygmenimis dirba gerai, bet sunkiai su kitais, o tai rodo, kad reikia toliau tobulinti algoritmą.

Autoriai pateikia bazinio modelio, kurio entropijos koeficientas (EC) yra 0,001, mokymosi kreivių ir veiklos metrikų analizę. Žvelgiant į bazinio EC: 0,001 modelio mokymosi kreivę paveikslėlyje (šaltinio 5 figūra), akivaizdu, kad agentas greitai išmoksta ir pagerina savo veiklą, o tai rodo epizodų atlygio padidėjimas. Tačiau pasiekus 400 000 žingsnių, epizodo atlygis smarkiai sumažėja, todėl treniruotės visiškai nutrūksta. Toks elgesys buvo pastebėtas ir kituose eksperimentuose pradinės analizės metu. Ši problema yra siejama su pakankamai žema veiksmų entropija, dėl kurios agentas pakartotinai pasirenka tą patį blogą veiksmą ir užpildo mokymo pavyzdžius blogais stebėjimais. Be to, problemą apsunkina tai, kad negaliojantys veiksmai nekeičia žaidimo būsenos, o algoritmas netrukdo pasirinkti negaliojančių veiksmų. Dėl to mokymo duomenų pavyzdžiai tampa identiški ir mokymosi procesas nutrūksta.

PPO apima kelis mini paketų atnaujinimus kiekvienam duomenų pavyzdžiui. Tačiau tikslus PPO diegimas skirtingose bibliotekose gali šiek tiek skirtis dėl papildomų funkcijų, pvz., vertės mastelio keitimo arba paketinio normalizavimo. Autoriai siūlo optimizuoti PPO funkciją kelių nuostolių funkcijų suma, pavaizduota formule:

$$L^{\text{CLIP+VF+S}}(\theta)_t = \hat{\mathbb{E}}_t \left[L^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta](\theta) \right] \tag{2}$$

Šioje lygtyje $L^{CLIP+VF+S}(\theta)_t$ reiškia nukirptą pakaitinę tikslo funkciją, $L_t^{VF}(\theta)$ nurodo vertės funkcijos kvadrato klaidos praradimą, S yra entropijos priedas, o c1 ir c2 yra koeficientai. Terminas $L^{CLIP+VF+S}(\theta)_t$ užtikrina, kad politikos atnaujinimai nebūtų per dideli, o $L_t^{VF}(\theta)$ užtikrina, kad būtų atsižvelgta į neuroninių tinklų politikos ir vertės funkcijų nuostolius. Entropijos terminas S skatina labiau atsitiktinę politiką (skatinančią tyrinėjimą), o tai galima padaryti su c2 koeficientu. Apskritai, PPO yra metodas, optimizuojantis pakaitinę tikslo funkciją, naudojant kelis miniatiūrinius atnaujinimus vienam duomenų pavyzdžiui. Konkretus įgyvendinimas įvairiose bibliotekose gali šiek tiek skirtis, tačiau pagrindinė idėja yra optimizuoti kelių praradimo funkcijų sumą, įskaitant nukirptą pakaitinę tikslo funkciją, vertės funkcijos kvadrato klaidos praradimą ir entropijos priedą.

Norėdami palyginti agento veiklą su žaidėjais, autoriai įtraukia faktinius žaidėjų duomenis: jie vizualizuoja judesių, reikalingų norint užbaigti lygį, pasiskirstymą ir palygina jį su žmogaus duomenimis. Ši analizė padeda lygio dizaineriams nustatyti judėjimo ribą ir įvertinti, ar agentas elgiasi kaip žmogus. Tačiau autoriai pažymi, kad nė vienas iš modelių nuosekliai

nerodo geresnio elgesio, kaip žmogaus ar žemesnio, pabrėžiant tolesnio tyrimo ir tobulinimo poreikį. Taip pat aptaria automatizuotų sistemų, tokių kaip PPO, naudojimo gamybos aplinkoje iššūkius, pabrėždami patikimumo ir prieinamumo svarbą, nes SM sistemos paprastai yra trapios ir priklauso nuo įgyvendinimo. Norint sukurti žaidimo testavimo įrankį, kurį būtų galima lengvai pataisyti ir neeikvoti išteklių, kai kyla problemų, reikia stabilių diegimų ir strategijų.

Apskritai, autorių tyrimas apie proksimalinės politikos optimizavimo taikymą Lily's Garden žaidime suteikia vertingų įžvalgų apie metodus, rezultatus ir iššūkius, susijusius su agento mokymu mobiliajame galvosūkių žaidime. Mokymosi kreivių, galiojančių judesių procentų, atlikimo rodiklių ir palyginimų su žmonių duomenimis analizė padeda geriau suprasti agento veiklą ir pabrėžia tobulinimo sritis.

Privalumai:

Šiame darbe pagrindinis dėmesys skiriamas PPO patikimumo gerinimui treniruočių metu ir apibendrinimui žaidimo metu. Įdiegdami ir išbandydami skirtingas strategijas, autoriai siekia užtikrinti stabilesnį algoritmo elgesį mobiliuosiuose galvosūkių žaidimuose. Pabrėžiamas PPO automatizavimo potencialas atliekant žaidimų testavimą ir žaidimų turinio vertinimą. Naudodami prižiūrimo ir sustiprinto mokymosi algoritmai, PPO gali prisidėti prie žaidimų turinio testavimo automatizavimo, todėl jis tampa efektyvesnis ir reikalauja mažiau darbo.

Trūkumai:

Dokumente minima, kad kai kurie PPO variantai nėra visiškai suprantami ir gali sukelti netikėtą mokymosi elgesį. Dėl šio neapibrėžtumo gali kilti sunkumų efektyviai naudojant PPO, ir dėl to gali prireikti tolimesnio tyrimo ir eksperimentų. PPO, kaip ir kiti SM metodai, gali būti trapus ir priklausomas nuo įgyvendinimo - PPO stabilumas ir mastelio keitimas gamybos (angl. production) aplinkoje gali kelti susirūpinimą, nes gali pareikalauti papildomų išteklių ir pastangų norint užtikrinti patikimumą ir prieinamumą. Taip pat aptariamas negaliojančių veiksmų, galinčių turėti įtakos mokymo procesui, klausimas. Autoriai pažymi, kad netinkamų veiksmų pasirinkimas gali sukelti identiškus mokymo duomenų pavyzdžius ir sutrikdyti mokymąsi. Siekiant veiksmingo mokymo, labai svarbu išspręsti šią problemą ir neleisti algoritmui pasirinkti netinkamų veiksmų.

1.3.2. Proximal policy optimisation. Nr. 2

Šiame straipsnyje [HDW+23] nagrinėjamas proksimalinės politikos optimizavimo (PPO) algoritmo taikymas sprendžiant stochastinės talpos partijos dydžio problemą (S-CLSP). PPO algoritmas yra sustiprinimo mokymosi technika, kuri pastaraisiais metais išpopuliarėjo dėl gebėjimo spręsti sudėtingas optimizavimo problemas. S-CLSP yra gerai žinoma gamybos planavimo ir atsargų valdymo problema, kurios tikslas yra nustatyti optimalius gamybos kiekius per planavimo laikotarpį, atsižvelgiant į pajėgumų apribojimus ir neapibrėžtą paklausą. Stochastinės talpos partijos dydžio problema (S-CLSP) yra sudėtinga optimizavimo problema, kylanti įvairiose pramonės šakose, įskaitant gamybą, logistiką ir tiekimo grandinės valdymą. Tradiciniai S-CLSP sprendimo būdai dažnai remiasi euristika arba matematiniais programavimo metodais,

kurie ne visada gali pateikti optimalius sprendimus arba nesugebėti susidoroti su paklausos neapibrėžtumu. Pastaraisiais metais buvo naudojami sustiprinimo mokymosi algoritmai, tokie kaip proksimalinės politikos optimizavimas (PPO). Algoritmas, parodė perspektyvumą sprendžiant sudėtingas optimizavimo problemas. PPO algoritmas yra politikos optimizavimo metodas, apjungiantis tiek vertybiniais, tiek politika pagrįstų metodų privalumus. Jis buvo sėkmingai pritaikytas įvairiose srityse, įskaitant robotiką, žaidimų žaidimą, o dabar, šiame tyrime - S-CLSP.

Tyrimas naudoja PPO algoritmą, sprendžiant S-CLSP problemą. Algoritmas mokomas naudojant modeliavimu pagrįstą metodą, kai generuojami skirtingi problemų atvejai su skirtingais paklausos modeliais ir pajėgumų apribojimais. PPO algoritmas mokosi iš šių atvejų ir koreguoja savo politiką, kad planavimo laikotarpiu optimizuotų gamybos kiekius. Eksperimentiniai rezultatai parodo PPO algoritmo efektyvumą sprendžiant S-CLSP - algoritmas pasiekia aukštą našumo lygį, matuojant gama paslaugų lygiu, o tai rodo galimybę patenkinti klientų poreikiams. PPO algoritmas lenkia tradicinę euristiką, pvz., Vidutinio medžiagų balanso strategiją (AMBS), pagal vidutines išlaidas, atsargų lygius, užsakymus, sąrankas ir partijų dydžius.

Šio tyrimo išvadose pabrėžiamas PPO algoritmo potencialas sprendžiant S-CLSP keliamus iššūkius. Algoritmo gebėjimas išmokti politiką, kuri subalansuoja skirtingus sąnaudų komponentus, ir jo mastelio keitimas didesnių problemų atveju, todėl jis yra vertinga gamybos planavimo ir atsargų valdymo priemonė. Be to, PPO algoritmo veikimas neapibrėžtoje ir dinamiškoje aplinkoje, kur paklausos modeliai gali skirtis, rodo jo tvirtumą ir pritaikomumą.

Algoritmas naudoja veikėjo-kritiko struktūrą, kur politika, atstovaujama neuroninio tinklo su parametrais θ, vadovauja sprendimų priėmimui, o kritikų tinklas su parametrais φ įvertina vertės funkciją pagal politikos veiksmus. Algoritmas atnaujina neuroninio tinklo parametrus treniruotės metu, kad pagerintų našumą ir iškirpus atnaujinimus iš senos į naują politiką, išvengiama didelių politikos skirtumų, užtikrinant stabilų mokymo elgesį. Treniruočių duomenys, susidedantys iš būsenų, veiksmų ir atlygių, yra saugomi buferyje, kad būtų galima apskaičiuoti nuostolių funkcijas, reikalingas tinklo svoriams atnaujinti ir keli atnaujinimai atliekami naudojant tuos pačius mokymo duomenis, siekiant pagerinti mokymąsi. Neuroninio tinklo architektūra apima visiškai sujungtus pirmyn nukreiptus tinklus su dviem paslėptais 256 mazgų sluoksniais, naudojant hiperbolinės liestinės aktyvinimo funkciją. Didesniems problemų atvejams naudojami papildomi paslėpti sluoksniai su 512 mazgų. Aktorių tinklo įvesties ir išvesties sluoksniai atitinka atitinkamai būsenos ir veiksmo erdves - tačiau kritikų tinklas atspindi veikėjų tinklo struktūrą, bet išveda vertės funkcijos įvertinimą. Neuroniniam tinklui inicijuoti naudojamas Glorot ir Bengio metodas nuo 2010 m., naudojant Glorot-Uniform inicijavimą svoriams ir inicijavimo poslinkius iki 0. Šis inicijavimo metodas padeda skleisti gradientą ir greičiau suartėti treniruočių metu, taip pat būsenos ir veiksmų erdvės koreguojamos siekiant užtikrinti, kad neuroninis tinklas efektyviai apdorotų įvestis. Būsenos erdvės reikšmės padidinamos iki [-1, 1], o riba nustatoma pagal vidutinę paklausos normą. Šis mastelio keitimas padeda susiaurinti potencialią būsenos erdvę norint efektyviau tyrinėti. Be to, naudojama

atskira veiksmų erdvė, ribojanti galimų veiksmų rinkinį, siekiant pagerinti algoritmo našumą ir sumažinti skaičiavimo išlaidas.

Tinklo svorių atnaujinimo procese su Adam optimizatoriumi naudojama mini partijos gradiento mažinimo technika, apimanti 1e−4 mokymosi greitį. Šis metodas buvo veiksmingas remiantis ankstesniais tyrimais, kai mažojo paketo gradiento mažinimas apima duomenų padalijimą į paketus, apskaičiuojant vidutinį kiekvienos partijos gradientą. Kai partijos dydis yra 64, o sutrumpintas buferis yra 256, iš viso yra 4 partijos. Mokymo duomenys kartojami 10 kartų (epochas), todėl kiekvienoje mokymo iteracijoje iš viso atnaujinama 40 tinklų. Kad būtų išvengta per didelio suderinimo, kiekvienoje partijoje duomenys maišomi, koreliuojant atnaujinimus. Kiekvieno atnaujinimo metu apskaičiuojamos dvi praradimo funkcijos:

$$L_a \text{ naudojame } \pi\left(s_t;\theta\right), \text{ o } L_c \text{ naudojame } v^{\pi}\left(s_t;\phi\right)$$

Aktorių praradimo funkcija naudoja vidutinį visų partijos imties taškų praradimą, atsižvelgiant į apskaičiuotus pranašumus ir atnaujintos politikos santykį su sena politika. Dėl netiesinio neuroninių tinklų pobūdžio politikos koregavimai gali žymiai pakeisti būsenos pasiskirstymą, kurį aplanko atnaujinta politika, ir gali nukrypti nuo numatyto paskirstymo. Norėdami tai sušvelninti, PPO metodas naudoja tikslinės funkcijos apkarpymą, kad politikos paskirstymo pokyčiai būtų minimalūs, užtikrinant stabilumą mokymo metu. Atnaujintos politikos ir senosios politikos santykis yra ribojamas $1 \pm \varepsilon$ diapazone, norint išvengti pernelyg didelių politikos tinklo atnaujinimų, skatinančių nuoseklesnį ir stabilesnį mokymo elgesį. Nuostolių funkcija $L_a(\theta)$ apima tikėtiną vertę laikui bėgant, sumažindama $r_t(\theta)$ ir $\hat{A}_t$ sandaugą, nukirptą tarp $1 - \varepsilon$ ir $1 + \varepsilon$, kartu su papildomu terminu $c_e \cdot S$ tolesniam koregavimui.

Ši formulė nurodo praradimo funkciją, naudojamą atnaujinant tinklo svorius proksimalinės politikos optimizavimo (PPO) metode:

$$L_a(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) + c_e \cdot S \right] \tag{3}$$

- $\hat{\mathbb{E}}_t$ žymi numatomą vertę laikui bėgant, nurodant vidutinę atsitiktinio dydžio vertę per laikotarpį.
- min apskaičiuoja mažiausią reikšmę tarp dviejų reikšmių.
- $r_t(\theta)$ konkreti reikšmė, susijusi su politikos tinklu.
- $\hat{A}_t$ žymi veiksmą, atliktą tam tikru laiko tarpe.
- clip $\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)$ ši dalis apima $r_t(\theta)$ vertės apkarpymą tarp 1 - ε ir 1 + ε. Apkarpymas užtikrina, kad politikos tinklo atnaujinimai nebūtų per dideli, skatinant stabilų treniruočių elgesį.
- $c_e \cdot S$ šis komponentas įveda entropijos priedą politikos praradimo funkcijai, skatinančią tyrinėti veiksmus, ypač scenarijuose su didelėmis veiksmų erdvėmis.

Formulė apskaičiuoja nuostolių funkciją, kuri nukreipia PPO algoritmo politikos tinklo atnaujinimus, subalansuodama veiksmų tyrimą ir išlaikydama stabilumą treniruočių metu.

Privalumai (PPO): Algoritmas puikiai veikia be didelio hiperparametrų derinimo ir priartėja prie optimalių sprendimų. Pramoninio naudojimo etalonus jis lenkia maždaug 7% . Gali būti pritaikytas įtraukiant papildomą informaciją, pvz., paklausos prognozes, norint lengviau priimti sprendimus esant sudėtingiems realaus gyvenimo apribojimams.

Trūkumai (PPO): Didesnės veiksmų erdvės gali apriboti efektyvių sprendimų paiešką, o tai gali turėti įtakos algoritmo veikimui. Kadangi veiksmų erdvė labai padidėja dėl didesnio kiekio produktų ar pajėgumų apribojimų, gali padidėti skaičiavimo išlaidos, o tai turi įtakos algoritmo efektyvumui. Taip pat, norint išvengti per didelio pritaikymo ir išlaikyti pastovų našumą, labai svarbu užtikrinti stabilų treniruočių elgesį, ypač atnaujinant tinklo svorį.

Galimas PPO panaudojimas:

1. 1 pavyzdys: Tarkime, kad tam tikro laiko žingsnio t atlygis (r_t) yra 0,5, pranašumo įvertinimas $\hat{A}_t$ yra 0,2, kirpimo parametras (ϵ) yra 0,1, o entropijos koeficientas c_e yra 0,01. Be to, vidutinės entropijos (S) reikšmė yra 0,05. Įtraukus šias reikšmes į formulę, gauname: $L_a(\theta)$ = $\hat{\mathbb{E}}_t$ [min(0,5 * 0,2, clip(0,5, 1-0,1, 1+0,1) * 0,2) + 0,01 * 0,05]
2. 2 pavyzdys: Tarkime, kad tam tikro laiko žingsnio t atlygis (r_t) 0,8, pranašumo įvertinimas $\hat{A}_t$ yra 0,3, kirpimo parametras (ϵ) yra 0,2 ir entropijos koeficientas (c_e) yra 0,02. Vidutinė entropija (S) yra 0,1. Pritaikius šias reikšmes formulei, gauname: $L_a(\theta) = \hat{\mathbb{E}}_t$ [min(0,8 * 0,3, clip(0,8, 1-0,2, 1+0,2) * 0,3) + 0,02 * 0,1]

1.3.3. Proximal policy optimisation. Nr. 3

Pastaraisiais metais buvo pasiūlyta keletas skirtingų metodų, kaip sustiprinti mokymąsi su neuroninių tinklų funkcijų aproksimatoriais. Vienas iš panaudojimų yra gilus Q mokymasis [MKS+15], „vanilės“ politikos gradiento metodai [MBM+16] ir pasitikėjimo regiono / gamtos politikos gradiento metodai [SLM+17]. Tačiau yra galimybių tobulinti metodą, kurį galima keisti (iki didelių modelių ir lygiagretaus diegimo), kuris yra veiksmingas ir patikimas (t. y. sėkmingas įvairiose srityse, sprendžiant problemas be hiperparametrų derinimo). Q-mokymasis (su funkcijos aproksimacija) neišsprendžia daug paprastų problemų ir yra menkai suprantamas: „vanilės“ politikos gradiento metodai turi prastą duomenų efektyvumą ir tvirtumą ir pasitikėjimo regiono politikos optimizavimas (TRPO) yra gana sudėtingas ir nesuderinamas su architektūromis, kuriose yra triukšmo (pvz., iškritimo) arba parametrų bendrinimo (tarp politikos ir vertės funkcijų arba su pagalbinėmis užduotimis).

Straipsnyje [SWD+17a] siūlomas naujas politikos gradiento metodų rinkinys, skirtas sustiprinti mokymąsi, kai kaitaliojama duomenų atranka, sąveikaujama su aplinka ir optimizuojama „pakaitinė“ tikslo funkcija. Naudojamas stochastinis gradiento kilimas. Nors standartiniai politikos gradiento metodai atlieka vieną gradiento atnaujinimą kiekvienam duomenų pavyzdžiui, siūlomas naujas tikslas: funkcija, kuri įgalina kelias mažas paketų atnaujinimų epochas.

Nauji metodai, vadinami proksimalinės politikos optimizavimu (PPO), turi tam tikrų pasitikėjimo regiono politikos optimizavimo (TRPO) pranašumų, juos daug lengviau įgyvendinti, jie yra bendresni ir turi geresnį sudėtingumo pavyzdį (empiriškai). Eksperimente išbandytas PPO, atliekant etalonines užduotis, įskaitant imituojamą robotų judėjimą ir žaidimą „Atari“. Parodyta, kad PPO yra geresnis už kitus internetinės politikos gradiento metodus ir sukuria palankią imties pusiausvyrą tarp sudėtingumo, paprastumo ir sienos laiko. PPO turi pasitikėjimo regiono metodų stabilumą ir patikimumą, tačiau yra daug paprasčiau įgyvendinamas, reikalaujantis pakeisti tik kelias kodo eilutes į vanilės politikos gradiento įgyvendinimą, pritaikomas daug plačiau (pavyzdžiui, kai politikos ir vertės funkcijai naudojama bendra architektūra) ir turi geresnį bendrą našumą.

Šiuo straipsniu siekiama pagerinti dabartinę padėtį, įdiegiant algoritmą, kuris pasiekia duomenų efektyvumą ir patikimą TRPO veikimą, naudojant tik pirmos eilės optimizavimą. Siūlomas naujas tikslas su „nukirptais“ tikimybių koeficientais, kurie sudaro pesimistinį politikos vykdymo įvertinimą (t. y. apatinė riba). Norint optimizuoti politiką, keičiamas duomenų atrinkimas iš politikos ir atliekama keletą atrinktų duomenų optimizavimo epochų. Eksperimentai lygina įvairių skirtingų pakaitinio tikslo versijų našumą ir nustato, kad versija su „nukirptais‘‘ tikimybės koeficientais veikia geriausiai. Taip pat palyginta PPO prie kelių ankstesnių algoritmų iš literatūros. Ji atlieka nuolatinio valdymo užduotis geriau nei algoritmai, su kuriais lyginame. „Atari“ veikia žymiai geriau (kalbant apie imties sudėtingumą) nei A2C ir panašiai kaip ACER, nors jis yra daug paprastesnis.

Straipsnyje siūlomas algoritmas apima nedidelį tipinio politikos gradiento diegimo pakeitimą, įdiegimams, kuriuose naudojamas automatinis diferencijavimas ir tuo pačiu praradimo funkcija sukonstruota kaip LCLIP arba LKLPEN, o ne kaip LPG. Šis pakeitimas leidžia apskaičiuoti ir diferencijuoti pakaitinius nuostolius ir tada šiam tikslui atliekami keli stochastinio gradiento kilimo žingsniai. Norint apskaičiuoti sumažintų dispersijų pranašumų ir funkcijų įverčius, algoritmas naudoja išmoktą būsenos vertės funkciją V(s). Šią funkciją galima išgauti naudojant tokius metodus kaip apibendrintas pranašumų įvertinimas (angl. generalized advantage estimation) arba baigtinio horizonto įvertinimai (angl. finite-horizon estimators). Jei naudojama neuroninio tinklo architektūra, kuri dalijasi parametrais tarp strategijos ir vertės funkcijų, reikalinga praradimo funkcija, kuri apjungia politikos pakaitalą ir vertės funkcijos klaidos terminą. Siekiant paskatinti politikos tyrinėjimą, prie tikslo galima pridėti entropijos priedą. Ši premija padeda užtikrinti pakankamą tyrinėjimą mokymosi proceso metu. Tikslas, kuris yra apytiksliai maksimalus kiekvienos iteracijos metu, pateikiamas lygtimi:

$$L_t^{\text{CLIP+VF+S}}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S \left[\pi_\theta\right](s_t) \right] \tag{4}$$

Čia c1 ir c2 yra koeficientai, S žymi entropijos premiją, o L_t^{VF} yra kvadratinės klaidos praradimo terminas. Apskritai, šis algoritmas suteikia pagrindą skaičiuojant pakaitinius nuosto-

lius ir optimizuojant politiką įgyvendinant politikos gradientą, taip pat įtraukiant dispersijos mažinimo ir tyrimo metodus.

Pakaitinių nuostolių skaičiavimo algoritmas įgyvendinant politikos gradientą: eksperimente buvo lyginami keli skirtingi pakaitiniai tikslai pagal skirtingus hiperparametrus kaip ir surogatinis tikslas LCLIP prie kelių natūralių variantų:

1. Jokio apkirpimo ar baudos (angl. no clipping or penalty): $L_t(\theta) = r_t(\theta)\hat{A}_t$
2. Su apkarpymu (angl. clipping): $L_t(\theta) = \min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta)\right), 1-\epsilon, 1+\epsilon\right)\hat{A}_t$
3. KL bauda (fiksuota arba adaptyvi): $L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\text{KL}\left[\pi_{\theta_{\text{old}}}, \pi_\theta\right]$

KL baudai galima naudoti fiksuotą baudos koeficientą β arba adaptyvųjį koeficientą naudojant tikslinę KL reikšmę dtarg. Taip pat buvo bandyta iškirpti pačioje erdvėje, tačiau buvo prieita prie išvados, kad tai nebuvo geresnis sprendimas. Kadangi ieškomas kiekvieno algoritmo variantas su skirtingais hiperparametrais, algoritmams išbandyti buvo pasirinktas „pigus“ skaičiavimo etalonas. Nuspręsta panaudoti 7 imituotas robotikos užduotis, kur įdiegta OpenAI Gym [BCP+16], kurioje naudojamas MuJoCo [TET12] fizinis variklis. Buvo treniruojama po vieną milijoną treniruočių kartų be hiperparametrų, naudojamų karpymui (ϵ) ir KL baudos (β, dtarg), kurios ieškoma. Norint parodyti politiką, naudojomas prijungtas MLP su dviem paslėptais 64 sluoksniais, ir tanh netiesiškumą, išvedant Gauso skirstinio vidurkį su kintamu standartu nukrypimams. Nebuvo dalinamasi parametrais tarp politikos ir vertės funkcijų (taigi koeficientas c1 šio atveju nėra svarbus), taip pat nebuvo naudojama entropijos premija. Kiekvienas algoritmas buvo paleistas visose 7 aplinkose, kiekvienoje po 3 atsitiktines sėklas (angl. seeds). Kiekvienas paleistas algoritmas, apskaičiuojant vidutinį bendrą atlygį už paskutinius 100 epizodų. Kiekvienos aplinkos balai buvo keičiami taip, kad atsitiktinė politika duotų 0 ir geriausią balą, to pasėkoje rezultatas buvo nustatytas į 1. Autoriai atkreipia dėmesį, kad įvertinimas yra neigiamas, jeigu nėra nenukirpta (angl. without clipping) arba baudos, nes vienai aplinkai gaunamas labai neigiamas balas, kuris yra blogiau nei pradinė atsitiktinė politika.

Privalumai:

Algoritmas sukurtas spręsti didelio masto problemas ir lygiagrečius įgyvendinimus, todėl jis tinka sudėtingoms aplinkoms su didelės apimties veiksmų erdvėmis. Jis taip pat yra efektyvus duomenims, o tai reiškia, kad jis gali pasiekti gerą našumą naudojant mažiau pavyzdžių, palyginti su kitais sustiprinimo mokymosi algoritmais. Tai naudinga, kai duomenų rinkimas atima daug kaštų (yra brangu) arba atima daug laiko. Be to, algoritmas yra tvirtas ir gali prisitaikyti prie skirtingų scenarijų nereikalaujant didelių pakeitimų. Jis pašalina kitų esamų metodų, pvz., Q-mokymosi ir vanilės politikos gradiento metodų, apribojimus, pateikdamas labiau keičiamo dydžio, efektyvesnį ir patikimesnį sprendimą.

Trūkumai:

Algoritmas priklauso nuo kelių hiperparametrų, tokių kaip c1 ir c2, kuriuos reikia atsargiai koreguoti, kad būtų užtikrintas optimalus veikimas. Pasirinkus netinkamas šių hiperparametrų

vertes, rezultatai gali būti neoptimalūs. Antra, nors algoritmas apima entropijos premijos terminą, skatinantį tyrinėjimą, vis tiek gali būti sunku efektyviai ištirti sudėtingą aplinką. Dėl to algoritmas gali „užstrigti“ neoptimalioje politikoje. Be to, algoritmui gali prireikti didelių skaičiavimo išteklių, ypač sprendžiant didelio masto problemas arba sudėtingas neuroninių tinklų architektūras, nes politikos ir vertės funkcijų tinklų mokymas gali būti intensyvus skaičiavimas. Nors algoritmas parodė daug žadančius empirinius rezultatus, jam gali trūkti tvirtų teorinių garantijų, palyginti su kai kuriais kitais sustiprinimo mokymosi algoritmais.

Galimi patobulinimai PPO algoritmui: PPO galėtų veikti geriau naudojant metodus kaip epsiloninis godus (angl. epsilon-greedy) tyrinėjimas, kai agentas pasirenka atsitiktinį veiksmą su maža tikimybe (pvz., 0,1) paskatinant tyrinėjimą ir atrandant geresnę politiką. Kitas patobulinimas galėtų būti adaptyvaus mokymosi rodiklių įdiegimas (angl. adaptive learning rates). Pavyzdžiui, PPO galėtų naudoti tokius metodus kaip mokymosi greičio mažėjimas: kai mokymosi greitis laikui bėgant mažėja, kad būtų galima tiksliai suderinti politikos tinklą ir pagerinti konvergenciją.

1.3.4. Proximal policy optimisation. Nr. 4

Šaltinyje siūlomas naujas metodas „Fast-PPO“ [XXY+20], kuriuo siekiama padidinti politikos atnaujinimo greitį, palyginant su tradiciniais PPO algoritmais. Straipsnio autoriai atliko eksperimentus, siekdami parodyti siūlomo metodo efektyvumą ir palygino jį su pagrindiniais sustiprinimo mokymosi (SM) algoritmais. Autoriai pradeda pristatydami „Fast-PPO“ algoritmą, kuris apima optimalų pradinį lygį pagerinant PPO metodų efektyvumą ir paaiškina, kad jų požiūrio idėja buvo įkvėpta moderniausio politikos gradiento (PG) metodo, vadinamo politikos gradientu su parametrais pagrįstu tyrinėjimu (PGPE) (angl. policy gradient with parameter-based exploration) su optimaliu pradinės linijos atėmimu. Derindami dispersijos reguliavimo metodus su parametrais pagrįstu tyrimu ir optimalia bazine linija, autoriai siekia atrasti tinkamą politikos atnaujinimo kryptį. Norėdami įvertinti „Fast-PPO“ našumą, autoriai atliko eksperimentus naudodami „Unity“ aplinkas, konkrečiai valdydami ne tikrus žaidėjus (NPC) žaidimuose. Jie palygino „Fast-PPO“ rezultatus su kitais SM algoritmais ir nustatė, kad „Fast-PPO“ pasiekė didesnį atlygį. Autoriai taip pat teigia, kad „Fast-PPO“ konvergencijos greitis yra greitesnis, palyginti su kitais algoritmais. Eksperimentuose buvo apmokyti agentai, naudojantys „Fast-PPO“ ir kitus algoritmus, o gautos strategijos išsaugomos kaip „.byte“ failai. Šie failai buvo naudojami kontroliuoti agentų elgesį teniso žaidime, kurį taip pat sukūrė autoriai. Eksperimentai parodė, kad „Fast-PPO“ viršijo kitus algoritmus pagal kaupiamą atlygį (tai galima pastebėti šaltinio 3 figūroje).

Be to, buvo atlikti papildomi eksperimentai pasitelkiant 3DBall žaidimu, lyginant Fast-PPO ir PPO konvergencijos tendencijas. Buvo nustatyta, kad „Fast-PPO“ buvo stabilesnis nei PPO, o standartinis nuokrypis buvo mažesnis. Tačiau, PG ir R-PGPEOB algoritmai pakankamai anksti pasiekė vietines didžiausias reikšmes ir nebegalėjo mokytis iš aplinkos. Autoriai taip pat aptaria tradicinių SM algoritmų, tokių kaip politikos gradiento (PG) algoritmo

ir pasitikėjimo regiono politikos optimizavimo (TRPO) algoritmo, apribojimus. Jie teigia, kad TRPO yra sudėtingas ir nesuderinamas dėl tinklo elementų atmetimo ir parametrų dalijimosi tarp politikos ir vertės funkcijų, o pavyzdžiui PPO - lengviau įgyvendinamas ir efektyvesnis. Autoriai pateikia PPO algoritme naudojamą tikslo funkciją, kuri remiasi suvaržytos politikos tobulinimo (CPI) metodu.

Autoriai taip pat siūlo naudoti Actor-Critic algoritmą sprendžiant gilaus sustiprinimo mokymosi (DRL) problemas nepertraukiamo judėjimo erdvėse. Tačiau AC sudaro du modeliai, būtent Aktorius ir Kritikas, kuriems reikalingas atskiras mokymas ir atitinkamų svorių rinkinių optimizavimas. Kritiko modelis apskaičiuoja veiksmo balą (q reikšmę) dabartinėje būsenoje, o veikėjo modelis naudoja šią q reikšmę savo politikos svoriui atnaujinti. Nepaisant to, vertės funkcijomis pagrįstas metodas dažnai rodo reikšmingus mokymo svyravimus. Norėdami išspręsti šią problemą, A3C algoritmas suteikia pranašumo funkciją, kuri kiekybiškai įvertina vidutinio veiksmo pranašumo vertę, lyginant su dabartine būsena. Siekdami panaudoti parametrų pasidalijimo tarp politikos ir vertės funkcijų koncepciją naudojant neuroninio tinklo architektūrą, taip pat yra siūloma praradimo funkcija, kuri sujungia politikos pakaitalą ir vertės funkcijos klaidos terminą. Šį tikslą galima dar labiau sustiprinti įtraukiant entropijos priedą, kad būtų užtikrintas tinkamas tyrinėjimas, kaip buvo pasiūlyta ankstesniuose tyrimuose, kurį minėjo autoriai. Šis algoritmas padeda įgyvendinti šią logiką:

$$L_t^{\text{Fast-PPO+VF+S}}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{\text{Fast-PPO}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S\left[\pi_\theta\right](s_t) \right] \tag{5}$$

- $L_t^{Fast-PPO+VF+S}(\theta)$: tai yra bendra greitojo PPO algoritmo tikslo funkcija. Tai reiškia kombinuotą nuostolių funkciją, kuri optimizuojama kiekvienos algoritmo iteracijos metu.
- $\hat{E}_t[L_t^{\text{Fast-PPO}}(\theta)]$ reiškia tikėtiną nuostolių funkcijos $L_t^{\text{Fast-PPO}}(\theta)$ reikšmę trajektorijos atkarpoje. Ji apskaičiuoja vidutinį nuostolį per kelias iteracijas.
- $L_t^{\text{Fast-PPO}}(\theta)$: tai politikos pakaitinio nuostolio funkcija, kuri matuoja skirtumą tarp dabartinės ir atnaujintos politikos. Ji naudojama politikos parametrams atnaujinti.
- $c_1 L_t^{VF}(\theta)$ reiškia reikšmės funkcijos paklaidos terminą, kuris matuoja skirtumą tarp numatomos reikšmės funkcijos ir tikslinės reikšmės funkcijos. Jis dauginamas iš koeficiento c1, norint kontroliuoti jo poveikį bendrai tikslo funkcijai.
- $c_2 S\left[\pi_\theta\right](s_t)$ šis terminas reiškia entropijos premiją, kuri skatina tyrinėti politiką. Jis dauginamas iš koeficiento c2, norint kontroliuoti jo poveikį bendrai tikslo funkcijai.
- θ kintamasis nurodo politikos parametrus, kurie optimizuojami naudojant Fast-PPO algoritmą. Jis nuolat atnaujinamas siekiant tobulinti politiką.

Apibendrinant, autoriai siūlo Fast-PPO algoritmą kaip būdą padidinti politikos atnaujinimo greitį teisinga kryptimi, palyginti su tradiciniais PPO algoritmais. Eksperimentais, atliktais „Unity“ aplinkoje, jie parodo, kad „Fast-PPO“ atlygio ir konvergencijos greičio atžvilgiu lenkia pagrindinius SM algoritmus. Teigia, kad „Fast-PPO“ gali būti taikomas norint kontroliuoti NPC elgesį įvairiuose veiksmo žaidimuose. Tačiau svarbu pažymėti, kad dokumente nėra

išsamiai aptariami konkretūs eksperimentuose naudojami metodai ar pasiekti rezultatai, išskyrus atlygio ir konvergencijos greičio palyginimą.

Privalumai:

Šaltinyje siūlomas Fast-PPO algoritmas siekia padidinti politikos atnaujinimo greitį tinkama kryptimi, palyginant su kitais PPO algoritmais. Eksperimentiškai buvo įrodyta, kad jis turi geresnį našumą ir greitesnę konvergenciją nei įprasti SM algoritmai. Fast-PPO algoritmas parodė efektyvumą valdant NPC tiek nuolatinėje, tiek atskiroje žaidimų aplinkoje. Dėl to tai tinkamas pasirinkimas žaidimų kūrėjams, norintiems pagerinti AI valdomų personažų elgesį. Minima, kad Fast-PPO algoritmas gali būti taikomas treniruojant įvairius agentus, pavyzdžiui, valdyti raketes teniso žaidime arba mokyti šunį atnešti lazdas. Tai rodo, kad algoritmas turi daugybę pritaikymų - ne tik žaidimams.

Trūkumai:

Šaltinyje visų pirma lyginamas Fast-PPO algoritmas su PPO algoritmu, todėl sunku įvertinti jo veikimą, lyginant su kitais SM algoritmais. Platesnis palyginimas su platesniu algoritmų spektru leistų atlikti išsamesnį įvertinimą. Nors ir aptariami teoriniai Fast-PPO algoritmo aspektai ir veikimas, jame nepateikiami išsamūs įgyvendinimo žingsniai ar kodo pavyzdžiai. Dėl to kūrėjams savo projektuose šį algoritmą gali būti sunku integruoti.

1.3.5. **Proximal policy optimisation. Nr. 5**

Šaltinyje galima pastebėti platų mokymosi agentų būklės tyrimą mašininio mokymosi ir duomenų mokslo kontekste. Gilinamasi į šios srities ekspertų grupės atliktus tyrimus, daugiausia dėmesio skiriant „Jacobian“ sąlygos poveikiui sustiprinimo mokymosi agentų veikimui ir stabilumui. Pabrėžiama „Jacobian“ sąlygų svarba gerinant mokymosi sustiprinimo (SM) agentų veiklą. O tai kelia du pagrindinius tyrimo klausimus: ar „Jacobian“ sąlygos turi įtakos SM agentų veiklai ir ar „Jacobian“ sąlygos reguliavimas gali padėti formuoti stabilesnę ir bendresnę SM agentų politiką. Tyrėjai atliko tyrimą, tiriant politikos veiksmingumo ir sąlygų santykį, nustatydami, kad geresnė (tuo pačiu ir paprastesnė) politika dažnai yra geriau aprašoma (aprašomos sąlygos). Viena iš pagrindinių tyrimo išvadų yra tokia, kad „Jacobian“ sąlygos daro teigiamą įtaką politikos optimizavimo agento veikimui. Siūlydami sąlygų reguliavimo metodą, mokslininkai siekė pagerinti SM agentų politikos stabilumą ir apibendrinimą. Tyrime pabrėžiama, kaip svarbu kontroliuoti jautrumą įvesties pokyčiams atliekant sustiprinimo mokymosi užduotis, taip pasiekiant stabilesnės politikos.

Straipsnyje aptariami iššūkiai, susiję su politikos atnaujinimu SM, pabrėžiant veiklos žlugimo riziką, kai politika atnaujinama ne optimaliai. Pabrėžiama, kad svarbu pasirinkti tinkamus atnaujinimo žingsnius, išvengiant politikos pablogėjimo ir užtikrinat naudingų pastabų rinkimą siekiant tobulinti politiką. Be to, mokslininkai siūlo algoritmą, kuris naudoja agento sąlygos numerį, sudarant tvirtą politiką, žinomą kaip Jacobian Policy Optimization (JPO). Šis algoritmas įvertina agento „Jacobian“ sąlygos numerį ir atitinkamai pakoreguoja politikos tendenciją. Lygindami JPO su tradiciniais SM algoritmais, tokiais kaip Proksimalinės politikos

optimizavimas (PPO), mokslininkai įrodo, kad JPO užtikrina stabilesnį ir efektyvesnį atlygio augimą įvairaus pobūdžio agentuose. Taip pat yra paliečiamas būklės įvertinimas sustiprinimo mokymosi agentuose, pabrėžiant nuo užduoties priklausomą būklės indėlį į pasiektą atlygį. Įvairios aplinkos struktūros ir dinamika daro įtaką sąlygų poveikio politikos stabilumui ir apibendrinimui, taigi autoriai siūlo algoritmą, kuris apkarpo politikos pakeitimus pagal sąlygas, padidinant mokymosi užduočių stabilumą ir efektyvumą. Be to, aptariama sustiprinimo mokymosi agentų būklė, pateikiama įžvalgų apie politikos optimizavimo metodus, įskaitant „vanilės“ politikos gradiento metodus ir jų apribojimus, susijusius su efektyvumu ir tyrinėjimu. Tačiau autoriai pabrėžia, kad reikia metodų, kurie griežčiau įgyvendintų pasitikėjimo regionus arba sukurtų griežtą pasitikėjimo regiono sušvelninimo teoriją, norint pašalinti šiuos apribojimus.

Siekdami įgyvendinti sąlygų sureguliavimą optimizuodami politiką, tyrėjai naudoja sąlygos numerį kaip nuobaudą. Modifikuota nuostolių funkcija sujungia pradinį PPO politikos praradimą su papildomomis sąlyginės baudos, vertės praradimo ir politikos entropijos sąlygomis. Sąlygų sureguliavimo technika panaudoja sąlygos skaičių kaip nuobaudą, padidinant politikos optimizavimo stabilumą ir efektyvumą atliekant sustiprinimo mokymosi užduotis, ypač atsižvelgiant į proksimalinės politikos optimizavimą (PPO). Tyrime naudojama formulė:

$$L_t^{\mathrm{CLIP}+\psi+\mathrm{VF}+\mathrm{S}}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{\mathrm{CLIP}}(\theta) + c_1 \psi - c_2 L_t^{\mathrm{VF}}(\theta) + c_3 S\left[\pi_\theta\right](s_t) \right] \tag{6}$$

- $L_t^{CLIP+\psi+VF+S}(\theta)$ - modifikuota nuostolių funkcija, apimanti sąlyginį įteisinimą į Proksimalinės politikos optimizavimą (PPO).
- $\hat{\mathbb{E}}_t$ - laukimo operatorius, nurodantis vidutinę skirstinio vertę.
- $L_t^{CLIP}(\theta)$ - pradinis PPO politikos praradimas, matuojant numatomų ir faktinių politikos rezultatų neatitikimą.
- c_1 kondicionavimo baudos koeficientas, koreguojantis kondicionavimo termino poveikį.
- ψ vertė, pridėta prie pakaitinės politikos praradimo funkcijos, atitinkančios sąlyginio įteisinimo terminą.
- c_2 vertės praradimo termino koeficientas, įvertinantis vertės praradimo poveikį bendrajai nuostolių funkcijai.
- $L_t^{VF}(\theta)$ vertės praradimo terminas, kiekybiškai įvertinantis skirtumą tarp numatomos vertės ir tikslinės vertės.
- c_3 politikos entropijos termino koeficientas, koreguojantis politikos entropijos įtaką nuostolių funkcijai.
- $S\left[\pi_\theta\right](s_t)$ - būsenos politikos entropija, matuojanti politikos veiksmų neapibrėžtumą tam tikroje būsenoje.

Taip pat apimamas sustiprinimo mokymosi agentų būklės įvertinimas, daugiausia dėmesio skiriant „Jacobian“ įvesties-išvesties tinklo vidutinės kvadratinės vienaskaitos vertės įvertinimui, naudojant vienaskaitos vertės skaidymą (SVD). Pritaikoma Jacobian Clamping

technika, įvertinant SM agentų Jacobian sąlygas, pabrėžiant efektyvios būklės įvertinimo svarbą greitesniam mokymuisi atliekant SM užduotis.

Apibendrinant, šaltinyje pateikiama išsami ekspertų grupės atlikto tyrimo, tiriančio ir panaudojant sustiprinimo mokymosi agentų būklę, apžvalga. Jų išvados atskleidė „Jacobian" sąlygos svarbą didinant politikos stabilumą, apibendrinimą ir našumą atliekant sustiprinimo mokymosi užduotis. Siūlomais algoritmais ir metodais siekiama spręsti iššūkius, susijusius su politikos optimizavimu, ir pagerinti sustiprinimo mokymosi agentų efektyvumą ir stabilumą įvairiose srityse.

Privalumai:

Pateiktas Jacobian algoritmas suteikia keletą pranašumų mokymosi sustiprinimo srityje. Vienas iš pagrindinių privalumų yra jo gebėjimas padidinti mokymosi efektyvumą, ypač linijiniuose ir netiesiniuose tinkluose, taip pat generuojamuose priešpriešiniuose tinkluose (GAN). Įtraukus Jacobian sąlygų įteisinimą, algoritmas „palengvina" greitesnius mokymosi procesus ir pagerina konvergencijos greitį, o tai leidžia efektyviau mokyti sustiprinimo mokymosi agentus. Be to, algoritmu siekiama pagerinti politikos optimizavimo stabilumą ir apibendrinimą, suteikiant tvirtą pagrindą stabiliai ir patikimai įvairių užduočių strategijų formuluotei.

Trūkumai:

Vienas reikšmingas iššūkis su Jacobian įgyvendinimu yra susijęs sudėtingumas ir skaičiavimo sąnaudos. Jacobian įvesties-išvesties tinklo vidutinės kvadratinės vienaskaitos vertės įvertinimas naudojant tokius metodus kaip vienaskaitos vertės skaidymas (SVD) gali būti intensyvus ir daug laiko reikalaujantis skaičiavimas, todėl jo mastelio keitimas gali būti apribotas iki didesnių tinklų arba realaus laiko taikomųjų programų. Be to, algoritmo jautrumas hiperparametrams ir tinklo architektūrai gali kelti sunkumų siekiant optimalaus našumo įvairiuose nustatymuose, todėl norint maksimaliai padidinti jo efektyvumą, reikia kruopštaus derinimo ir eksperimentavimo.

1.3.6. Generative Adversarial Imitation Learning

Ne žaidėjo veikėjo (NPC) kūrimas vaizdo žaidimuose išlieka dideliu iššūkiu žaidimų pramonėje, šie veikėjai turi sklandžiai orientuotis su žaidimo aplinka ir žaidėjais, taip sustiprindami įtraukiantį potyrį. Tradiciniai NPC elgesio kūrimo metodai dažnai yra atliekami „rankiniu" būdu ir daug darbo reikalaujantys, todėl tam reikia daug skirtingų scenarijų ir testavimo. Atsižvelgiant į didžiulį NPC skaičių šiuolaikiniuose žaidimuose ir jų galimų veiksmų sudėtingumą, rankiniai metodai tampa vis nepraktiškesni. Vienas iš būdų norint to išvengti, yra Mokymosi imitacija (IL) integravimas [CCK+22], kuris siūlo mokant NPC imituoti žaidėjus ir taip sukurti natūralesnį ir į žmogų panašesnį elgesį. Ši technika vis labiau populiarėja, nes leidžia efektyviai programuoti savarankišką elgesį, apeinant išsamaus rankinio scenarijaus sudarymo poreikį.

Mokymosi imitacija (IL) apima agento mokymą stebint ir imituojant žaidėjų veiksmus. Šiame šaltinyje pristatomas IL metodas, kuris panaudoja šiuos duomenis – išsamius žaidėjų veiksmų ir jų konteksto žaidimo metu įrašus. Skirtingai nuo tradicinių žurnalų, duomenys

apima išsamią informaciją apie priežasties ir pasekmės ryšį tarp veiksmų ir žaidimo būsenų. Šis metodas buvo patvirtintas naudojant DodgeBall žaidimą, esantį Unity ML-Agents Toolkit. Žaidėjai, kurie bendravo su apmokytais NPC, pranešė, kad šie veikėjai elgėsi labiau žmogiškai ir įtikinamiau, lyginant su NPC, apmokytais pagal tradicinius atlygio modelius. Modelis, siūlomas mokyti NPC naudojant šiuos duomenis, apima kelis pagrindinius veiksmus:

1. Reikia nustatyti, kuriuos žaidimo veiksmus ir įvestis reikia įrašyti. Pavyzdžiui, DodgeBall žaidime tokie veiksmai kaip kamuolio metimas į priešininką yra labai svarbūs ir duomenys turi apimti visus susijusius veiksmus, pvz., ar žaidėjas nešasi kamuolį.
2. Naudojant tokį įrankį kaip PinGU, žaidimas fiksuoja išsamius duomenis žaidėjo seansų metu, kurie apima įvairias veiklas ir jų kontekstus, būtinus norint suprasti sekas ir sąlygas, kuriomis žaidėjai priima sprendimus.
3. Žaidėjų sesijos atkuriamos naudojant užfiksuotus duomenis, tada NPC yra mokomi imituoti šias sesijas. Skirtingai nuo tradicinių metodų, kuriuose naudojami apdovanojimai ir nuobaudos, šis metodas apdovanoja NPC už veiksmus, kurie atitinka tuos, kurie įrašyti kilmės duomenyse.
4. Mokymosi priešpriešinio imitavimo sistema dar labiau patobulina NPC elgesį. GAIL naudoja generatorių ir diskriminatorių, norint pagerinti NPC gebėjimą imituoti žmogaus veiksmus. Generatorius kuria veiksmus, o diskriminatorius įvertina jų patikimumą pagal kilmės duomenis.

Siūlomo modelio efektyvumas buvo išbandytas DodgeBall žaidime. Septyni studentai žaidė žaidimą prieš NPC, kurie buvo apmokyti pagal tradicinį atlygio modelį ir naujos kilmės pagrįstą modelį. NPC, apmokyti pagal kilmę pagrįstą modelį, gavo aukštesnius įvertinimus už patikimumą. Žaidėjai pastebėjo, kad šių NPC elgesys buvo labiau panašus į žmogų, o tai pagerino bendrą žaidimų patirtį.

Apibendrinant galima pasakyti, kad mokymosi imitacijos naudojimas, pagrįstas kilmės duomenimis, yra didelė pažanga kuriant labiau patikimus ir į žmones panašius NPC vaizdo žaidimuose. Šis metodas ne tik sumažina „rankinių“ pastangų, susijusių su NPC elgsenos scenarijais kūrimu, bet ir pagerina žaidėjų įtraukiantį potyrį, suteikdamas natūralesnę ir patrauklesnę sąveiką.

Galimi GAIL mokymo patobulinimai:

Surinkimas duomenų iš didesnio ir įvairesnio žaidėjų skaičiaus. Tai padeda NPC išmokti įvairesnio elgesio ir prisitaikyti prie skirtingų žaidimo stilių, taip pat rinkti duomenis iš ilgų žaidimo seansų, norint užfiksuoti platesnį scenarijų ir žaidėjo strategijų spektrą. Naudoti sudėtingesnes neuroninių tinklų architektūras, tokias kaip ilgalaikės trumpalaikės atminties (LSTM) tinklai arba transformatorius, kurie gali geriau valdyti laikinąsias priklausomybes ir nuoseklų sprendimų priėmimą. Taip pat GAIL derinimas su kitais sustiprinimo mokymosi metodais, pvz., Proksimalinės politikos optimizavimu (PPO) arba Deep Q-Learning (DQL), norint padidinti mokymosi efektyvumą ir elgesio generavimą. Taip pat vienas iš būdų yra NPC

supažindinimas su vis sudėtingesnėmis užduotimis - šis metodas palaipsniui didina mokymo aplinkos sudėtingumą ir padeda NPC laipsniškai tobulinti savo įgūdžius. Taip pat galimas atlygio funkcijos pakoregavimas, norint pateikti daugiau informatyvių atsiliepimų, įskaitant tarpinius apdovanojimus už dalinį užduočių atlikimą arba nuobaudas už klaidingus veiksmus.

GAIL mokymo trūkumai:

GAIL reikalauja didelių skaičiavimo išteklių dėl priešingo mokymo proceso. Tiek generatorius, tiek diskriminatorius turi būti mokomi vienu metu, dažnai įtraukiant gilius neuroninius tinklus, o tai gali pareikalauti daug išteklių ir atimti daug laiko. GAIL generatorius gali pernelyg prisitaikyti prie konkrečių elgsenų, pastebėtų treniruočių duomenyse. Tai gali sutrikdyti NPC, kurie gerai veikia pagal pažįstamus scenarijus, bet sunkiai apibendrina naujas ar šiek tiek kitokias situacijas.

1.3.7. Actor-Critic Nr. 1

Remiantis šiuo šaltiniu [FHI+18], pristatomi gilaus sustiprinimo mokymosi (SM) modeliai, algoritmai ir metodai. Jame paaiškinama, kad gilus SM yra mašininio mokymosi tipas, apimantis agentą, kuris mokosi sąveikauti su aplinka, optimizuojant tam tikrą tikslą. Agentas gauna grįžtamąjį ryšį kaip atlygį, o jo tikslas yra išmokti politiką, kuri maksimaliai padidintų numatomą kaupiamąjį atlygį. Aptariami veikėjų kritikos metodai, kurie yra SM algoritmo tipas, jungiantis reikšmes ir politika pagrįstus metodus. Aktorių ir kritikų metodai naudoja du neuroninius tinklus: veikėjų tinklą, kuris išmoksta politiką, ir kritikų tinklą, kuris mokinasi pagal reikšmės funkciją. Aktorių tinklas atnaujinamas naudojant politikos gradiento metodą, tačiau kritikų tinklas - pagal laikiną klaidos skirtumą. Plačiai nagrinėjamas Soft Actor-Critic (SAC), kuris yra veikėjo kritiko algoritmo tipas, naudojamas nuolatinėms valdymo užduotims atlikti. Jis buvo kuriamas taip, kad būtų efektyvus ir stabilus, ir tai pasiekiama naudojant minkštosios vertės funkciją (angl. soft value function) ir entropijos reguliavimą. Minkštosios reikšmės funkcija naudojama norint įvertinti numatomą politikos grąžą, o entropijos įgalinimas skatina tyrinėjimą ir neleidžia strategijai tapti pernelyg deterministine. Dokumentas taip pat apima veikėjų kritikų architektūrų naudojimą kelių agentų sistemą. Šioje aplinkoje įprasta naudojama centralizuota kritika mokymosi metu, kuri decentralizuoja veikėją ir leidžia agentams veikti savarankiškai. Dokumente dažnai cituojamas darbas [SLG+17], kuriame buvo tiriamas šis metodas, kurį aprašė Peter Sunehag autorius.

Labai plačiai analizuojama aktoriaus-kritiko architektūra, kuri susideda iš dviejų dalių: aktoriaus ir kritiko. Aktorius nurodo politiką, o kritikas nurodo vertės funkcijos įvertinimą, pavyzdžiui, Q vertės funkciją. Ir veikėją, ir kritiką galima pavaizduoti netiesiniais neuroninių tinklų funkcijų aproksimatoriais. Kritikas įvertina apytikslę dabartinės politikos reikšmės funkciją π. Paprasčiausias ne politikos metodas vertinant kritiką yra naudoti „Bootstrapping“ metodą TD(0), kai dabartinė reikšmė $Q(s, a; \theta)$ atnaujinama į tikslinę vertę. Tačiau šis metodas nėra efektyvus skaičiavimo požiūriu, nes naudojamas Bootstrapping metodas, kuris yra linkęs į nestabilumą ir lėto atlygio gavimą. Tačiau ideali architektūra turėtų būti efektyvi imčių

ir skaičiavimo požiūriu. Yra daug metodų, kurie sujungia politikos įgalinus ne politinius duomenis, norint įvertinti politiką, o „Retrace“ algoritmas yra vienas iš tokių metodų, kuris gali panaudoti pavyzdžius, surinktus iš bet kokios elgesio politikos, neįvedant šališkumo ir yra veiksmingas, nes leidžia geriausiai panaudoti pavyzdžius, surinktus iš politikos elgesio politikos. Aktorių kritikų architektūros, naudojančios šį algoritmą, yra veiksmingos pavyzdyje, nes naudojama atkūrimo atmintis, ir efektyvaus skaičiavimo požiūriu, nes jos naudoja kelių žingsnių grąžą, o tai pagerina mokymosi stabilumą ir padidina atlygio gavimą.

Kritikas yra funkcija, kuri įvertina laukiamą atlygį už veiksmą tam tikroje būsenoje ir tam tikros politikos laikymąsi. Formulė naudojama atnaujinant Q funkciją, kuri yra funkcija, įvertinanti numatomą atlygį už veiksmą tam tikroje būsenoje ir bet kokios politikos laikymąsi po to.

$$Y_k^Q = r + \gamma Q\left(s', a = \pi\left(s'\right); \theta\right) \tag{7}$$

Čia Y_k^Q – dabartinė Q funkcijos reikšmė, kuri atnaujinama kiekvienos iteracijos metu; r – šis kintamasis parodo atlygį, gautą už dabartinę būsenos ir veiksmų porą; γ – nuolaidos koeficientas, lemiantis būsimo atlygio svarbą lyginant su tiesioginiais atlygiais. $\gamma Q\left(s', a = \pi\left(s'\right); \theta\right)$ – tai yra Q funkcija, kuri įvertina numatomą atlygį už veiksmą (a) esant s0 būsenai ir po to, kai laikomasi dabartinės politikos π. Q funkcija parametrizuojama θ. Apibendrinant, ši formulė naudojama Q funkcijai atnaujinti iki tikslinės vertės, kuri yra atlygio, gauto esant dabartinės būsenos ir veiksmo porai (r), ir numatomo atlygio už veiksmą (a) kitoje būsenoje ir dabartinės politikos π laikymąsi suma.

Privalumai:

Dokumente pateikiama sustiprinto mokymosi analizė, apimanti įvairius modelius, algoritmus ir metodus – kas padeda suprasti švelnų kritiko ir aktoriaus aspektą. Aptariamas veikėjų-kritikų architektūros naudojimas kelių agentų sistemose. Jame pabrėžiami centralizuoto kritiko naudojimo mokymosi metu ir decentralizuotų veikėjų, kuriuos galima panaudoti savarankiškai, pranašumai. Šis metodas leidžia efektyviai mokytis ir koordinuoti kelių agentų veiklą. Taip pat minimas pakartojimo atminties naudojimas ir kelių žingsnių grąža aktorių kritikų architektūrose. Šie metodai prisideda prie mokymosi proceso atrankos ir skaičiavimo efektyvumo, todėl jie tampa efektyvesni ir greitesni.

Trūkumai:

Dokumente nepateikiami konkretūs pavyzdžiai ar atvejų studijos (angl. case studies), rodančios praktinį švelnaus kritiko ir veikėjo metodo taikymą realaus pasaulio scenarijuose. Nors dokumente trumpai minimas tikslo funkcijos modifikavimas giliuose SM algoritmuose, jame nesigilinama į detales ir nepateikiama išsami skirtingų požiūrių analizė.

1.3.8. Actor-Critic Nr. 2

Straipsnyje [LWT+20] nagrinėjami iššūkiai ir sprendimai, susiję su mokymosi iš kelių agentų sustiprinimo (MARL), daugiausia dėmesio skiriant mišrioms kooperatyvinėms ir konkurencinėms aplinkoms. Aptariami tradicinių politikos gradiento metodų apribojimai paprastų kelių agentų scenarijuose ir pristatomas naujas algoritmas „Multi-Agent Actor-Critic“, skirtas šiems iššūkiams spręsti. Algoritmu siekiama leisti agentams išmokti efektyvios politikos sudėtingose aplinkose, kur agentai turi bendradarbiauti, konkuruoti ir strategiškai bendrauti. Pabrėžiama, kaip svarbu kurti algoritmus, kurie galėtų veiksmingai valdyti kelių agentų sąveiką. Tradiciniai politikos gradiento metodai dažnai susiduria su daugelio agentų nustatymais ir parametrais dėl tokių problemų kaip aplinkos ne stacionarumas ir nuoseklių gradiento signalų trūkumas.

„Multi-Agent Actor-Critic“ algoritmas sprendžia šiuos iššūkius, leisdamas agentams išmokti centralizuotų kritikų, kartu išlaikant decentralizuotą vykdymą, todėl jie gali naudoti papildomą informaciją treniruočių metu nepasikliaujant ja testavimo metu. Vienas iš svarbiausių šio straipsnio indėlių yra kelių agentų veikėjo-kritiko algoritmo, kuris pagerina agentų koordinavimą ir našumą mišrioje kooperatyvinėje ir konkurencinėje aplinkoje, įdiegimas. Algoritmas pasitelkia aktoriaus-kritiko metodus ir į kritiką įtraukia papildomos informacijos apie kitų agentų politiką, o aktorius turi prieigą tik prie vietinės informacijos. Šis metodas leidžia agentams išmokti veiksmingą politiką, kurioje atsižvelgiama į kitų agentų veiksmus, o tai pagerina našumą sudėtingų kelių agentų scenarijuose. Straipsnyje ne tik pristatomas kelių agentų veikėjo ir kritikos algoritmas, bet ir aptariamas susijęs darbas MARL srityje. Siūlomas metodas lyginamas su esamais metodais, pvz., politikos gradiento metodais su centralizuotu kritiku ir pasikartojančia politika su grįžtamaisiais kritikais. Taip pat pripažįstami iššūkiai, su kuriais susiduria tradiciniai sustiprinimo mokymosi metodai, kai naudojami keli agentai, ir pabrėžiama, kad reikia algoritmų, kurie galėtų veiksmingai valdyti ir kooperacinę, ir konkurencinę sąveiką. Dokumente pateikti eksperimentiniai rezultatai parodo Multi-Agent Actor-Critic algoritmo efektyvumą atliekant įvairias užduotis. Užduočių pavyzdžiai yra tokie, kaip: bendravimas, bendradarbiavimas naviguojant ir fizinė apgaulė. Bendradarbiavimo užduotyje agentai turi efektyviai bendrauti, kad pasiektų bendrą tikslą, o fizinės apgaulės užduotyje agentai yra mokomi apgauti kitus agentus koordinuotu elgesiu. Šaltinyje plačiau buvo analizuojamas politikos gradiento algoritmas, kuriuo siekiama maksimaliai padidinti tikslo funkciją $J(\theta)$, koreguodami politikos parametrus θ, didinant laukiamą grąžą $E[R]$. Gradiento skaičiavimas, pagrįstas politikos gradiento teorema, apima $\nabla\theta J(\theta)$ skaičiavimą, atsižvelgiant į lūkesčius, susijusius su politikos būsenomis ir veiksmais, įtraukiant logaritminį tikimybės gradientą ir veiksmo vertės funkciją $Q\pi(s, a)$. Būsenų pasiskirstymas $p\pi$ yra labai svarbus šiame procese, atspindintis būsenų tikimybių pasiskirstymą pagal politiką. Taipogi, veiksmo vertės funkcijos $Q\pi$ įvertinimas skiriasi įvairiuose algoritmuose: nuo pavyzdinių grąžų naudojimo iki tikrosios funkcijos apytikslio nustatymo naudojant tokius metodus kaip laiko skirtumo

mokymasis. Tačiau didelė gradiento įverčių dispersija, ypač kelių agentų nustatymuose, kelia iššūkių dėl atlygio kintamumo, kuriam įtakos turi agentų sąveika. Bazinės linijos, kaip ir reikšmių funkcijos bazinės linijos, yra naudojamos dispersijai sumažinti, tačiau jų veiksmingumui kelių agentų aplinkoje trukdo ne stacionarumas. Be to, autorių pasiūlymas pabrėžia teigiamų gradiento įverčio rezultatų mažėjančią tikimybę didėjant veiksnių skaičiui, kaip išsamiai yra aprašyta priede, pateiktame įrodyme.

Rezultatai rodo, kad siūlomas algoritmas šiose užduotyse pranoksta tradicinius SM metodus, pabrėžiant jo gebėjimą valdyti sudėtingas kelių agentų sąveikas. Apskritai, pateikiamos vertingos įžvalgos apie mokymosi iš kelių agentų stiprinimo iššūkius mišrioje kooperatyvinėje ir konkurencinėje aplinkoje ir pristatomas naujas algoritmas „Multi-Agent Actor-Critic“, skirtas šiems iššūkiams spręsti. Leisdamas agentams išmokti centralizuotų kritikų ir palaikyti decentralizuotą vykdymą, algoritmas leidžia efektyviai koordinuoti ir palaikyti ryšį tarp agentų, todėl pagerėja įvairių kelių agentų užduočių našumas. Eksperimentiniai rezultatai patvirtina siūlomo algoritmo veiksmingumą tvarkant sudėtingas kelių agentų sąveikas ir pabrėžia jo taikymo galimybes įvairiose srityse.

Privalumai:

Straipsnyje pabrėžiamas siūlomo požiūrio stiprybės, lyginant su jau esamais metodais tiek bendradarbiavimo, tiek konkurencijos scenarijuose, atvaizduojant agentų populiacijų gebėjimą atrasti sudėtingas fizines ir komunikacines koordinavimo strategijas. Be to, pristatomas mokymo režimas, pagal kurį kiekvienam agentui naudojamas įvairių strategijų rinkinys, o tai leidžia sukurti tvirtesnę kelių agentų politiką.

Trūkumai:

Aptariamas aplinkos nestacionarumas dėl besikeičiančios atskirų agentų politikos mokymo metu, o tai gali kelti mokymosi stabilumo iššūkių ir trukdyti tiesiogiai panaudoti ankstesnės patirties atkūrimą. Be to, minima didelė dispersija, kurią gali turėti politikos gradiento metodai, kai reikalingas kelių agentų koordinavimas - o tai rodo galimus sunkumus efektyviai mokyti agentus sudėtinguose kelių agentų scenarijuose. Šie iššūkiai pabrėžia apribojimus, su kuriais susiduriama stiprinant mokymąsi aplinkoje, kurioje yra daug agentų, ir pabrėžia naujoviškų metodų poreikį šioms problemoms spręsti.

1.3.9. Actor-Critic Nr. 3

Šiame straipsnyje [PWY+21] gilinamasi į Soft Actor-Critic (SAC) algoritmo taikymą kelių agentų sistemų kontekste, ypač daug dėmesio skiriant bendradarbiavimo užduotims. SAC algoritmas yra gilaus sustiprinimo mokymosi metodas, kuris parodė didelę sėkmę įvairiose sudėtingose aplinkose. Šiame darbe autoriai siūlo naują metodą, vadinamą Decomposed Soft Actor Critic (mSAC), kuriuo siekiama sujungti kelių agentų vertės funkcijų skaidymo ir kelių agentų politikos gradientų pranašumus. SAC algoritmas, kuris yra mSAC metodo pagrindas, taip pat trumpai yra pristatomas šiame šaltinyje. Pabrėžiama, kad SAC algoritmas yra veiksmingas sprendžiant kredito priskyrimo problemą vieno agento scenarijuose. Algoritmas įtraukia

agentą, sąveikaujantį su aplinka Markovo sprendimų procese (MDP), kur jis stebi būsenas, imasi veiksmų, gauna atlygį ir pereina į kitą būseną pagal politiką. MSAC metodas apima kelis pagrindinius komponentus, pagerinant jo veikimą kelių agentų nustatymuose. Vienas iš esminių aspektų - išskaidyta Q tinklo architektūra, kuri apima jungtinės Q vertės suskaidymą į atskiras kiekvieno agento Q vertes. Šis išskaidymas leidžia efektyviau įvertinti vertės funkciją ir pagerinti agentų koordinavimą. Be to, metodas naudoja decentralizuotą tikimybių politiką, leidžiančią kiekvienam agentui turėti savo politiką, taip pagerinant bendrą kelių agentų sistemų koordinavimą. Kitas svarbus mSAC metodo komponentas yra priešingos padėties pranašumo funkcija. Ši funkcija atlieka gyvybiškai svarbų vaidmenį sprendžiant kreditų priskyrimo problemą kelių agentų sistemose, ypač scenarijuose su atskiromis ir nuolatinėmis veiksmų erdvėmis. Įtraukus priešingos padėties pranašumo funkciją, mSAC metodu siekiama efektyvesnio mokymosi už politikos ribų ir pagerinti bendrą agentų našumą atliekant bendradarbiavimo užduotis.

Šaltinyje pateikiamos išsamios įžvalgos apie matematines formules ir lygtis, naudojamas mSAC metodu. Jame apibūdinama mSAC metodo kritinė praradimo funkcija kelių agentų aplinkoje, pabrėžiant minkštųjų Q vertės tinklų ir tikslinių verčių svarbą mokymosi procese. Viena iš šių formulių paprastai vadinama tikslo funkcija arba efektyvumo metrika sustiprintame mokyme, pritaikius algoritmą – tokiame kaip „Soft Actor-Critic“ (SAC). Ši formulė parodo tikėtiną grąžą arba kaupiamąjį atlygį, kurį agentas gali pasiekti laikydamasis konkrečios politikos π, apimančios ir tiesioginį gautą atlygį, ir entropijos įteisinimo terminą, norint, kad mokymosi proceso metu būtų galima subalansuoti tyrinėjimą ir išnaudojimą.

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r\left(s_t, a_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right) \right] \tag{8}$$

- $J(\pi)$ nurodo tikslo funkciją, kurią siekiame optimizuoti. Sustiprintame mokyme, ši funkcija paprastai parodo laukiamą grąžą arba kaupiamąjį atlygį, gautą laikantis politikos π.
- T nurodo laiko horizontą arba laiko žingsnių skaičių, į kurį atsižvelgiama sumuojant.
- t=0 nustato pradinį sumavimo laiko žingsnį.
- $E_{(s_t,a_t)\sim\rho_\pi}$ žymi būsenų s_t ir veiksmų a_t lūkesčių operatorių, atrinktų iš būsenos-veiksmo skirstinio, kurį sukelia politika π. Tai reiškia, kad mes apskaičiuojame numatomą išraiškos reikšmę, atsižvelgiant į būsenos ir veiksmų poras, atrinktas iš politikos π.
- $[r\left(s_t, \mathbf{a}_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right)]$ nurodo išraišką laukimo operatoriuje. Čia: $r(s_t, a_t)$ yra tiesioginis atlygis, gautas imantis veiksmų a_t būsenoje s_t. O α hiperparametras valdo kompromisą tarp politikos entropijos padidinimo ir laukiamos grąžos.
- $\mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right)$ - nurodo politikos π entropiją, sąlygojamą būsenos s_t. Entropija matuoja politikos veiksmų pasiskirstymo neapibrėžtumą arba atsitiktinumą. Entropijos padidinimas gali paskatinti tyrinėjimą mokymosi procese.

Be to, šaltinyje pateikti eksperimentiniai rezultatai parodo mSAC metodo efektyvumą

atliekant įvairias bendradarbiaujančias kelių agentų užduotis, ypač StarCraft II mikrovaldymo scenarijuose. Metodas pranoksta esamus politika pagrįstus algoritmus, tokius kaip COMA, ir pasiekia panašų našumą su verte pagrįstais metodais, tokiais kaip Qmix. Rezultatai pabrėžia mSAC metodo tvirtumą ir efektyvumą sprendžiant iššūkius kelių agentų sistemose.

Apibendrinant galima pasakyti, kad išskaidytas minkštųjų veikėjų kritiškumo metodas, skirtas bendradarbiaujančiam kelių agentų stiprinimo mokymuisi, pristato naują metodą, kuris sujungia minkštųjų veikėjų kritikos algoritmų principus su vertės funkcijų skaidymu kelių agentų nustatymuose. Naudodamas decentralizuotą politiką, priešingos padėties pranašumų funkcijas ir išskaidytą Q tinklo architektūrą, mSAC metodas rodo daug žadančius rezultatus gerinant agentų našumą atliekant bendradarbiavimo užduotis. Šaltinyje pateikiama išsami metodo komponentų ir eksperimentinių vertinimų apžvalga, parodomas jo potencialas tobulinant kelių agentų sustiprinimo mokymosi tyrimus.

1.3.10. Actor-Critic Nr. 4

Šiame šaltinyje [GVC+21] pristatomas pažangus NPC navigacijos metodas, naudojant Soft Actor-Critic (SAC) algoritmą. Tyrime nagrinėjami tradicinių navigacijos metodų, pvz., „Waypoint“ grafikų ir „NavMesh“, apribojimai, pasitelkiant gilų mokymąsi, norint užtikrinti lankstesnį ir labiau prisitaikantį navigacijos elgesį. Tradiciniai metodai, tokie kaip maršruto taško grafikai, apima rankiniu būdu sukonstruotus navigacijos taškus, kurie yra linkę į klaidas ir yra nelankstūs, ypač dinamiškoje aplinkoje. „NavMesh“, nors ir patobulinta, vis dar susiduria su mastelio keitimu ir sudėtingų navigacijos įgūdžių integravimu. Mokymasis, ypač be modelių, pvz., SAC, yra daug žadanti alternatyva mokantis optimalaus navigacijos elgesio tiesiogiai sąveikaujant su aplinka.

Tyrime naudojamas Unity žaidimo variklis 3D aplinkai su įvairiomis kliūtimis imituoti, kurioje NPC naršo aplinkoje naudodami dviejų lygių valdymo sistemą:

1. Aukšto lygio planuotojas – atsakingas už bendrą kelio planavimą ir kelio taško nustatymą.
2. Žemo lygio valdiklis – naudoja SAC algoritmą, atliekant navigacijos veiksmus tarp kelio taškų, užtikrinant reguliavimą realiuoju laiku ir kliūčių išvengimą.

Būsenos ir veiksmo atstovavimas:

1. Būsenos erdvė apima NPC padėtį, orientaciją, jutiklio duomenis (pvz., atstumus iki kliūčių) ir būsenos informaciją (pvz., šokinėjimą ar kontaktą su žeme).
2. Veiksmo erdvė apima judėjimo komandas, pvz., pirmyn, atgal, sukimąsi ir šokinėjimą.

SAC algoritmas sujungia aktoriaus kritikos metodus su maksimaliu entropijos stiprinimo mokymusi. Šiuo metodu siekiama maksimaliai padidinti laukiamą atlygį ir politikos entropiją, skatinant tyrinėjimą ir užkertant kelią ankstyviai konvergencijai prie neoptimalios politikos. SAC algoritmas naudoja atskirus tinklus, skirtus veikėjui (politikai) ir kritikui (vertės funkcija), o kaitalioja politikos vertinimą (kritiko atnaujinimas) ir politikos tobulinimą (aktoriaus

atnaujinimą), naudodamas pakartojimo buferį, stabilizuojant mokymą. NPC yra mokomi taikant mokymosi programos metodą, kuris prasideda nuo paprastesnių užduočių ir palaipsniui sunkumas yra didinamas. Atlygio funkcija yra skirta skatinti į tikslą nukreiptą elgesį, tuo pačiu bausdama už susidūrimus ir neefektyvius kelius. Taip pat, mokymas apima nuolatinį politikos atnaujinimą, pagrįstą surinkta patirtimi, panaudojant SAC algoritmo galimybes, norint veiksmingai suderinti tyrinėjimą ir naudojimą. Eksperimentai apėmė NPC mokymą įvairiose 3D žaidimų scenose, kurios tampa vis sudėtingesnės. Našumo rodikliai apėmė navigacijos efektyvumą, kliūčių išvengimo sėkmės rodiklį ir bendrą kelio optimalumą. Be to, SAC pagrįstas valdiklis buvo lyginamas su kitais sustiprinimo mokymosi algoritmais (pvz., DDPG, PPO) ir tradiciniais navigacijos metodais.

Privalumai:

SAC algoritmas parodė puikų našumą navigacijos efektyvumo požiūriu, todėl maršrutai buvo sklandesni ir efektyvesni. Algoritmu valdomi NPC sėkmingiau išvengė kliūčių, palyginti su tais, kurie naudoja tradicinius metodus ar kitus sustiprinimo mokymosi algoritmus. Taip pat SAC pagrindu veikiančio valdiklio sugeneruoti keliai buvo optimalūs, o tai rodo algoritmo efektyvumą atliekant realiojo laiko navigacijos koregavimus.

Galimi tobulinimai:

Hierarchinės sustiprinimo mokymosi architektūros integravimas, leidžiančias labiau struktūrizuoti sprendimų priėmimo procesus. Pavyzdžiui, aukšto lygio politika gali valdyti strateginius sprendimus, o žemo lygio politika – konkrečius veiksmus. Taip pat būtų galima įtraukti kelių tikslų atlygio funkcijas, kurios subalansuoja įvairius navigacijos užduoties aspektus, tokius kaip greitis, saugumas ir efektyvumas. Tai galima pasiekti dinamiškai pasveriant skirtingus atlygio funkcijos komponentus treniruotės metu.

1.4. Pagalbinis metodas

Šiame skyriuje yra nagrinėjama prižiūrimo mokymosi elgsenos klonavimo technika. Ši technika gali būti naudojama norint apmokyti agentus iš anksto, kuris vėliau bus naudojamas sustiprintame mokyme. Tai galima laikyti kaip atspirties tašku, kuris pagreitintų agento mokymąsi.

1.4.1. Behavioural Cloning

Elgesio klonavimas (BC) yra mašininio mokymosi technika, kai kompiuteris mokosi atlikti užduotį pagrįstomis demonstracijomis ir šis metodas yra taikomas įvairiose srityse, įskaitant vaizdo žaidimus ir robotiką. BC yra galutinis metodas, kai sistema mokosi iš neapdorotų įvesties (pvz., žaidimo ekrano vaizdų) ir išveda veiksmus (pvz., klavišų paspaudimus) be išankstinio įvesties funkcijų apdorojimo. Šiame darbe [KPH20] vertinamas bendras BC pritaikymas keliuose vaizdo žaidimuose, siekiant suprasti jo efektyvumą, duomenų reikalavimus ir duomenų kokybės poveikį.

Elgesio klonavimas veikia imitacinio mokymosi principu. Tai apima modelio mokymą numatyti veiksmus, pagrįstus stebimomis būsenomis, iš esmės traktuojant jį kaip prižiūrimą mokymosi problemą, kai modelis išmoksta susieti būsenas (pvz., žaidimo vaizdus) ir veiksmus (pvz., klavišų paspaudimus). Apmokytas modelis gali imituoti žmogaus žaidimą, numatydamas veiksmus iš žaidimo būsenų – tikslas yra pasiekti žmogaus lygio našumą be didelių žaidimui būdingų modifikacijų ar ilgesnių treniruočių laiko, būdingų sustiprinimo mokymuisi (SM).

Autoriai atliko eksperimentus naudodami žmogaus žaidimo duomenis, norint išmokyti gilius neuroninius tinklus žaisti įvairius vaizdo žaidimus. Tyrime buvo dvylika žaidimų, įskaitant klasikinius „Atari 2600“ žaidimus ir modernius vaizdo žaidimus, išleistus po 2010 m. Treniruočių duomenis sudarė įrašytas žmogaus žaidimo būdas, o modeliai buvo įvertinti pagal jų gebėjimą atkartoti žmogaus veiklą. Pagrindiniai sąrankos komponentai apima:

1. Žmonių žaidėjų žaidimo eiga buvo įrašyta, fiksuojant ir žaidimo būsenas (ekrano vaizdus), ir atitinkamus veiksmus (klavišų paspaudimus). Surinktų duomenų kokybė ir kiekis skyrėsi, kad būtų galima įvertinti jų poveikį modelio veikimui.
2. Modeliuose buvo naudojami konvoliuciniai neuroniniai tinklai (CNN) žaidimo vaizdams apdoroti ir veiksmams numatyti, norint išlaikyti skaičiavimo efektyvumą ir išlaikyti pakankamai paprastą architektūrą.
3. Modeliai buvo mokomi taip, kad būtų sumažintas kryžminės entropijos praradimas tarp numatytų ir faktinių veiksmų. Vertinimas buvo atliktas leidžiant apmokytiems modeliams žaisti žaidimus ir lyginant jų pasirodymą su žmonėmis.

Remiantis šiais sąrankos komponentais, BC agentai išmoko pagrindinės žaidimo dinamikos, bet paprastai neprilygo žmogaus rezultatams. Tačiau kai kuriuose žaidimuose, pvz., Q*bert ir Space Invaders, agentai pasiekė apie 20 % žmogaus sugebėjimų. Taip pat aukštesnės kokybės duomenys (iš geriausių žaidėjų) žymiai pagerino modelio našumą, palyginti su didesniais prastesnės kokybės duomenų kiekiais. Pavyzdžiui, naudojant tik 5 % geriausių žmonių žaidimo duomenų, našumas buvo geresnis nei naudojant visus turimus duomenis. Įtraukus veiksmų delsą (atsižvelgiant į žmogaus refleksus), pagerėjo modelio veikimas. Dviejų ar penkių kadrų delsimas davė geresnių rezultatų, ypač tokiuose žaidimuose kaip Q*bert ir Space Invaders.

Tyrimas rodo, kad elgesio klonavimas gali veiksmingai išmokyti modelius pagrindinių žaidimo mechanikų ir taisyklių, nors pasiekti žmogaus lygio našumą išlieka sudėtinga. Mokymo duomenų kokybė vaidina lemiamą vaidmenį BC sėkmei, dažnai nei duomenų kiekis. Žmogaus veiksmų vėlavimų įtraukimas į mokymo procesą taip pat pagerina modelio veikimą. Išvados rodo, kad nors vien BC sudėtinguose žaidimuose gali nepasiekti žmogaus lygio žaidimo, jis yra vertingas atspirties taškas tolesniems mokymosi metodams arba kaip išsamesnių AI sistemų komponentas. Tyrimas padeda suprasti BC potencialą ir apribojimus vaizdo žaidimų programose, atverdamas kelią labiau apibendrintam mokymosi imitaciniam metodui.

Bendrojo elgesio klonavimo trūkumai:

Šiuolaikiniai žaidimai dažnai veikia didele raiška, todėl neapdorotų vaizdų apdorojimas

yra brangus, o sumažinus vaizdus gali būti prarasta svarbi informacija. Vaizdo žaidimuose paprastai yra sudėtingos veiksmo erdvės, apimančios kelis klavišus ir pelės judesius, todėl supaprastinti šias veiksmų erdves išlaikant žaidimo galimybes yra sudėtinga. Taip pat žmonių žaidėjai ir žaidimų aplinka veikia asinchroniškai, todėl atsakymai į veiksmus vėluoja. Šis neatitikimas gali neigiamai paveikti treniruočių duomenų kokybę.

1.5. Aktyvacijos funkcija

1.5.1. RMAF: ReluMemristor Like Activation

Šiame šaltinyje [YAT+20]aptariamas RMAF „ReLU-Memristor-Like“ aktyvinimo funkcija ir galimi jos pranašumai, palyginti su dažniausiai naudojama ReLU aktyvinimo funkcija giliame mokyme. ReLU aktyvinimo funkcija turi tam tikrų apribojimų, pvz., „mirštantis ReLU“ problema, kai neigiamų įėjimų gradientas tampa nuliu, todėl neuronas veiksmingai „miršta“ ir neprisideda prie tinklo išvesties. RMAF pristato pastovų parametrą (α) ir slenkstinį parametrą (p), kad funkcija būtų sklandi, nemonotoniška ir tinkle būtų įvestas netiesiškumas (angl. non-linearity). RMAF išnaudoja neigiamų verčių neuroniniuose tinkluose privalumus ir tuo buvo įrodyta, kad jis pranoksta ReLU ir kitas aktyvinimo funkcijas gilesniuose modeliuose ir daugelyje sudėtingų duomenų rinkinių. RMAF formulė:

$$L_t^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{9}$$

Formulės parametrai: x - parametras nurodo aktyvinimo funkcijos įvesties reikšmę ; j - parametras yra pastovus hiperparametras, padauginamas iš aktyvinimo funkcijos išvesties; t - parametras yra konstanta, naudojama funkcijos statumui (angl. steepness) reguliuoti ; p - parametras yra slenksčio parametras, valdantis srities lygumą. Jis naudojamas, kad funkcija būtų sklandi, nemonotoniška ir galėtų būti keičiama bet kuriame tinkle.

Autoriai pateikia eksperimentinius rezultatus, lyginančius ReLU, Tanh, ir RMAF įvairiuose duomenų rinkiniuose ir neuroninių tinklų architektūrose. Rezultatai rodo, kad RMAF pasiekia didesnį mokymo ir patvirtinimo tikslumą, palyginti su ReLU ir Tanh CIFAR100, Resnet50, Alexnet ir kituose duomenų rinkiniuose. Taip pat aptariamas DenseNets, neuroninio tinklo architektūros tipo, turinčio keletą pranašumų, palyginti su tradicinėmis architektūromis, projektavimas ir įgyvendinimas. „DenseNets“ sprendžia nykstančio gradiento problemą, sustiprina savybių plitimą, skatina pakartotinį funkcijų naudojimą ir žymiai sumažina parametrų skaičių. Atsižvelgiant į mokymosi nustatymus, autoriai naudojo „Keras“ gilaus mokymosi bibliotekas su „TensorFlow“ programa, kad galėtų įdiegti ir įvertinti aktyvinimo funkciją. Jie naudojo 128 partijos dydį (angl. batch size), 0,1 mokymosi greitį ir Adam optimizavimo priemonę kategoriškam entropijos praradimui. Taip pat buvo pastebėta, kad RMAF turi panašią savybę kaip „Swish“, kurią pasiūlė „Google Brain“. Palyginimas rodo (3 figūra, c paveikslėlis šaltinyje), kad Swish pranoksta ReLU keliuose giluminiuose modeliuose vaizdų klasifikavimo ir

mašininio vertimo problemų atžvilgiu. Tačiau „Swish“ išvestinės medžiaga susideda iš didelės negausios savybės (angl. nonsparse property), o tai padidina tinklo skaičiavimo sudėtingumą. Kita vertus, RMAF gali išlaikyti savo kietojo nulio savybę (angl. hard zero property), dėl kurios ReLU dažnai deaktyvuoja daugumą neuronų sklidimo į priekį ir atgal metu. RMAF turi pranašumų prieš ReLU, nes išlaiko kietojo nulio savybę ir išvengia neuronų deaktyvavimo. Tačiau keliuose giluminiuose modeliuose „Swish“ vis dar lenkia RMAF ir ReLU, nors jo skaičiavimo sąnaudos yra didesnės. Apskritai RMAF aktyvinimo funkcija yra perspektyvi ReLU alternatyva, pašalinanti kai kuriuos jos apribojimus ir pagerinanti įvairių duomenų rinkinių našumą. Autoriai pateikia išsamius eksperimentinius rezultatus ir metrikas pagrindžiant savo teiginius ir įrodant RMAF efektyvumą.

Privalumai:

RMAF įveda pastovų parametrą (α) ir slenkstinį parametrą (p), kad funkcija būtų sklandi, nemonotoniška ir palaikytų netiesiškumą tinkle. Autoriai įvertino siūlomos RMAF funkcijos efektyvumą septyniose duomenų bazėse, įskaitant Viskonsino krūties vėžį, MNIST, Iris, automobilių vertinimą, CIFAR-10, CIFAR-100 ir ImageNet. RMAF išnaudoja neigiamų reikšmių naudą neuroniniuose tinkluose ir buvo įrodyta, kad ReLU ir kitos aktyvinimo funkcijos nėra tokios efektyvios kaip RMAF modeliuose ir daugelyje sudėtingų duomenų rinkinių.

Trūkumai:

RMAF funkcija turi apribojimų artėjant prie nulio, kai didelės neigiamos įvestys yra neatkuriamos. Taip pat, RMAF funkcijai reikia modifikuoti parametrus j ir p, kad funkcija būtų panaši į ReLU ir galėtų keisti mastelį bet kuriame tinkle.

Galimas RMAF aktyvacijos funkcijos panaudojimas:

Eksperimentavimas su skirtingais hiperparametrais, tokias kaip - j, t ir p reikšmėmis, randant optimalias reikšmes, kurios geriausiai tinkų naudojamai neuroninio tinklo architektūrai.

1.6. Išanalizuotos literatūros apibendrinimas

Nuodugniai išnagrinėjus literatūrą, tikslas tobulinti ir taikyti sustiprintus mašininio mokymosi metodus kompiuteriniuose žaidimuose yra pasiekiamas. Esami tyrimai suteikia tvirtų žinių apie tokius algoritmus kaip Proksimalinės politikos optimizavimas (PPO) ir Soft Actor-Critic (SAC) žinių pagrindą. Šie metodai buvo veiksmingi įvairiose programose ir yra daug galimybių juos patobulinti. „Unity“ žaidimo variklis siūlo palankią aplinką šiems sustiprinto mašininio mokymosi metodams realizuoti ir išbandyti. Dabartinė literatūra palaiko galimybę sukurti patobulintus metodams, kurie galėtų efektyviau ir tiksliau išmokti sudėtingas užduotis, o tai rodo, kad šiame darbe išdėstyti tikslai yra realūs ir pasiekiami.

Literatūros apžvalgoje buvo nustatytos kelios esamų mašininio mokymosi metodų tobulinimo sritys. Padidinti PPO ir SAC stabilumą ir efektyvumą galima atitinkamai naudojant prisitaikančius kirpimo mechanizmus ir hiperparametrų derinimą. Šie patobulinimai gali padėti sukurti tvirtesnius ir efektyvesnius mokymosi procesus. Be to, integruojant hibridinius

metodus, pavyzdžiui, derinant entropijos reguliavimą su smalsumu grindžiamu tyrinėjimu, galima dar labiau pagerinti mokymosi efektyvumą ir tvirtumą. Neuroninių tinklų architektūros koregavimai taip pat suteikia daug galimybių tobulėti. Padidinus neuronų skaičių paslėptuose sluoksniuose, pakeitus aktyvinimo funkcijas į efektyvesnes, tokias kaip ReLU, ir kruopščiai subalansavus neuroninių tinklų gylį, galima optimizuoti mokymosi pajėgumus ir sušvelninti tokias problemas kaip per didelis mokymasis ir nykstantys gradientai. Be to, įtraukus pagalbines užduotis ir panaudojus lygiagrečias mokymo aplinkas, galima pagerinti šių algoritmų našumą ir mastelį.

Tačiau, remiantis dabartine literatūra ir šio baigiamojo darbo tikslais, tam tikrų mašininio mokymosi metodų tolesniuose etapuose nėra verta tyrinėti. Konkrečiai, generatyvus priešiškas mokymasis (GAIL) ir elgesio klonavimo (BC) technika nebus įtraukti į tolesnius tyrimus. Šie metodai, nors ir vertingi, neparodo tokio paties veiksmingumo ir universalumo kaip PPO ir SAC šio tyrimo kontekste. GAIL ir BC susiduria su apribojimais, tokiais kaip treniruočių nestabilumu, pasitikėjimu aukštos kokybės ekspertų duomenimis ir neprisitaikymo skirtingose (panašiose) aplinkose.

Dėmesys į PPO ir SAC, kurie yra geriau palaikomi ir pasirodė esantys veiksmingesni sudėtingoje aplinkoje, leis nuodugniau ištirti šiuos metodus ir galimus jų patobulinimus. Šie metodai supaprastins tyrimų procesą ir leis daugiau prisidėti prie žaidimų DI srityje.

Apibendrinant galima teigti, kad literatūra patvirtina šio darbo tikslų pasiekiamumą, išryškina aiškius esamų mašininio mokymosi metodų tobulinimo kelius. Sutelkiant dėmesį į pažangius metodų patobulinimus ir sudėtingas modeliavimo aplinkas, šis tyrimas gali reikšmingai prisidėti prie pažangių žaidimų agentų kūrimo.

2. Technologijos ir testavimo aplinka

Šiame skyriuje yra pateikiama darbe naudojamų technologinių ir testavimo aplinkos aprašymas. Pažangių mašininio mokymosi sistemų integravimas su Unity žaidimo varikliu sudaro šio darbo pagrindą, ypač mokant ir vertinant sustiprinimo mokymosi (SM) agentus. Pagrindinės naudojamos technologijos yra „Unity“, „Unity ML-Agents Toolkit“, „Python API“ ir „TensorFlowSharp“, kurių kiekviena atlieka svarbų vaidmenį palengvinant mokymo procesą.

2.1. Unity Game Engine

Unity yra plačiai naudojama realaus laiko 3D kūrimo platforma, žinoma dėl savo universalumo kuriant interaktyvią aplinką. Dėl gebėjimo imituoti labai dinamišką, vizualiai turtingą ir interaktyvią aplinką ji išpopuliarėjo atliekant mokslinius tyrimus, susijusius su dirbtiniu intelektu (AI) ir mokymosi stiprinimu. Dėl „Unity“ lankstumo kuriant sukurtus scenarijus, ši platforma yra naudinga kuriant eksperimentines sąrankas, kuriose agentai sąveikauja su aplinka ir mokosi iš savo patirties. Visų pirma, Unity šiame tyrime naudojama kuriant procedūriškai sukurtą aplinką, pvz., kliūčių ruožus, kurie padeda agentams ugdyti problemų sprendimo įgūdžius ir apibendrinti savo mokymąsi atliekant įvairias užduotis. Antra, „Unity“ realaus laiko fizikos variklis leidžia modeliuoti realią agentų ir jų aplinkos sąveiką, o tai būtina kuriant SM agentus, kurie gali atlikti užduotis, kurioms reikia fizinio manipuliavimo, orientavimosi ir sprendimų priėmimo esant neapibrėžtumui. Be to, Unity palaikymas kelių platformų kūrimui užtikrina, kad modeliavimas gali būti efektyviai vykdomas įvairiose sistemose, palengvinant paskirstytą ir lygiagretų modeliavimą.

2.2. Unity ML-Agents Toolkit

„Unity ML-Agents Toolkit“ arba „ML-Agents“ yra atvirojo kodo sistema, kuri pagerina „Unity for AI“ tyrimą, suteikdama reikalingus įrankius agentams mokyti naudojant mašininio mokymosi metodus, pvz., gilųjį mokymąsi. „ML-Agents“ sklandžiai integruojasi su „Unity“, leisdamas tyrėjams apibrėžti pasirinktines SM aplinkas, nurodyti stebėjimo ir veiksmų erdves bei įgyvendinti atlygio struktūras, kurios vadovaujasi agento elgesiu. Šis įrankių rinkinys yra ypač naudingas, nes palaiko įvairius mašininio mokymosi metodus, įskaitant Proximal Policy Optimization (PPO) ir Soft Actor-Critic (SAC), kurie abu yra naudojami šiame tyrime.

Vienas iš svarbiausių Unity ML-Agents Toolkit pranašumų yra jo Python API, kuris palengvina ryšį tarp Unity ir išorinių mašininio mokymosi bibliotekų, tokių kaip TensorFlow. Šis API leidžia sukurti mokymo aplinką, kurioje agentai mokosi sąveikaudami su imituojama aplinka. Stebėjimo erdvės (ką agentas gali suvokti), veiksmų erdvės (galimi veiksmai, kuriuos agentas gali atlikti) ir atlygio funkcijos (atsiliepimai, kuriuos agentas gauna pagal savo veiksmus) yra pritaikomos, todėl įrankių rinkinys yra geras pasirinkimas sudėtingoms SM užduotims atlikti.

Galimybė lygiagrečiai paleisti kelias aplinkas žymiai pagreitina mokymo procesą, padidina imties efektyvumą ir leidžia agentams veiksmingiau mokytis iš įvairios patirties. Šis lygiagretinimas yra labai svarbus siekiant didelių skaičiavimo reikalavimų, susijusių su mokymosi pastiprinimu, kai agentai turi ištirti dideles būsenos veiksmų erdves optimizuojant savo našumą.

2.3. Python API

Unity ML-Agents Toolkit su Python API yra neatsiejama mokymo ir testavimo proceso dalis, kuri leidžia tyrėjams programiškai susieti su Unity aplinka. Per šį API, Python galima naudoti mokymo procedūroms apibrėžti, eksperimentams nustatyti ir agento veikimui stebėti. API leidžia kontroliuoti agento elgseną, rinkti duomenis ir atnaujinti agento politiką, atsižvelgiant į atlygį, gautą sąveikaujant su aplinka. Python API taip pat palengvina išorinių mašininio mokymosi sistemų integravimą, ypač „TensorFlow“, kuris naudojamas neuroniniams tinklams diegti politikos mokymuisi ir vertės funkcijų aproksimavimui. Šis lankstumas leidžia naudoti tinklinius giluminio mokymosi modelius, skirtus SM, pagerinant agento gebėjimą mokytis sudėtingų užduočių. API palaikymas paskirstytoms aplinkoms taip pat leidžia vienu metu mokyti kelis agentus, taip pagreitinant mokymosi procesą ir įgalinant didelio masto eksperimentus.

2.4. TensorFlowSharp integracija

Norint supaprastinti mašininio mokymosi modelių vykdymo procesą Unity aplinkoje, naudojama TensorFlowSharp. TensorFlowSharp yra .NET susiejimas su TensorFlow, leidžiantis TensorFlow modelius paleisti tiesiogiai su Unity, taip sumažinant priklausomybę nuo išorinių Python pagrįstų mašininio mokymosi sistemų darant išvadas. TensorFlowSharp leidžia sudaryti išvadas realiuoju laiku, o tai leidžia iš anksto paruoštus modelius vykdyti pačioje Unity aplinkoje, optimizuoja našumą ir sumažina delsą agento vertinimo metu. Naudojant TensorFlowSharp, šis darbas integruoja iš anksto paruoštus modelius į Unity modeliavimą, kurioje išmoktą politiką galima įvertinti realiuoju laiku. Šis metodas sumažina pačio proceso ryšio tarp Unity ir išorinių mašininio mokymosi struktūrų sąnaudas, todėl modeliavimas vyksta efektyviau, ypač esant aukšto dažnio sprendimų priėmimo scenarijams, pvz., kliūčių navigacijai.

2.5. Mokymo ir testavimo aplinka

SM agentų mokymo procesas šiame darbe vykdomas Unity variklyje, naudojant Unity ML-Agents Toolkit. Aplinkos projektavimo procedūrinis pobūdis užtikrina agentams susidurimą su įvairiais scenarijais, o tai padidina jų gebėjimą apibendrinti užduotis. Pavyzdžiui, kliūčių bokšto iššūkis (angl. Obstacle Tower), populiarus sustiprinimo mokymosi tyrimų būdas, naudojamas siekiant patikrinti agentų gebėjimus spręsti procedūriškai sugeneruotus galvosūkius, naršyti sudėtinguose labirintuose ir atlikti užduotis, kurioms reikia ilgalaikio planavimo.

Unity naudojama kuriant pritaikytas įvairaus sudėtingumo aplinkas. Šios aplinkos skirtos patikrinti SM agentų prisitaikymą, mokymosi gebėjimus ir problemų sprendimo gebėjimus. Įvairūs elementai, tokie kaip dinaminės kliūtys, atrakinamos durys ir procedūriškai sugeneruoti lygiai, sukuria sudėtingą agento veikimo testavimo sąranką. Taip pat, gautus rezultatus galima lyginti su kitų programuotojų gautais pasiekimais, kuriuos galima rasti viešai paskelbtoje svetainėje [AIc19]. Mokymosi sustiprinimo metu atlygis naudojamas siekiant sustiprinti pageidaujamą agento elgesį. Šiame darbe atlygio struktūra yra sukurta norint paskatinti agentus efektyviai atlikti užduotis, pvz., spręsti galvosūkius ar pasiekti konkrečius tikslus, tuo pačiu bausti už neveiksmingus ar neteisingus veiksmus. Tuo tarpu, Python API leidžia vienu metu pasileisti kelias aplinkas, todėl galima lygiagrečiai treniruotis. Ši sąranka žymiai sumažina laiką, reikalingą agentams apmokyti, nes jie gali vienu metu kaupti kelių modeliavimų patirtį. Šis metodas yra ypač naudingas stiprinant mokymąsi, kai agentai turi ištirti dideles būsenų erdves ir mokytis iš įvairių sąveikų. Išmokę agentai yra vertinami toje pačioje aplinkoje, siekiant užtikrinti, kad mokymasis būtų nuoseklus ir agentas galėtų apibendrinti savo įgūdžius naujoms užduotims atlikti. Kliūčių bokšto iššūkis suteikia išsamią bandymų eigą, kuri leidžia įvertinti agentus skirtingomis sąlygomis, įskaitant anksčiau nematytus scenarijus, norint įvertinti jų „suvokimą“ ir pritaikymą.

Šiame darbe agentai yra išbandomi ir apmokomi dviejose aplinkose – rankiniu būdu sukurtoje ne tokioje sudėtingoje aplinkoje (žr. 1 pav. prieduose) ir sudėtingoje aplinkoje ”Desert – Low Poly Toon Battle Arena / Tower Defense Pack”[Aur19] (žr. 2 pav. prieduose). Tai yra skirta nustatyti, kaip išmokinti agentai elgiasi kitose aplinkose, kuriose dar nėra buvę testuoti, kaip orientuojasi ir pasiekia nustatytus tikslus gaunant atlygius. Kadangi sustiprinto mokymosi literatūroje nėra vieningo sudėtingumo matavimo standarto, šiame darbe aplinkos sudėtingumas apibrėžiamas kaip sprendimų sudėtingumas, su kuriuo susiduria agentas siekdamas tikslo. Sudėtingumas siejamas su kliūčių bei baudų skaičiumi, kurie tiesiogiai lemia grįžtamojo ryšio išsibarstymą bei agento veikimo efektyvumą. Eksperimentų metu buvo pastebėta, kad esant mažiau nei 15 kliūčių, agentas greičiau pasiekia optimalų elgesį ir susiduria su mažiau baudų, todėl tokios aplinkos laikomos paprastesnėmis. Priešingai, aplinkose su daugiau nei 15 kliūčių, padidėja ir agento trajektorijų variacijų skaičius, tikslų pasiekimas tampa mažiau nuspėjamas, o mokymosi stabilumas sumažėja. Todėl šis kliūčių skaičius buvo pasirinktas kaip atskaitos taškas klasifikuojant aplinkas į paprastas ir sudėtingas. Šios aplinkos yra sukurtos taip, kurios keistų scenarijų su kiekviena iteracija, t.y., žaidimo agentas ir atlygis keistų savo startinę poziciją aplinkoje. Tokiu būdu agentas neišnaudoja aplinkos simuliacijos spragų ir išmoksta žaidimo logiką. Taip pat, išmokintas agentas galėtų būti perkeliamas „zero-shot“ režimu į kitą panašią žaidimo aplinką ir joje orientuotis, jeigu nauja aplinka yra panaši į šiame darbe sukurtas aplinkas ir žaidimo logika būtų ta pati – agentas turi rasti kelią iki atlygio.

3. Metodika

Šiame skyriuje aprašoma metodika, naudojama diegiant, mokant ir įvertinant sustiprinimo mokymosi (SM) agentus kontroliuojamoje modeliavimo aplinkoje. Metodika suskirstyta į tris dalis: eksperimentinė sąranka (angl. setup), Proksimalinės politikos optimizavimo (PPO) ir Soft Actor-Critic (SAC) metodų patobulinimai ir našumo metrikos, skirtos agentų mokymosi ir apibendrinimo gebėjimams matuoti. Pagrindinis tikslas yra optimizuoti agentų darbą dinamiškoje ir sudėtingoje aplinkoje, naudojant praeitame skyriuje paminėtas mokymo priemones (įrankius).

3.1. Eksperimentinė sąranka

Eksperimentinė sąranka sutelkta į agentų mokymą imituotoje aplinkoje, naudojant „Unity“ žaidimo variklį ir „Unity ML-Agents Toolkit“, suteikiant galimybę mokyti agentus „sustiprintai“. Aplinka ir skaičiavimo įrankiai atlieka esminį vaidmenį užtikrinant, kad agentai susidurtų su įvairiomis užduotimis, imituojančių realias sąlygas, rinkinį. Šiame tyrime naudojamos aplinkos yra sukurtos Unity, imituojant užduotis su nuolat kintančia dinamika. Dinaminė aplinka užtikrina, kad agentai negalėtų įsiminti aplinkos ir turi pasikliauti apibendrintomis strategijomis, kurios gali prisitaikyti prie naujų konfigūracijų.

Be „Unity“ variklio, sąrankoje naudojamos papildomos bibliotekos, įrankiai ir priemonės. „Unity ML-Agents Toolkit“ suteikia sąsają mašininio mokymosi algoritmams integruoti su modeliavimo aplinka. Tai palengvina sąveiką tarp mokymosi algoritmų ir virtualaus pasaulio valdymo, stebėjimo erdvės, veiksmų erdvės ir atlygio struktūros. Šis įrankių rinkinys jungiasi prie Python API, kuri kontroliuoja mokymosi algoritmų vykdymą ir palaiko ryšį tarp Unity aplinkos ir neuroninių tinklų, atsakingų už sprendimų priėmimą. „TensorFlow“ sistema, pasiekiama per „TensorFlowSharp“, naudojama kurti ir mokyti neuroninius tinklus, naudojamus tiek PPO, tiek SAC metoduose. Ši integracija su TensorFlowSharp leidžia Unity varikliui atlikti išvadas realiuoju laiku treniruočių metu, užtikrinant, kad agentai dinamiškai pritaikytų savo politiką modeliavimo metu. Visa sistema veikia naudojant didelio našumo skaičiavimo sąranką, kurią sudaro „NVIDIA RTX 2070 SUPER” vaizdo korta pagreitintam neuroninio tinklo skaičiavimui ir „Intel i5-12400F“, kuris turi 12 gijų, procesorius efektyviam lygiagrečiam modeliavimo užduočių apdorojimui.

„Unity“ aplinka yra sukonfigūruota teikiant agentams nuolatinį grįžtamąjį ryšį su atlygiais, atsižvelgiant į jų veiklą. Agentai gauna jutimo įvestį, pvz., padėties duomenis, orientaciją ir aplinkos sąveiką, kurią apdoroja neuroniniai tinklai, sprendžiant dėl tolimesnių veiksmų. Agentų sėkmė naršant procedūriškai sukurtoje aplinkoje matuojama pagal sukauptą atlygį, gautą per jų mokymo ciklus.

3.2. Metodų patobulinimai

Norint pasiekti optimalų našumą šiose aplinkose yra naudojami patobulinti standartiniai PPO ir SAC metodai. Šie patobulinimai yra skirti mokymosi stabilizavimui, tyrinėjimo gerinimui ir konvergencijos pagreitinimui, kuriose naudojamos matematinės formulės yra pritaikytos sudėtingoms ir nuolat kintančios aplinkos reikalavimams.

3.2.1. PPO su adaptyviąja KL bauda

Šioje PPO versijoje vietoj atnaujinimo santykio apkarpymo naudojamas KL skirtumo baudos strategija [SWD+17b], siekiant apriboti, kiek naujoji politika gali nukrypti nuo senosios. Adaptyvusis KL baudos (angl. Kullback-Leibler penalty) koeficientas β dinamiškai koreguojamas pagal tai, kiek naujoji politika skiriasi nuo senosios politikos, naudojant KL skirtumą kaip skirtumo tarp dviejų paskirstymų matų. Pseudo – kodas:

Pradėti politiką π_θ, vertės funkcija V_θ, ir KL baudos kooficiantas β
for Kiekvienai iteracijai **do**
    for Kiekvienam epizodui **do**
        Trajektorijos gavimas $\{s_t, a_t, r_t\}$ iš aplinkos
        Apskaičiuojamas pranašumų įvertinimai:

$$A_t = r_t + \gamma \cdot V_\theta(s_{t+1}) - V_\theta(s_t)$$

    end for
    for Kiekvienai transakcijai $\{s_t, a_t, r_t, A_t\}$ **do**
        Apskaičiuoti tikimybės santykį:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

        Apskaičiuojama KL divergencija:

$$\text{KL}_t = D_{\text{KL}}[\pi_{\theta_{\text{old}}}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)]$$

        Apskaičiuojamas tikslas su adaptyvia KL bauda:

$$L_{\text{KL}} = r_t(\theta) \cdot A_t - \beta \cdot \text{KL}_t$$

        Apskaičiuojamas vertės funkcijos praradimas:

$$L_{\text{VF}} = (V_\theta(s_t) - r_t)^2$$

12: Apskaičiuojamas bendras praradimas:

$$\text{Loss} = L_{\text{KL}} - c_1 \cdot L_{\text{VF}} + c_2 \cdot S[\pi_\theta(s_t)]$$

13: Atjauninamas θ naudojant „Adam“ optimizatorių
14: Pritaikomas β remiantis tiksline KL divergencija
15: **end for**
16: **end for**

Pagrindiniai parametrai yra tokie: β yra adaptyvus KL baudos koeficientas, kuris keičiasi atsižvelgiant į politikos nukrypimą nuo senojo. Jei KL skirtumai tarp politikos krypčių viršija iš anksto nustatytą slenkstį, bauda β padidėja, sumažinant būsimų atnaujinimų mastą. Ir atvirkščiai, jei skirtumas yra per mažas, β sumažėja, todėl galima atlikti daugiau atnaujinimų. KL-divergencija D_{KL} apskaičiuojama kaip $D_{\text{KL}}(\pi_{\theta_{\text{old}}} \parallel \pi_\theta)$, kuris matuoja, kiek naujoji politika nukrypo nuo ankstesnės. Nuostolių funkcija L_{KL} apima standartinį politikos santykio terminą $r_t(\theta) \cdot A_t$, su papildoma bauda už pernelyg didelį KL skirtumą. Vertės funkcijos praradimas L_{VF} išlieka panašus į standartinį PPO, matuojant agento vertės įvertinimų klaidą. Entropijos terminas $S[\pi_\theta(s_t)]$ vėl skatina tyrinėti, o c_1 ir c_2 atitinkamai koreguoja vertės funkcijos ir entropijos santykinius svorius. KL skirtumo slenkstis δ užtikrina, kad naujinimai būtų grąžinti, jei politikos naujinimas sukelia pernelyg didelį skirtumą.

3.2.2. PPO su pasitikėjimo regiono optimizavimu

PPO su patikimumo regiono optimizavimu apriboja atnaujinimo veiksmus naudojant patikimumo regiono apribojimą [WHT+19b], pagrįstą KL skirtumu. Skirtingai nuo iškirpimo ar adaptyviosios KL baudos, pasitikėjimo regiono metodas užtikrina, kad politika būtų atnaujinta ribotame regione ir ją būtų galima toliau tobulinti. Pseudo – kodas:

1: Būsena inicija politiką π_θ ir reikšmės funkciją V_θ
2: **for** Kiekvienai iteracijai **do**
3: **for** Kiekvienam epizodui **do**
4: Trajektorijos gavimas $\{s_t, a_t, r_t\}$ iš aplinkos
5: Apskaičiuojamas pranašumų įvertinimai:

$$A_t = r_t + \gamma \cdot V_\theta(s_{t+1}) - V_\theta(s_t)$$

kur γ yra nuolaidos koeficientas.
6: **end for**
7: **for** Kiekvienai mažesnei iteracijai **do**
8: Apskaičiuoti tikimybės santykį:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

kur $r_t(\theta)$ yra naujosios ir senosios politikos santykis.

Apskaičiuojamas KL skirtumas:

$$\mathrm{KL}_t = D_{\mathrm{KL}}[\pi_{\theta_{\text{old}}}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)]$$

Apribojami politikos atnaujinimai:

if $\mathrm{KL}_t > \delta$ **then**

Grąžinamas politikos atnaujinimas, kur δ yra KL skirtumo slenkstis.

end if

Apskaičiuojama tikslo funkcija:

$$L_{\mathrm{TRPO}} = r_t(\theta) \cdot A_t$$

Apskaičiuojamas vertės funkcijos praradimas:

$$L_{\mathrm{VF}} = (V_\theta(s_t) - r_t)^2$$

Apskaičiuojamas bendras praradimas:

$$\mathrm{Loss} = L_{\mathrm{TRPO}} - c_1 \cdot L_{\mathrm{VF}} + c_2 \cdot S[\pi_\theta(s_t)]$$

kur $S[\pi_\theta(s_t)]$ yra entropijos narys, o c_1 ir c_2 yra vertės funkcijos praradimo ir entropijos reguliarumo koeficientai.

Atnaujinamas θ naudojant pasitikėjimo regiono metodą.

end for

end for

Pagrindiniai parametrai yra tokie: Pasitikėjimo regiono dydis δ yra hiperparametras, kontroliuojantis didžiausią leistiną KL skirtumą tarp naujos ir senos politikos. Jei KL skirtumas D_{KL} viršija šią ribą, politikos atnaujinimas atšaukiamas, užtikrinant stabilumą mokymosi proceso metu. Privalumų funkcija A_t matuoja, kiek geresnis pasirinktas veiksmas, palyginti su vidutiniu veiksmu tam tikroje būsenoje, o $r_t(\theta)$ rodo tikimybės santykį tarp naujos ir senos politikos, kaip standartiniame PPO metode.

Pasitikėjimo regiono tikslo funkcija L_{TRPO} užtikrina, kad politikos pranašumu svertinis tikimybės koeficientas neviršytų saugaus pasitikėjimo regiono. Vertės funkcijos praradimas L_{VF}, entropijos terminas $S[\pi_\theta(s_t)]$ ir konstantos c_1 ir c_2 nepakitę lyginant pagal standartinį PPO metodą, kontroliuodami vertės įvertinimo tikslumą ir skatinant tyrinėjimą. Pasitikėjimo regionas užtikrina, kad politikos atnaujinimai būtų suvaržyti (apriboti), o tai leidžia stabiliau mokytis.

3.2.3. SAC su fiksuota temperatūra

SAC leidžia didinti atlygį ir taip pat maksimaliai padidinti politikos entropiją skatinant tyrinėjimą. Fiksuotos temperatūros variante temperatūros (angl. fixed-temperature variant) α parametras išlieka pastovus per visą mokymą [WN20], kontroliuojant tyrinėjimo ir išnaudojimo kompromisą. Tai naudinga, nes leidžia agentui mokytis iš ankstesnių epizodų, nereikalaujant laikytis tos pačios tyrinėjimo strategijos, kaip procese, kuris vyksta tuo metu. Pseudo – kodas:

1: Inicijuojama politika π_θ, Q tinklai Q_{ϕ_1} ir Q_{ϕ_2}, reikšmės funkcija V_ψ su fiksuota temperatūra α
2: **for** Kiekvienai iteracijai **do**
3: **for** Kiekvienam epizodui **do**
4: Perėjimų (angl. transitions) surinkimas $\{s_t, a_t, r_t, s_{t+1}\}$ iš aplinkos
5: Perėjimų saugojimas pakartojimo buferyje
6: **end for**
7: **for** Kiekvienai mažesnei iteracijai (pakartojimo buferyje) **do**
8: Apskaičiuojamas Q tikslas:

$$y = r_t + \gamma \cdot V_\psi(s_{t+1})$$

9: Apskaičiuojamas Q tinklo praradimas:

$$L_Q = (Q_\phi(s_t, a_t) - y)^2$$

10: Atnaujinami Q tinklai su gradiento mažėjimu
11: Apskaičiuojamas politikos praradimas:

$$L_\pi = \mathbb{E}\left[\log(\pi_\theta(a_t \mid s_t)) - Q_\phi(s_t, a_t)\right]$$

12: Atnaujinamas politikos tinklas naudojant gradiento mažėjimą
13: Apskaičiuojamas vertės funkcijos tikslas:

$$V_{\text{target}} = \min(Q_{\phi_1}(s_t, a_t), Q_{\phi_2}(s_t, a_t)) - \alpha \cdot \log(\pi_\theta(a_t \mid s_t))$$

14: Apskaičiuojamas vertės funkcijos praradimas:

$$L_V = (V_\psi(s_t) - V_{\text{target}})^2$$

15: Atnaujinama vertės funkcija V_ψ naudojant gradiento mažėjimą
16: **end for**
17: **end for**

Pagrindiniai parametrai apima temperatūros parametrą α, kuris nustato, kiek atsitiktinumo

politika yra įvedama arba pasitelkiama pasirenkant konkretų veiksmą. Didesnė α vertė skatina daugiau tyrinėti didinant entropiją, o mažesnė vertė skatina išnaudojimą, daugiau dėmesio skiriant deterministiniams veiksmams. SAC algoritmas taip pat apima du Q vertės tinklus Q_{ϕ_1} ir Q_{ϕ_2}, kurie naudojami Q mokymosi metodų pervertinimui sumažinti. Vertės funkcija V_ψ numato numatomą grąžą iš nurodytos būsenos, naudojama apskaičiuoti numatomą politikos našumą.

Politikos nuostolis L_π apskaičiuojamas kaip numatoma atliktų veiksmų tikimybė, pakoreguota pagal tų veiksmų Q reikšmę. Tai skatina politiką pasirinkti veiksmus, kurių numatoma grąža didesnė, išlaikant pakankamą atsitiktinumą. Vertės tinklo tikslas apskaičiuojamas pasirinkus mažiausią iš dviejų Q reikšmių ir atėmus strategijos entropijos svertinę loginę tikimybę. Tai užtikrina, kad vertės funkcija tiksliai atspindėtų stochastinės politikos našumą.

3.2.4. SAC su automatiniu temperatūros reguliavimu

SAC automatinio temperatūros reguliavimo variante temperatūros parametras α nėra fiksuotas, bet dinamiškai koreguojamas pagal politikos entropiją [TYL+24]. Tai leidžia agentui pritaikyti savo tyrinėjimo lygį pagal aplinkos sudėtingumą ir dabartinę politikos būklę. Pseudo – kodas:

1: Inicijuojama politika π_θ, Q tinklai Q_{ϕ_1} and Q_{ϕ_2}, reikšmės funkcija V_ψ su temperatūra α
2: **for** Kiekvienai iteracijai **do**
3: **for** Kiekvienam epizodui **do**
4: Perėjimų surinkimas $\{s_t, a_t, r_t, s_{t+1}\}$ iš aplinkos
5: Perėjimų saugojimas pakartojimo buferyje
6: **end for**
7: **for** Kiekvienai mažesnei iteracijai (pakartojimo buferyje) **do**
8: Apskaičiuojamas Q tikslas:

$$y = r_t + \gamma \cdot V_\psi(s_{t+1})$$

9: Apskaičiuojamas Q tinklo praradimas:

$$L_Q = (Q_\phi(s_t, a_t) - y)^2$$

10: Atnaujinami Q tinklai su gradiento mažėjimu
11: Apskaičiuojamas politikos praradimas:

$$L_\pi = \mathbb{E}\left[\log(\pi_\theta(a_t \mid s_t)) - Q_\phi(s_t, a_t)\right]$$

12: Atnaujinamas politikos tinklas naudojant gradiento mažėjimą

13: Reguliuojama temperatūra pagal entropiją:

$$\alpha_t = \alpha_{t-1} \cdot \exp(\kappa \cdot (\text{target\_entropy} - \text{current\_entropy}))$$

14: Apskaičiuojama tinklo tikslo vertė:

$$V_{\text{target}} = \min(Q_{\phi_1}(s_t, a_t), Q_{\phi_2}(s_t, a_t)) - \alpha_t \cdot \log(\pi_\theta(a_t \mid s_t))$$

15: Apskaičiuojamas vertės funkcijos praradimas:

$$L_V = (V_\psi(s_t) - V_{\text{target}})^2$$

16: Atnaujinama vertės funkcija V_ψ naudojant gradiento mažėjimą

17: **end for**

18: **end for**

Pagrindiniai parametrai apima prisitaikantį temperatūros parametrą α_t, kuris koreguojamas kiekvienu laiko žingsniu, atsižvelgiant į skirtumą tarp esamos entropijos ir iš anksto nustatytos tikslinės entropijos. Tikslinė entropija dažnai nustatoma atsižvelgiant į veiksmo erdvės matmenis (pvz., $-\dim$(veiksmo erdvė)) ir nustato norimą tyrinėjimo lygį. Temperatūros atnaujinimo taisyklė naudoja konstantą κ, kuri reguliuoja temperatūros prisitaikymo greitį. Jei dabartinė entropija yra mažesnė už tikslinę entropiją, α_t yra padidinama skatinant didesnį tyrinėjimą. Ir atvirkščiai, jei entropija viršija tikslą, α_t sumažinamas, kad sumažėtų stochastika. Likusi algoritmo dalis apima standartinius SAC komponentus, įskaitant du Q tinklus Q_{ϕ_1} ir Q_{ϕ_2}, vertės tinklą V_ψ, ir politikos praradimą L_π, kuris koreguojamas siekiant išlaikyti pusiausvyrą tarp tyrinėjimo ir eksploatavimo. Vertės funkcijos tikslas apima dinamiškai koreguojamą temperatūros terminą α_t, užtikrinant, politika laikui bėgant bus veiksmingai pritaikyta.

3.3. Palyginimas tarp keturių metodų

Šioje lentelėje pateiktas keturių skirtingų metodų versijų: PPO su adaptyviąja KL bauda, PPO su pasitikėjimo regiono optimizavimu, SAC su fiksuota temperatūra ir SAC su automatiniu temperatūros reguliavimu – palyginimas. Lentelėje aptariami pagrindiniai metodų skirtumai, privalumai ir trūkumai, siekiant pabrėžti jų ypatumus ir tinkamumą skirtingoms problemoms.

1 lentelė. Palyginimas tarp PPO ir SAC metodų su modifikacijomis

Metodas	**Pagrindiniai skirtumai**	**Privalumai**	**Trūkumai**
PPO su adaptyvia KL bauda	Naudoja adaptyvų KL divergencijos baudos koeficientą β, kuris dinamiškai reguliuoja politikos atnaujinimus, remiantis politikos pokyčiais.	1. Automatiškai reguliuojamas β, leidžia optimalų tyrinėjimo ir eksploatavimo balansą. 2. Stabilus net sudėtingose aplinkose.	1. Reikalauja papildomų skaičiavimų KL divergencijos skaičiavimui. 2. Hiperparametro β optimizavimas gali būti sudėtingas.
PPO su pasitikėjimo regiono optimizavimu	Apriboja politikos atnaujinimus KL divergencijos slenksčiu δ. Jei divergencija viršija δ, atnaujinimas atšaukiamas.	1. Užtikrina stabilų mokymąsi, ribojant didelius politikos pokyčius. 2. Lengviau valdomas nei adaptyvios KL baudos metodas.	1. Griežti apribojimai gali sulėtinti mokymosi greitį. 2. Netinkamas δ pasirinkimas gali riboti politikos progresą.
SAC su fiksuota temperatūra	Naudoja pastovų temperatūros koeficientą α, kuris reguliuoja politikos entropiją, o tyrinėjimo intensyvumas lieka nekintamas.	1. Pastovi α vertė užtikrina paprastumą ir stabilumą. 2. Naudojamas dvigubas Q tinklas mažina pervertinimo riziką.	1. Nereguliuoja tyrinėjimo intensyvumo pagal aplinkos sudėtingumą. 2. Netinkamai parinkta α vertė gali lemti suboptimalų veikimą.
SAC su automatiniu temperatūros reguliavimu	Temperatūros parametras α dinamiškai reguliuojamas pagal skirtumą tarp politikos entropijos ir tikslinės entropijos.	1. Prisitaiko prie aplinkos sudėtingumo. 2. Sumažina rankinio (angl. manual) parametrų derinimo poreikį. 3. Optimizuoja tyrinėjimo ir eksploatavimo balansą.	1. Papildomi skaičiavimai dėl α reguliavimo. 2. Sunku parinkti tinkamą prisitaikymo greičio parametrą κ.

4. Tyrimo rezultatai

Šioje dalyje yra atliktas tyrimas - išsiaiškinti, kuris metodas agentui mokyti yra geriausias ir pagal kokius parametrus. Mokymosi procesas yra pradedamas paleidžiant Python kodą (atitinkamai kiekvienam metodui), kuris sąveikauja su Python API ir Unity Engine programa, t.y. instrukcijos yra siunčiamos į aplinką nurodant, kaip ir kur agentas turi judėti, o aplinka grąžina rezultatą - teigiamą arba neigiamą. Kadangi agentai yra mokomi dviejose aplinkose (pirmoji aplinka lengvesnė, o antroji sudėtingesnė), iš viso yra 8 mokymo variacijos - 4 metodai pirmoje aplinkoje (žr. 1 pav. prieduose), ir tie patys 4 metodai antrojoje aplinkoje [Aur19](žr. 2 pav. prieduose).

4.1. Parametrų radimas metodams

Prieš pradedant pagrindinį agentų mokymąsi, reikia išsiaiškinti, kokias priskirti reikšmes patiems parametrams. Kadangi yra siekiama metodus ištestuoti lygiagrečiai, pagrindiniai parametrai metodams (PPO - „gamma“, c1, c2, „learning rate“, „beta initial“ ir SAC - „gamma“, „learning rate“, „batch size“, „batch size“) yra naudojamos tos pačios reikšmės. Iš viso yra išskirtos po 10 geriausių variacijų kiekvienam metodui, o pagrindiniai parametrai yra paryškinti pirmose lentelių eilutėse(žr. 2,3,4 ir 5 lenteles).

4.1.1. PPO su adaptyviąja KL bauda parametrai

Parametrų reikšmės: „gamma“ (nuolaidos koeficientas) siekiama įvertinti ilgalaikį ir trumpalaikį atlygio prioritetą; c1 (vertės praradimo koeficientas) svyravo nuo 0,3 iki 0,7, siekiant ištirti jo poveikį vertės funkcijų atnaujinimų svarbai; c2 (entropijos koeficientas) koreguojamas tarp 0,05 ir 0,2, valdant tyrinėjimo lygį; „target KL divergence“ (Tikslinis KL skirtumas) svyruoja nuo 0,005 iki 0,02, įvertinant, koks griežtas turėtų būti KL skirtumo apribojimas; „learning rate” (mokymosi greitis) koreguojamas tarp 3e-5 ir 5e-4, tikrinant stabilumą ir konvergencijos greitį; „beta initial“ (beta pradinė KL bauda) keičiama nuo 0,05 iki 0,2, ištiriant jos poveikį KL skirtumui treniruotės pradžioje (žr. 2 lentelę).

2 lentelė. Parametrų variacijos „PPO su adaptyviąja KL bauda“ metodui

Variacija	Gamma	C1	C2	Target KL	Learning Rate	Beta Initial
1	**0.99**	**0.5**	**0.1**	0.01	**3e-4**	**0.1**
2	0.99	0.6	0.05	0.01	1e-4	0.05
3	0.99	0.4	0.2	0.015	5e-4	0.15
4	0.99	0.5	0.15	0.02	2e-4	0.1
5	0.99	0.7	0.1	0.01	3e-5	0.2
6	0.99	0.6	0.08	0.005	4e-4	0.08
7	0.99	0.5	0.1	0.02	3e-4	0.1

Variacija	Gamma	C1	C2	Target KL	Learning Rate	Beta Initial
8	0.99	0.4	0.12	0.01	1e-4	0.05
9	0.99	0.3	0.1	0.008	5e-5	0.2
10	0.99	0.5	0.15	0.012	2e-4	0.12

4.1.2. PPO su pasitikėjimo regiono optimizavimu parametrai

Parametrų reikšmės: „gamma“ (nuolaidos koeficientas) siekiama įvertinti ilgalaikį ir trumpalaikį atlygio prioritetą; c1 (vertės praradimo koeficientas) svyravo nuo 0,3 iki 0,7, siekiant ištirti jo poveikį vertės funkcijų atnaujinimų svarbai; c2 (entropijos koeficientas) koreguojamas tarp 0,05 ir 0,2, valdant tyrinėjimo lygį; „KL threshold“ (KL slenkstis) eksperimentuojama tarp 0,008 ir 0,02, siekiant patikrinti politikos stabilumą; „learning rate“ (mokymosi greitis) koreguojamas tarp 1e-4 ir 5e-4, tikrinant stabilumą ir konvergencijos greitį; „beta“ (beta pradinė KL bauda) keičiama nuo 0,05 iki 0,2, ištiriant jos poveikį KL skirtumui treniruotės pradžioje (žr. 3 lentelę).

3 lentelė. Parametrų variacijos „PPO su pasitikėjimo regiono optimizavimu“ metodui

Variacija	Gamma	C1	C2	KL Threshold	Learning Rate	Beta Initial
1	**0.99**	**0.5**	**0.1**	0.01	**3e-4**	**0.1**
2	0.99	0.6	0.05	0.015	1e-4	0.2
3	0.99	0.4	0.15	0.02	5e-4	0.05
4	0.99	0.7	0.2	0.008	2e-4	0.12
5	0.99	0.3	0.08	0.012	3e-5	0.15
6	0.99	0.5	0.12	0.01	4e-4	0.08
7	0.99	0.6	0.1	0.015	3e-4	0.1
8	0.99	0.4	0.07	0.02	1e-4	0.2
9	0.99	0.7	0.15	0.01	5e-5	0.05
10	0.99	0.5	0.1	0.008	2e-4	0.12

4.1.3. SAC su fiksuota temperatūra parametrai

Parametrų reikšmės: „gamma“ (nuolaidos koeficientas) siekiama įvertinti ilgalaikį ir trumpalaikį atlygio prioritetą; „alpha“ (fiksuota temperatūra arba entropijos koeficientas) eksperimentuojama tarp 0.1 ir 0.35,valdo kompromisą tarp tyrinėjimo (aukštesnė „alpha“ reikšmė) ir eksploatavimo (žemesnė „alpha“ reikšmė); „learning rate“ eksperimentuojama tarp 1e-4 ir 5e-4, tai greitis, kuriuo neuroninis tinklas atnaujinamas mažėjančio gradiento metu. Mažesnės vertės lemia stabilesnę treniruotę; „batch size“ (partijos dydis) eksperimentuojama

tarp 32 ir 128, mėginių, naudojamų per mokymo iteraciją, skaičius. Didesnės vertės leidžia sklandžiau atnaujinti gradientą, tačiau reikia daugiau atminties; „batch size“ (pakartojimo buferio dydis) eksperimentuojama tarp 500,000 ir 2,000,000, tai yra pakartojimo buferio talpa patirčiai saugoti. Didesnės reikšmės leidžia gauti įvairesnių treniruočių duomenų, tačiau reikia daugiau atminties (žr. 4 lentelę).

4 lentelė. Parametrų variacijos „SAC su fiksuota temperatūra“ metodui

Variacija	Gamma	Alpha	Learning Rate	Batch Size	Replay Buffer Size
1	**0.99**	0.2	**3e-4**	**64**	**1000000**
2	0.99	0.25	1e-4	128	500000
3	0.99	0.15	5e-4	64	2000000
4	0.99	0.3	2e-4	32	1000000
5	0.99	0.1	3e-5	128	1500000
6	0.99	0.35	4e-4	64	750000
7	0.99	0.2	3e-4	96	2000000
8	0.99	0.25	1e-4	64	500000
9	0.99	0.3	5e-5	128	1000000
10	0.99	0.2	2e-4	64	2000000

4.1.4. SAC su automatiniu temperatūros reguliavimu

Parametrų reikšmės: „gamma“ (nuolaidos koeficientas) siekiama įvertinti ilgalaikį ir trumpalaikį atlygio prioritetą; „initial alpha“ (pradinė temperatūra) eksperimentuojama tarp 0.1 ir 0.4, kontroliuojanti tyrinėjimo ir eksploatavimo lygį. Kuo didesnė jos vertė, tuo labiau skatinamas tyrinėjimas; „learning rate“ eksperimentuojama tarp 1e-4 ir 5e-4, tai greitis, kuriuo neuroninis tinklas atnaujinamas mažėjančio gradiento metu. Mažesnės vertės lemia stabilesnę treniruotę; „batch size“ (partijos dydis) eksperimentuojama tarp 32 ir 128, mėginių, naudojamų per mokymo iteraciją, skaičius. Didesnės vertės leidžia sklandžiau atnaujinti gradientą, tačiau reikia daugiau atminties; „batch size“ (pakartojimo buferio dydis) eksperimentuojama tarp 500,000 ir 2,000,000, tai yra pakartojimo buferio talpa patirčiai saugoti. Didesnės reikšmės leidžia gauti įvairesnių treniruočių duomenų, tačiau reikia daugiau atminties; „target entropy scale“ (tikslinės entropijos didinimas) eksperimentuojama tarp 0.002 ir 0.01, tikslinės entropijos daugiklis, skirtas valdyti norimą veiksmų atsitiktinumą (žr. 5 lentelę).

5 lentelė. Parametrų variacijos „SAC su automatiniu temperatūros reguliavimu“ metodui

Var.	Gamma	Initial Alpha	Learning Rate	Batch Size	Replay Buffer Size	Target Entropy Scale	Kappa
1	**0.99**	0.2	**3e-4**	**64**	**1000000**	-1.0	0.005
2	0.99	0.3	1e-4	128	500000	-0.5	0.01
3	0.99	0.1	5e-4	64	2000000	-1.5	0.002
4	0.99	0.4	2e-4	32	1000000	-0.8	0.01
5	0.99	0.25	3e-5	128	1500000	-1.2	0.007
6	0.99	0.35	4e-4	64	750000	-1.0	0.005
7	0.99	0.15	3e-4	96	2000000	-1.3	0.002
8	0.99	0.2	1e-4	64	500000	-1.0	0.01
9	0.99	0.3	5e-5	128	1000000	-0.9	0.007
10	0.99	0.25	2e-4	64	2000000	-1.4	0.005

4.2. Metodų palyginimas aplinkose

Kiekvieno metodo testo metu aplinkoje metodas buvo paleidžiamas po 10 kartų, o geriausi metodo rezultatai yra atvaizduojami 6 ir 7 lentelėse. Kiekvienu metodu treniruočių metu privaloma nustatyti, kaip metodas veikia - geriau ar blogiau, po 2 tūkstančių iteracijų lyginant su kitais paleidimais atsižvelgiant į parametrų reikšmes. Taip pat, agentų mokymosi metu yra fiksuojamas vidutinis atlygio vidurkis tuo metu (žr. 3 pav. ir 4 pav. prieduose) - galima pastebėti, kad laikas kiekvienam metodui mokyti yra skirtingas - taip yra dėl to, nes agentui užbaigti vieną iteraciją gali užtrukti daugiau arba mažiau laiko.

4.2.1. Mokymosi procesas pirmojoje aplinkoje

Palyginamieji pirmosios aplinkos rezultatai rodo, kad SAC su automatiniu temperatūros reguliavimu pranoko visus kitus metodus ir gavo didžiausią atlygį (48,2) ir mažiausią vertės praradimą (0,025). Tai parodo metodo gebėjimą dinamiškai pritaikyti savo entropijos koeficientą, efektyviai optimizuojant tyrinėjimą ir naudojimą aplinkoje. Priešingai, PPO su patikimo regiono optimizavimu, išlaikant stabilumą ir pasiekus tinkamą atlygį (42,8), pritaikomumu ir efektyvumu pranoko SAC su dinaminiu temperatūros reguliavimu. Panašiai PPO su adaptyviąja KL bauda gavo 40,5 atlygį, demonstruodamas stabilius atnaujinimus, bet šiek tiek sunkiau sekėsi dinamiškoje aplinkoje dėl iš anksto nustatytų baudų koregavimų. Galiausiai, SAC su fiksuota temperatūra pasirodė silpniausiai, surinkęs 38,7 atlygį, kurį ribojo nesugebėjimas dinamiškai reguliuoti entropijos, o tai ribojo tyrinėjimą šioje aplinkoje. Šiais rezultatais remiantis, galima pabrėžti, kad SAC pranašumas su automatiniu temperatūros reguliavimu scenarijuose yra

didžiausias, kuriuose reikia prisitaikyti ir yra efektyvus optimizuojant politiką ypač lengvesnėse (netokiose sudėtingose) aplinkose (žr. 6 lentelę).

6 lentelė. Metodų testų rezultatai pirmojoje aplinkoje

Metodas	**Apdovanojimas /epizodas**	**Nuostoliai /politika**	**Nuostolis /vertė**	**Kt. metrikos**
PPO su adapt. KL bauda	40.5	-0.040	0.048	KL divirgacija: 0.012
PPO su pasitik. reg. opt.	42.8	-0.035	0.041	Beta: 0.08
SAC su fiksuota temp.	38.7	-	0.036	Alpha: 0.2 (Užfiksuota)
SAC su aut. temp. reg.	48.2	-	0.025	Alpha: 0.18 (Keičiama)

4.2.2. Mokymosi procesas antrojoje aplinkoje

Keturių taikomų mokymo metodų lyginamoji analizė antrojoje aplinkoje (žr. 7 lentelę) išryškina unikalias jų stipriąsias ir silpnąsias puses. PPO su adaptyviąja KL bauda pasiekė vidutinį našumą, už kiekvieną seriją gaunamas 25,2 atlygį, naudodamasis stabiliais atnaujinimais, bet tuo pačiu rodydamas didesnį vertės praradimą, o tai rodo, kad kyla problemų, susijusių su vertės funkcijos aproksimavimu. PPO su patikimo regiono optimizavimu parodė geriausią našumą, pasiekęs didžiausią atlygį (27,5) ir pademonstravęs didelį stabilumą dėl pasitikėjimo regiono apribojimų, todėl jis puikiai tiko stabiliems ir didelio atlygio scenarijams. Priešingai, SAC su fiksuota temperatūra surinko mažiausią atlygį (22,1), nes fiksuotas entropijos koeficientas apribojo jo prisitaikymą dinaminėje aplinkoje ir trukdė tyrinėti. Tačiau SAC su automatiniu temperatūros reguliavimu pasižymėjo dideliu prisitaikymu, gaudamas 26,3 atlygį, panaudojęs dinaminį entropijos reguliavimą norint efektyviai optimizuoti tyrinėjimo ir eksploatavimo kompromisą, išlaikant santykinai mažą vertės praradimą. Remiantis rezultatais, PPO su patikimo regiono optimizavimu yra efektyviausias būdas pasiekti aukštą atlygį ir stabilumą, o SAC su automatiniu temperatūros reguliavimu puikiai tinka aplinkoje, kurioje reikia prisitaikyti. Tačiau, tinkamiausio metodo pasirinkimas galiausiai priklauso nuo aplinkos sudėtingumo ir poreikio dinamiškai koreguojant politiką mokymo metu (žr. 7 lentelę).

7 lentelė. Metodų testų rezultatai antrojoje aplinkoje

Metodas	**Apdovanojimas /epizodas**	**Nuostoliai /politika**	**Nuostolis /vertė**	**Kt. metrikos**
PPO su adapt. KL bauda	25.2	-0.034	0.042	KL divirgacija: 0.011
PPO su pasitik. reg. opt.	27.5	-0.028	0.037	Beta: 0.09
SAC su fiksuota temp.	22.1	-	0.031	Alpha: 0.2 (Užfiksuota)
SAC su aut. temp. reg.	26.3	-	0.029	Alpha: 0.19 (Keičiama)

4.3. Rezultatų validavimas

„Kruskal-Wallis“ testas[Con99], po kurio atliekami „Dunno post hoc“[Zar10] palyginimai, naudojami statistiškai patvirtinti keturių sustiprinimo mokymosi metodų, įvertintų skirtingose Unity varikliu sukurtose aplinkose, našumo skirtumus. Atsižvelgiant į tai, kad epizodų atlygio pasiskirstymas dažnai nukrypsta nuo normalumo ir gali turėti išskirtinių verčių, tinkamas neparametrinis metodas. „Kruskal-Wallis“ testas įvertina, ar yra statistiškai reikšmingų skirtumų tarp metodų medianinių našumų, nedarant prielaidos, kad dispersijos ar normalusis pasiskirstymas yra vienodi. Aptikus reikšmingą bendrą efektą, „Dunno“ testas taikomas keliems poriniams palyginimams atlikti, kontroliuojant I tipo klaidas tokiais metodais kaip „Bonferroni„ korekcija. Ši korekcija leidžia nustatyti konkrečias metodų poras, kurių našumas labai skiriasi. Kartu šie testai pateikia griežtus statistinius įrodymus, pagrindžiančius išvadas apie tai, kurie metodai geriau veikia skirtingomis aplinkos sąlygomis, užtikrinant, kad stebimi skirtumai nėra atsitiktiniai, o atspindi reikšmingus mokymosi efektyvumo ar apibendrinimo gebėjimo skirtumus.

4.3.1. Rezultatų validavimas pirmojoje aplinkoje

Atliekama 10 vertinimo epizodų kiekvienam metodui ir užfiksuojamas bendras epizodų atlygio skaičius. Iš viso yra 40 stebėjimų.

8 lentelė. Vertinimo epizodo apdovanojimai metodams pirmojoje aplinkoje

Metodas	**Epizodų įvertinimai**
PPO su adapt. KL bauda	39.8, 41.0, 40.1, 39.5, 40.6, 40.7, 40.3, 40.4, 40.0, 41.2
PPO su pasitik. reg. opt.	42.1, 42.8, 42.6, 42.4, 42.9, 43.0, 42.3, 42.5, 42.7, 43.1
SAC su fiksuota temp.	38.1, 38.7, 39.0, 39.3, 39.1, 39.2, 38.6, 39.4, 38.9, 39.5
SAC su aut. temp. reg.	47.5, 48.0, 47.8, 48.3, 48.5, 48.6, 48.1, 48.4, 47.9, 48.2

Metodams apskaičiuojamos rango sumos. Svarbu pabrėžti, kad tai nėra apdovanojimų suma, o rangų - apdovanojimai yra išdėstomi didėjančia tvarka, taip priskiriant rango reikšmę (nuo mažiausios iki didžiausios). Pagal 8 lentelės įvertinimus yra atrenkami priskirti rangai. Taip pat gaunamas rango vidurkis tolimesniam testavimui - kadangi kiekvienas metodas turi po 10 epizodų, rango suma yra dalinama iš 10.

9 lentelė. Kiekvieno metodo rangų suma ir jų vidurkiai 1 aplinkoje

Metodas	$\sum R_i$	$\bar{R}_i$
PPO su adapt. KL bauda	120.5	12.05
PPO su pasitik. reg. opt.	285.5	28.55
SAC su fiksuota temp.	55.0	5.50
SAC su aut. temp. reg.	379.0	37.90

Rangų sumos apskaičiavimas visoms grupėms, gauta bendra suma, kuri yra $\sum R = 820$.

$$\begin{aligned} \text{Stebėjimų skaičius} \quad & N = 40 \\ \text{Rangų suma nuo 1 iki } N \quad & = \sum_{i=1}^{40} i = \frac{N(N+1)}{2} \\ & = \frac{40 \cdot 41}{2} = 820 \end{aligned} \tag{10}$$

Pagal gautą „Kruskal-Wallis“ H statistikos reikšmę, kuri yra 54.58, galima nustatyti, kad bent vienas sustiprinto mokymosi metodas vertintoje aplinkoje savo veikimu reikšmingai skiriasi nuo kitų. Todėl yra būtina naudoti „Dunno post hoc“ testą porų palyginimų, siekiant nustatyti, kurie konkretūs metodai skiriasi ir kuris metodas veikia geriausiai.

$$\begin{aligned} H &= \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1) \\ &= \frac{12}{40 \cdot 41} \left(\frac{120.5^2}{10} + \frac{285.5^2}{10} + \frac{55.0^2}{10} + \frac{379.0^2}{10} \right) - 3 \cdot 41 = 54.58 \end{aligned} \tag{11}$$

Reikalingas „p-value“ vertės apskaičiavimas „Dunno post hoc“ testui, su žinomomis reikšmėmis: H=54.58, judėjimo kryptys df = k - 1 = 4 - 1 = 3. Gauta „p-value“ reikšmė yra $p \approx 1.41 \times 10^{-11}$. Kadangi p $\ll$ 0.05, tai įrodo ženklų metodų našumo skirtumą. Python kodas:

```
from scipy.stats import chi2
p_value = 1 - chi2.cdf(54.58, df=3)
```

Čia apskaičiuojama „Dunno post hoc“ testo Z reikšmė, skirta mašinio mokymosi metodų poroms palyginti pagal jų vidutinius rangus. Kiekvienas metodas turi 10 stebėjimų, o jų rangų sumos dalijamos iš 10, gaunant vidutinį rangą. Z formulės vardiklis apskaičiuojamas iš bendro stebėjimų skaičiaus N = 40, gaunant bendrą dispersiją $\sqrt{27.334} \approx 5.23$. Ši reikšmė naudojama vidutinių rangų skirtumui tarp kiekvienos poros normalizavimui, leidžiant apskaičiuoti Z balus reikšmingumo testavimui.

$$Z = \frac{|\bar{R}_j - \bar{R}_k|}{\sqrt{\frac{N(N+1)}{12}\left(\frac{1}{n_j} + \frac{1}{n_k}\right)}} = \frac{|\bar{R}_j - \bar{R}_k|}{\sqrt{\frac{40 \cdot 41}{12} \cdot \left(\frac{1}{10} + \frac{1}{10}\right)}} = \frac{|\bar{R}_j - \bar{R}_k|}{5.23} \tag{12}$$

Pagal gautas Z reikšmes, metodai yra lyginami tarpusavyje. Pirmojoje aplinkoje geriausiai pasirodė SAC su automatiniu temperatūros reguliavimu. „Kruskal-Wallis“ testas ($H = 54.58$, $p \approx 1.41 \times 10^{-11}$) parodė, kad tarp metodų egzistuoja statistiškai reikšmingi našumo skirtumai. „Dunno post hoc“ analizė atskleidė, kad šis SAC su automatiniu temperatūros reguliavimu

metodas reikšmingai pranoko PPO su adapt. KL bauda ($Z = 4.94$, $p < 0.0001$) ir SAC su fiksuota temperatūra ($Z = 6.20$, $p < 0.0001$). Skirtumas tarp SAC su automatiniu temperatūros reguliavimu ir PPO su pasit. reg. optimizavimu metodais nebuvo statistiškai reikšmingas ($Z = 1.79$, $p = 0.0738$), tad abu metodai veikia panašiai. Visgi SAC su automatiniu temperatūros reguliavimu turėjo aukščiausią vidutinį rangą ($\bar{R}_i = 37.90$), todėl šis metodas yra efektyvesnis paprastesnėje aplinkoje, ypač dėl dinaminio entropijos reguliavimo, kuris leidžia greičiau ir stabiliau pasiekti optimalią politiką.

10 lentelė. „Dunno“ testo porinių metodų palyginimų rezultatai su Bonferroni korekcija pirmojoje aplinkoje, ($N = 40$, $n_i = 10$)

Palyginimas	**Reikšmės skirtumas**	**Z**	**Pirminė p**	**Koreguota p (×6)**	**Rezultatas**
SAC aut. vs PPO adapt. KL	25.85	4.94	<0.0001	<0.0006	✓ Svarbus
SAC aut. vs PPO pasit. reg.	9.35	1.79	0.073	0.438	– Nesvarbus
SAC aut. vs SAC fiksuota temp.	32.40	6.20	<0.0001	<0.0006	✓ Svarbus
PPO pasit. reg. vs SAC fiksuota	23.05	4.41	<0.0001	<0.0006	✓ Svarbus
PPO pasit. reg. vs PPO adapt. KL	16.50	3.15	0.0016	0.0096	✓ Svarbus
PPO adapt. KL vs SAC fiksuota	6.55	1.25	0.210	1.260	– Nesvarbus

4.3.2. Rezultatų validavimas antrojoje aplinkoje

Atliekama 10 vertinimo epizodų kiekvienam metodui ir užfiksuojamas bendras epizodų atlygio skaičius. Iš viso yra 40 stebėjimų.

11 lentelė. Vertinimo epizodo apdovanojimai metodams antrojoje aplinkoje

Metodas	**Epizodų įvertinimai**
PPO su adapt. KL bauda	24.7, 25.1, 25.3, 24.9, 25.5, 25.0, 25.6, 25.3, 25.1, 25.3
PPO su pasitik. reg. opt.	27.0, 27.4, 27.6, 27.1, 27.8, 27.3, 27.9, 27.5, 27.2, 27.6
SAC su fiksuota temp.	21.8, 22.0, 22.2, 22.4, 22.3, 22.5, 22.1, 22.2, 21.9, 22.0
SAC su aut. temp. reg.	26.0, 26.2, 26.4, 26.1, 26.6, 26.5, 26.3, 26.4, 26.2, 26.3

Metodams apskaičiuojamos rango sumos. Pagal 11 lentelės įvertinimus yra atrenkami priskirti rangai. Taip pat gaunamas rango vidurkis tolimesniam testavimui - kadangi kiekvienas metodas turi po 10 epizodų, rango suma yra dalinama iš 10.

12 lentelė. Kiekvieno metodo rangų suma ir jų vidurkiai 2 aplinkoje

Metodas	$\sum \boldsymbol{R_i}$	$\bar{\boldsymbol{R}}_{\boldsymbol{i}}$
PPO su adapt. KL bauda	166.0	16.6
PPO su pasitik. reg. opt.	355.0	35.50
SAC su fiksuota temp.	46.0	4.60
SAC su aut. temp. reg.	255.0	25.5

Pagal gautą „Kruskal-Wallis“ H statistikos reikšmę, kuri yra 38.5, galima nustatyti, kad bent vienas sustiprinto mokymosi metodas vertintoje aplinkoje savo veikimu reikšmingai skiriasi nuo kitų. Todėl yra būtina naudoti „Dunno post hoc“ testą porų palyginimų, siekiant nustatyti, kurie konkretūs metodai skiriasi ir kuris metodas veikia geriausiai. Taip pat galima pastebėti, kad H reikšmė yra panaši, lyginant su pirmąja aplinka - skirtumas atsiranda dėl to, nes metodų rangų išsidėstymas ir jų kontrastai tarp grupių yra labai panašūs abiejose aplinkose, nepaisant skirtingų apdovanojimų reikšmių.

$$
\begin{aligned}
H &= \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1) \\
&= \frac{12}{40 \cdot 41} \left(\frac{166.0^2}{10} + \frac{355.0^2}{10} + \frac{46.0^2}{10} + \frac{255.0^2}{10} \right) - 3 \cdot 41 = 38.5
\end{aligned} \tag{13}
$$

Reikalingas „p-value“ vertės apskaičiavimas „Dunno post hoc“ testui, su žinomomis reikšmėmis: H=38.5, judėjimo kryptys df = k - 1 = 4 - 1 = 3. Remiantis Python kodu, kuris buvo naudojamas pirmajai aplinkai, gauta „p-value“ reikšmė yra $p \approx 1.41 \times 8.44^{-12}$. Kadangi $p \ll 0.05$, tai įrodo ženklų metodų našumo skirtumą.

Pagal gautas Z reikšmes (remiantis 12 formule), metodai yra lyginami tarpusavyje. Antrojoje aplinkoje statistinė analizė parodė, kad geriausiai pasirodė PPO su pasitikėjimo regiono optimizavimu. Atlikus Kruskal–Wallis testą ($H = 38.5$, $p \approx 8.44 \times 10^{-12}$), paaiškėjo, kad bent vienas metodas ženkliai skyrėsi nuo kitų. „Dunno post hoc“ analizė atskleidė, kad PPO su pasitik. reg. optimizavimu reikšmingai pralenkė SAC su fiksuota temperatūra ($Z = 5.74$, $p < 0.0001$) ir PPO su adapt. KL bauda ($Z = 3.83$, $p = 0.0001$) metodus. Skirtumas tarp šio metodo ir SAC su automatiniu temperatūros reguliavimu nebuvo statistiškai reikšmingas ($Z = 1.91$, $p = 0.0558$), todėl šie du metodai gali būti laikomi panašaus lygio. Tačiau SAC automatinio reguliavimo metodas nepranoko PPO su pasitikėjimo regiono optimizavimu, nes turint reikšmingus laimėjimus prieš du iš trijų metodų ir nė vieno pralaimėjimo, PPO su pasitik. reg. opt. pasižymi kaip efektyviausias metodas šioje sudėtingesnėje aplinkoje. Taip pat, turėjo aukščiausią rango įvertinimą (35.5).

13 lentelė. „Dunno“ testo porinių metodų palyginimų rezultatai su Bonferroni korekcija antrojoje aplinkoje, $N = 40$, $n_i = 10$)

Palyginimas	**Reikšmės skirtumas**	**Z**	**Pirminė p**	**Koreguota p (×6)**	**Rezultatas**
SAC aut. vs PPO adapt. KL	25.85	4.94	<0.0001	<0.0006	✓ Svarbus
SAC aut. vs PPO pasit. reg.	10.00	1.91	0.0558	0.3347	– Nesvarbus
SAC aut. vs SAC fiksuota temp.	32.40	6.20	<0.0001	<0.0006	✓ Svarbus
PPO pasit. reg. vs SAC fiksuota	30.00	5.74	<0.0001	<0.0006	✓ Svarbus
PPO pasit. reg. vs PPO adapt. KL	20.00	3.83	0.0001	0.0008	✓ Svarbus
PPO adapt. KL vs SAC fiksuota	10.00	1.91	0.0558	0.3347	– Nesvarbus

5. Diskusija

Iš gautų tyrimo rezultatų galima nustatyti mažus patobulinimus, lyginant jau su esamais sustiprinimo mokymosi metodais, ypač derinant ir konfigūruojant entropijos adaptacijos ir politikos atnaujinimo apribojimus. Pirmojoje aplinkoje „Soft Actor-Critic“ (SAC) su automatiniu temperatūros reguliavimu pasiekė didžiausią atlygį (48,2) ir mažiausią vertės praradimą (0,025), pranokdamas standartinį SAC metodą, kurį tyrė kiti autoriai [HZH+19] 2018 metais. Nors ši autorių versija buvo skirta „Mujoco“ ir „Atari“ užduotims atlikti, kur entropijos derinimas pagerino mokymosi stabilumą, šio darbo eksperimente metodas taikomas naujose aplinkose su dinaminėmis sąlygomis ir parodoma, kad entropijos planavimas išlaiko savo pranašumą net ir už standartinių metodų ribų. Šiame darbe pastebėtas ryškesnis stabilumas (agentas daro mažiau klaidų ir pasiekia greičiau tikslą) ir galutinis atlygis rodo geresnę vertės nuostolių tvarkymo ir adaptyvios entropijos kontrolės integraciją.

Antrojoje aplinkoje PPO su pasitikėjimo regiono optimizavimu pasiekė didžiausią atlygį (27,5) ir mažiausią agento nukrypimą nuo politikos (-0.028), nurodant jo pranašumą sudėtinguose ir kintamuose scenarijuose. Tačiau, skirtingai nei kiti autoriai [WHT+19a], kurie modifikavo PPO su pasitikėjimo regionu (TRGPPO) su konservatyviomis iškirpimo ribomis - čia išbandytas metodas subalansuoja pasitikėjimo regiono apribojimus su šiek tiek agresyvesniu tyrinėjimu, todėl gaunamas didesnis kaupiamasis atlygis per mažiau epizodų - bet tokiu būdu agentas gaudavo daugiau baudų ir turėdavo mokymo epizodą pradėti iš naujo.

Apibendrinant, patobulinus entropijos derinimo ir politikos atnaujinimo apribojimų taikymą, ypač nestandartinėje, dinamiškai besikeičiančioje aplinkoje, šis tyrimas rodo didesnį prisitaikomumą ir mokymosi efektyvumą, lyginant su ankstesniais lyginamaisiais tyrimais. Šie patobulinimai rodo, kad išbandytos metodų konfigūracijos siūlo patikimesnį apibendrinimą ir geresnį politikos veikimą esant realaus galvosūkio tipo žaidimo sudėtingumui.

Rezultatai

Vertinimas apėmė keletą pagrindinių parametrų, kurie buvo tiksliai suderinti, siekiant pasiekti optimalių agento veikimų. Buvo nustatytos geriausios parametrų reikšmės (atsižvelgiant į sukurtas aplinkas) PPO ir SAC metodams. PPO parametrai apėmė „gamma“ (nuolaidos koeficientą) - 0,99, c1 (vertės funkcijos praradimo koeficientą) - 0,5, c2 (entropijos reguliarumo koeficientą) - 0,1, „learning rate“ - 3e-4 ir „initial beta“ (KL bauda) - 0,1, kurios prisidėjo prie stabilaus ir efektyvaus mokymosi. SAC geriausi rezultatai buvo pasiekti naudojant „gamma“ - 0,99, „learning rate“ - 3e–4, „batch size“ (partijos dydis) – 64 ir „batch size“ (pakartojimo buferio dydis) - 1000000, užtikrinant veiksmingą patirties atkūrimą ir patikimus politikos atnaujinimus.

Iš šių rezultatų galima spręsti, kad metodai, pasižymintys prisitaikymu, pvz., SAC su automatiniu temperatūros reguliavimu, geriau tinka lengvesnei (paprastesnei) aplinkai, kuriai reikia dinaminių reakcijų, o metodai su apribojimais, orientuoti į stabilumą, pvz., PPO su patikimumo regiono optimizavimu, puikiai tinka santykinai struktūrizuotoms užduotims. Tokių metrikų kaip atlygis už epizodą, politikos praradimas, vertės praradimas ir entropijos koeficientai įtraukimas leido atlikti išsamų įvertinimą, išryškinant metodų stabilumo ir pritaikomumo kompromisus. Taip pat, remiantis atliktais statistiniais testais („Kruskal-Wallis“ ir „Dunno“) abiejose aplinkose, galima daryti aiškią išvadą apie sustiprinto mokymosi metodų efektyvumą skirtingo sudėtingumo sąlygomis. Pirmojoje, paprastesnėje aplinkoje, geriausią rezultatą parodė SAC su automatiniu temperatūros reguliavimu, kuris pasižymėjo aukščiausiu vidutiniu rangų įvertinimu ir reikšmingais pranašumais prieš kitus metodus. Tuo tarpu antrojoje, sudėtingesnėje aplinkoje, PPO su pasitikėjimo regiono optimizavimu rodė didesnį našumą, pasiekęs statistiškai reikšmingus laimėjimus ir aukščiausią rangą. Tuo svarbu pabrėžti, kaip svarbu pasirinkti sustiprinto mokymosi metodą, pagrįstą konkrečiomis aplinkos ypatybėmis, nes nė vienas metodas nuosekliai nepralenkė kitų abiejose aplinkose. Tai suteikia vertingų įžvalgų apie šiuolaikinių sustiprinimo mokymosi metodų taikymą, ypač atliekant užduotis, kurioms reikia tikslumo arba lankstumo, ir sudaro pagrindą tolesniam mišrių metodų tyrinėjimui, kuris galėtų sujungti kelių metodų stipriąsias puses, norint pasiekti geresnio našumo įvairiose srityse ir aplinkose.

Išvados

Remiantis darbo eiga ir tyrimo rezultatais, galima teigti, kad darbo tikslą pavyko pasiekti, nustatant, kuris patobulinto sustiprinto mokymosi metodas yra tinkamiausias įvairaus sudėtingumo aplinkoms ir gali būti pritaikomas galvosūkių tipo žaidimams. Gautos darbo išvados:

1. Nustatyta, kad agento elgesys ir mokymosi sėkmė yra labai glaudžiai susiję su aplinkos sudėtingumu, o optimalus metodas skirtingose sąlygose gali skirtis – tai paneigia prielaidą apie vieno „geriausio“ metodo universalumą.
2. Pagal gautus tyrimų rezultatus buvo nustatyta, kad SAC metodas su automatiniu temperatūros reguliavimu yra tinkamiausias lengvesnėse žaidimų aplinkose, o tuo tarpu PPO metodas su patikimumo regiono optimizavimu - tinkamesnis sudėtingesnėse galvosūkių žaidimų aplinkose (žr. 4.3 poskyrį).
3. Pastebėta, kad entropijos koeficientas „alpha“, jeigu yra reguliuojamas automatiškai, leidžia agentui dinamiškai prisitaikyti prie tyrinėjimo ir eksploatavimo balanso, tačiau tokia adaptacija yra efektyvi tik paprastesnėse, dažniau apdovanojančiose aplinkose. Sudėtingesnėse aplinkose automatinė adaptacija nebuvo pranašesnė už fiksuotą ar griežtai kontroliuojamą politikos atnaujinimo mechanizmą.
4. Remiantis 2-5 lentelėmis galima nustatyti, kad „gamma“ ir „learning rate“ parametrai daro didžiausią įtaką mokymosi stabilumui, o „target KL“ ir „alpha“ - mažiausią įtaką.

Šaltiniai

[AIc19] AIcrowd. *Unity Obstacle Tower Challenge* [https://www.aicrowd.com/challenges/unity-obstacle-tower-challenge]. 2019.

[Aur19] AurynSky. *Desert - Low Poly Toon Battle Arena / Tower Defense Pack*. 2019. Prieiga per internetą: https://assetstore.unity.com/packages/3d/environments/desert-low-poly-toon-battle-arena-tower-defense-pack-124507. Unity Asset Store paketas.

[BCP+16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba. *OpenAI Gym*. 2016.

[CCK+22] L. V. R. Cavadas, E. Clua, T. C. Kohwalter, S. A. Melo. Training human-like bots with Imitation Learning based on provenance data. Iš: *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2022, p. 1–6. Prieiga per internetą: https://doi.org/10.1109/SBGAMES56371.2022.9961077.

[Con99] W. Conover. *Practical Nonparametric Statistics*. Wiley, 1999. Wiley Series in Probability and Statistics. ISBN 9780471160687. Prieiga per internetą: https://books.google.lt/books?id=n_39DwAAQBAJ.

[FHI+18] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning*. 2018, tomas 11, numeris 3-4, p. 219–354. ISSN 1935-8237. Prieiga per internetą: https://doi.org/10.1561/2200000071.

[GVC+21] G. Gomes, C. A. Vidal, J. B. Cavalcante-Neto, Y. L. B. Nogueira. Two Level Control of Non-Player Characters for Navigation in 3D Games Scenes: A Deep Reinforcement Learning Approach. Iš: *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2021, p. 182–190. Prieiga per internetą: https://doi.org/10.1109/SBGames54170.2021.00030.

[HDW+23] L. van Hezewijk, N. Dellaert, T. V. Woensel, N. Gademann. Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem. *International Journal of Production Research*. 2023, tomas 61, numeris 6, p. 1955–1978. Prieiga per internetą: https://doi.org/10.1080/00207543.2022.205.

[HZH+19] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker ir kiti. *Soft Actor-Critic Algorithms and Applications*. 2019. Prieiga per internetą: https://arxiv.org/abs/1812.05905.

[YAT+20] Y. Yu, K. Adu, N. Tashi, P. Anokye, X. Wang, M. A. Ayidzoe. RMAF: Relu-Memristor-Like Activation Function for Deep Learning. *IEEE Access*. 2020, tomas 8, p. 72727–72741. Prieiga per internetą: https://api.semanticscholar.org/CorpusID:218473628.

[JBT+20] A. Juliani, V.-P. Berges, E. Teng, A. Cohen ir kiti. *Unity: A General Platform for Intelligent Agents*. 2020.

[JBV+18] A. Juliani, V. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, D. Lange. Unity: A General Platform for Intelligent Agents. *CoRR*. 2018, tomas abs/1809.02627. Prieiga per internetą: http://arxiv.org/abs/1809.02627.

[JKB+19] A. Juliani, A. Khalifa, V. Berges, J. Harper, H. Henry, A. Crespi, J. Togelius, D. Lange. Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. *CoRR*. 2019, tomas abs/1902.01378. Prieiga per internetą: http://arxiv.org/abs/1902.01378.

[KB20] J. T. Kristensen, P. Burelli. Strategies for Using Proximal Policy Optimization in Mobile Puzzle Games. Iš: *International Conference on the Foundations of Digital Games*. ACM, 2020. FDG '20. Prieiga per internetą: https://doi.org/10.1145/3402942.3402944.

[KPH20] A. Kanervisto, J. Pussinen, V. Hautamäki. *Benchmarking End-to-End Behavioural Cloning on Video Games*. 2020.

[LWT+20] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch. *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. 2020.

[Ma+19] X. Ma ir kiti. Extending a Game Engine with Machine Learning and Artificial Intelligence. 2019.

[MBM+16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning*. 2016.

[MKS+15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu ir kiti. Human-level control through deep reinforcement learning. *Nature*. 2015, tomas 518, p. 529–533. Prieiga per internetą: https://api.semanticscholar.org/CorpusID:205242740.

[PWY+21] Y. Pu, S. Wang, R. Yang, X. Yao, B. Li. *Decomposed Soft Actor-Critic Method for Cooperative Multi-Agent Reinforcement Learning*. 2021.

[Res23] Z. M. Research. *PC Games Market Size, Share, Growth, Trends, and Forecast 2030* [https:https://www.zionmarketresearch.com/report/pc-games-market]. 2023.

[SLG+17] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki ir kiti. *Value-Decomposition Networks For Cooperative Multi-Agent Learning*. 2017.

[SLM+17] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel. *Trust Region Policy Optimization*. 2017.

[SWD+17a] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. *Proximal Policy Optimization Algorithms*. 2017.

[SWD+17b] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. *Proximal Policy Optimization Algorithms*. 2017. Prieiga per internetą: `https://arxiv.org/abs/1707.06347`.

[TET12] E. Todorov, T. Erez, Y. Tassa. MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, p. 5026–5033. Prieiga per internetą: `https://api.semanticscholar.org/CorpusID:5230692`.

[TYL+24] X. Tang, Y. Yang, T. Liu, X. Lin, K. Yang, S. Li. Path Planning and Tracking Control for Parking via Soft Actor-Critic Under Non-Ideal Scenarios. *IEEE/CAA Journal of Automatica Sinica*. 2024, tomas 11, numeris 1, p. 181–195. Prieiga per internetą: `https://doi.org/10.1109/JAS.2023.123975`.

[WHT+19a] Y. Wang, H. He, X. Tan, Y. Gan. *Trust Region-Guided Proximal Policy Optimization*. 2019. Prieiga per internetą: `https://arxiv.org/abs/1901.10314`.

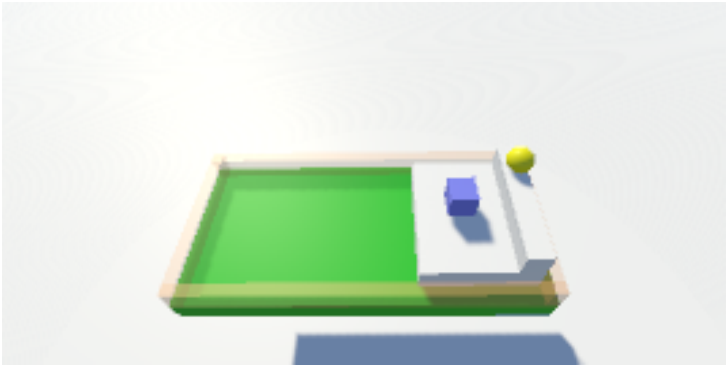
[WHT+19b] Y. Wang, H. He, X. Tan, Y. Gan. Trust Region-Guided Proximal Policy Optimization. Iš: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (sudarytojai). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, tomas 32. Prieiga per internetą: `https://proceedings.neurips.cc/paper_files/paper/2019/file/a666587afda6e89aec274a3657558a27-Paper.pdf`.

[WN20] Y. Wang, T. Ni. *Meta-SAC: Auto-tune the Entropy Temperature of Soft Actor-Critic via Metagradient*. 2020. Prieiga per internetą: `https://arxiv.org/abs/2007.01932`.

[XXY+20] Z. Xiao, N. Xie, G. Yang, Z. Du. Fast-PPO: Proximal Policy Optimization with Optimal Baseline Method. Iš: *2020 IEEE International Conference on Progress in Informatics and Computing (PIC)*. 2020, p. 22–29. Prieiga per internetą: `https://doi.org/10.1109/PIC50277.2020.9350833`.

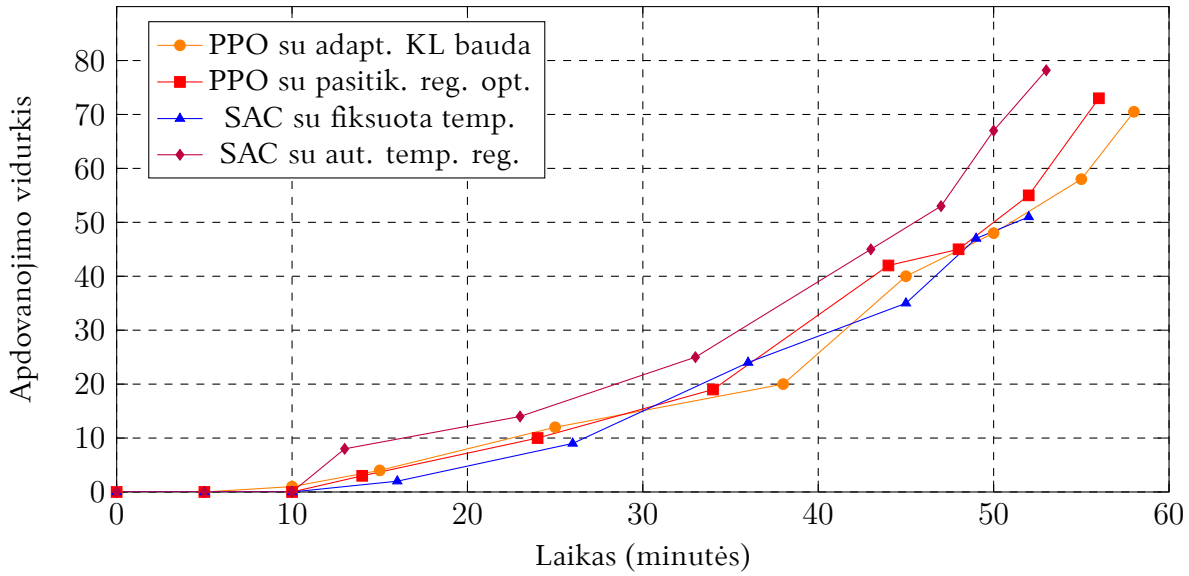
[Zar10] J. Zar. *Biostatistical Analysis*. Prentice Hall, 2010. ISBN 9780131008465. Prieiga per internetą: `https://books.google.lt/books?id=LCRFAQAAIAAJ`.

Priedai

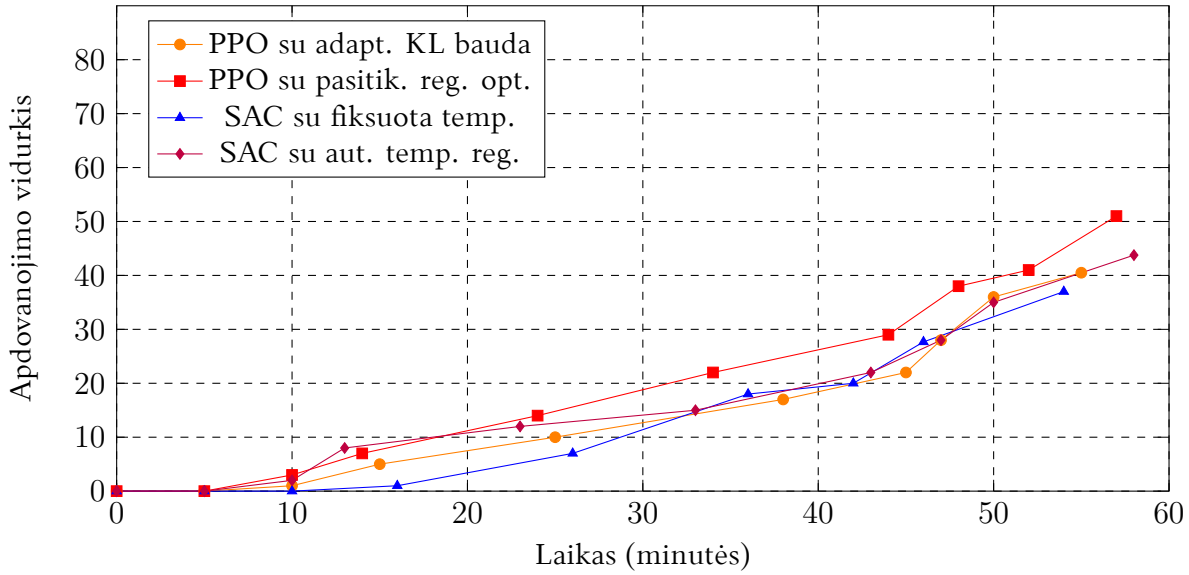


1 pav. Pirmoji mokymosi aplinka

2 pav. Antroji mokymosi aplinka



3 pav. Modelių apdovanojimų vidurkis viso mokymosi metu pirmojoje aplinkoje



4 pav. Modelių apdovanojimų vidurkis viso mokymosi metu antrojoje aplinkoje