# Vairavimo maršruto skaičiavimo, grindžiamo skatinamuoju mokymusi, vizualios aplinkos kūrimas

**Oskaras Klimašauskas, Gintautas Dzemyda**

Vilniaus universitetas, Duomenų mokslo ir skaitmeninių technologijų institutas,
Akademijos g. 4, LT-08412 Vilnius,
*oskaras.klimasauskas@mif.vu.lt*

**Santrauka.** Straipsnyje yra sprendžiamas optimalaus maršruto kelių tinkle paieškos uždavinys. Uždavinys yra modelinis, nes kelių tinklas pasirinktas stačiakampis su vienodomis tiesiomis atkarpomis, o kai kuriose sankryžose yra veikiantis šviesoforas. Uždavinys sprendžiamas naudojantis skatinamojo mokymosi algoritmais. Straipsnyje siekiama palyginti skirtingus skatinamojo mokymosi algoritmus, o taip pat sukurti vizualią aplinką, leidžiančią stebėti skatinamojo mokymosi procesą. Vizuali aplinka yra sudaryta iš automobilio, kelių ir šviesoforų tinklo, bei galutinio finišo. Mokymasis vyksta siekiant minimizuoti pravažiuotų atkarpų skaičių. Algoritmai, sunaudojantys mažiausią tokių atliktų žingsnių skaičių ir tuo būdu randantys sprendimą greičiausiai, yra geriausi. Tyrime buvo naudojami keturi skatinamojo mokymosi algoritmai: *Q-learning, Sarsa, Sarsa(λ), Actor-critic*. Pasiūlytos realizacijos, labiausiai tinkančios sprendžiamam uždaviniui. Aplinka naudinga susipažįstantiems su skatinamuoju mokymusi ir jo principais. Straipsnyje pateikiama nuoroda į aplinkos programos kodą ir instrukcijos, kaip ja pasinaudoti. Tai turėtų išplėsti skatinamojo mokymosi taikymus.

**Raktiniai žodžiai:** Mašininis mokymasis, Skatinamasis mokymasis, Maršruto paieška, Demonstracinė aplinka.

## 1 Įvadas

Skatinimasis mokymasis yra mašininio mokymo atšaka, kur egzistuoja tam tikras agentas, kuris mokosi spręsti sudėtingą uždavinį, atlikdamas elementarius veiksmus, bet siekdamas maksimizuoti kažkokį ilgalaikį atlygį ar pasiekimą. Skatinamasis mokymasis pagrįstas agento ir aplinkos sąveika: agentas stebi aplinką, pagal ją atlieka veiksmą, gauna atlygį priklausomai nuo jo veiksmų. Šių atlygių pagrindu agentas tobulina savo veiksmų strategiją.


DOI: https://doi.org/10.15388/LMITT.2024.7

Pagrindinis tikslas yra išmokti optimalią veiksmų seką, kuri leistų pasiekti geriausią galimą rezultatą konkrečioje užduotyje. Matematiškai apskaičiuoti geriausią rezultatą dažnai arba labai sunku, arba užtruktų labai daug laiko, arba visiškai neįmanoma. Skatinamasis mokymasis yra vienas iš būdų, kad surasti apytiksliai optimalią veiksmų seką.

Šiame straipsnyje nagrinėjamas uždavinys susijęs su optimalaus maršruto radimu modelinėje aplinkoje, kurioje kelių tinklas yra pavaizduotas kaip stačiakampė tinklas, sudarytas iš vienodų atstumų segmentų ir įrengtų šviesoforų tam tikrose sankryžose. Siekiant išspręsti šį uždavinį, straipsnyje taikomi ir analizuojami įvairūs skatinamojo mokymosi algoritmai: *Q-learning* [1]*, Sarsa* [2]*, Sarsa(λ)* [3] ir *Actor-critic* [4]. Šie metodai pasirinkti todėl, kad jų veikimas grindžiamas skirtingais principais arba bendrumu. Šių metodų efektyvumas lyginamas, kad būtų nustatytas efektyviausias algoritmas optimalaus maršruto paieškai. Tyrimas leidžia ne tik identifikuoti geriausias praktikas optimaliems maršrutams rasti, bet ir suteikia galimybę giliau suprasti, kaip skirtingi skatinamojo mokymosi algoritmai veikia.

Darbo tikslas yra palyginti skirtingus skatinamojo mokymosi algoritmus, o taip pat sukurti vizualią aplinką, leidžiančią stebėti skatinamojo mokymosi procesą. Tokia stebėjimo galimybė yra geras būdas pažinti skatinamąjį mokymąsi, jo veikimą.

## 2 Sprendžiamo uždavinio formuluotė

Uždavinys sudarytas iš prieš tai minėto kelių tinklo bei šviesoforų, kurie turi 2 stadijas: žalia ir raudwona. Uždavinys turi *n* įvažiavimų į kelių tinklą, į vieną kurių agentas (mūsų atveju tai vairuotojas) atsitiktinai įvažiuoja, ir *m* išvažiavimų, į bet kurį patekti yra to agento tikslas. Vairuotojas turi penkis judesių pasirinkimus: judėti į kairę, dešinę, žemyn, aukštyn, nejudėti. Jo galimi judesiai yra tose kryptyse, kuriose egzistuoja kelias ir jame nėra degančio raudono šviesoforo, nejudėti gali pasirinkti bet kokioj situacijoj.

Įsiveskime laiko sąvoką. Mokymosi laiką žymėsime *t*. Pradiniu momentu $t = 0$. Vienas laiko žingsnis yra sugaištamas kiekvienam agento pasirinktam veiksmui, t. y. po atlikto veiksmo *t* padidėja vienetu. Šviesoforų spalvos keičiasi (pasidaro iš žalios į raudoną, ar iš raudonos į žalią) sinchroniškai, kas *k* laiko žingsnių, t. y. kas *k* agento pasirinktų judesių.

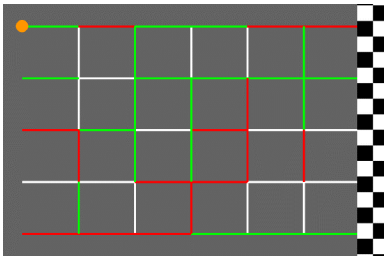
Agento dabartinė būsena yra jo koordinatės ir šviesoforų stadija. Kadangi šviesoforai keičiasi visi tuo pačiu metu, jų stadija globaliai yra reprezen-

tuojama kaip 1 arba -1, kur stadijos reikšmė pasikeičia kas *k* laiko žingsnių. Agento būseną momentu *t* galime aprašyti kaip (*x*, *y*, šviesoforų stadija), čia (*x, y*) yra sankryžos, kurioje randasi automobilis, koordinatės.

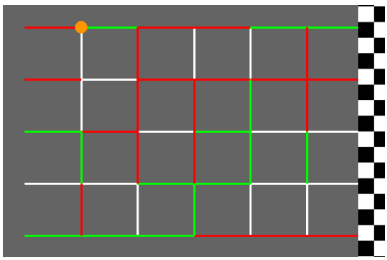
Skatinamajame mokymesi agentas gauna atlygį už sėkmę, ir yra baudžiamas už nesėkmę. Mūsų atveju agentas gauna +1 atlygį už pasiektą finišą ir -1 atlygį už paeitą žingsnį – kuo ilgiau agentas eis iki finišo, tuo labiau jis bus nubaustas. Siekiant gauti kuo didesnį atlygį, optimalus maršrutas turės mažiausiai panaudotą žingsnių skaičių. Žinoma, mūsų atveju visada atlygis bus neigiamas.

## 3 Aplinkos aprašymas

Vizuali aplinka yra sudaryta iš automobilio, kelių ir šviesoforų tinklo, bei galutinio finišo. Mokymasis vyksta siekiant minimizuoti pravažiuotų atkarpų skaičių. Numatyta galimybė pasirinkti skirtingus skatinamojo mokymosi algoritmus šiam uždaviniui spręsti. Algoritmai, sunaudojantys mažiausią tokių atliktų žingsnių skaičių ir tuo būdu randantys sprendimą greičiausiai, yra geriausi. Tyrime buvo naudojami keturi skatinamojo mokymosi algoritmai: *Q-learning, Sarsa, Sarsa(λ), Actor-critic*. Sukurta aplinka 1 pav., iliustruojanti skatinamojo mokymosi procesą, yra tinklas kelių ir šviesoforų, kuriame yra penkios pradinės (starto) būsenos kairėje, kurios atsitiktinai parenkamos agentui momentu $t = 0$, ir penkios finišo būsenos dešinėje, vieną kurių pasiekus pasibaigia simuliacija. Šviesoforų išsidėstymas yra atsitiktinai sugeneruojamas ant kelių tinklo. Šviesoforų atskirai nepaišysime. Sankryža, kur galimo judesio atkarpa yra nuspalvinta baltai reškia, kad ta kryptimi judėjimas nėra reguliuojamas šviesoforu ir ten galima pravažiuoti bet kokiu atveju. Ten kur žalia arba raudona, reiškia kad yra šviesoforas, kuris persijungia kas $k=1$ laiko momentų. Pradinės būsenos irgi suprantamos kaip sankryžos su galimu judėjimu tik į priekį ir tos sankryžos irgi gali būti reguliuojamos šviesoforu. Sankryžoje galima judėti ten, kur „dega“ žalia šviesa arba kur nėra eismo reguliavimo šviesoforu. Agentas turi penkis judesių pasirinkimus: judėti į kairę, dešinę, žemyn, aukštyn, nejudėti. Agento galimi judesiai yra ten, kur egzistuoja kelias ir jame nėra degančio raudono šviesoforo, o nejudėti gali pasirinkti bet kokioj situacijoj. 2 pav. atveju galimi judesiai šiuo atveju yra judėti į dešinę arba palaukti bent vieną laiko žingsnį vietoje. 1 pav. atveju jo galimi judesiai yra į dešinę, žemyn, nejudėti. Veiksmą jis pasirenka pagal naudojamo algoritmo apibrėžtas taisykles. Šviesoforų stadija keičiasi (pasidaro iš žalios į raudona, ar iš raudonos į žalią) kas kiekvieną laiko žings-



**1 pav.** Aplinkos stadija antrame laiko žingsyje



**2 pav.** Aplinkos pradinė stadija

nį. Šiuo atveju visi šviesoforai persijungs iš vienos stadijos į kitą priklausomai nuo *t* tokiu būdu: $-1^t$, kur *t* yra dabartinis laiko žingsnis.

## 4 Pritaikyti skatinamojo mokymosi algoritmai

Šiame straipsnyje keturi skirtingi skatinamojo mokymosi algoritmai buvo naudojami geriausiam maršrutui surasti. Agentas yra baudžiamas už kiekvieną praeitą žingsnį – kuo ilgiau eina, tuo daugiau prisirenka baudų. Agento tikslas tampa pasiekti bet kurį išėjimo tašką per kuo mažiau žingsnių.

Tyrinėjami buvo *off-policy, on-policy, eligibilty trace, policy gradient* tipo algoritmai:

1. *Q-learning (off-policy)*
2. *Sarsa (on-policy)*
3. *Sarsa(λ) (eligibility trace)*
4. *Actor-critic (policy gradient)*

Bendru atveju skatinamojo mokymosi algoritmuose disponuojama su [5]:

- Būsenų rinkiniu. Būsenos apima visas galimas agento vietas aplinkoje.
- Veiksmų rinkiniu. Veiksmai apima visus galimus veiksmus, kuriuos agentas gali paimti aplinkoje.
- Apdovanojimais/baudomis. Atlygio vertė priklauso nuo agento būsenos.
- Nuolaidos faktoriumi. Nuolaida nusako kiek vertinam ilgalaikius atlygius lyginant su trumpalaikiais.
- Elgesiu (*Policy*). Elgesys nusakomas veiksmų pasirinkimo taisykle, kuri apibrėžia tikimybes, su kuriomis agentas pasirenka kiekvieną galimą veiksmą.

**Q-learning**

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma\, Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (1)$$

*Q-learning* yra *off-policy* metodas, jis sukuria sąryšių lentelę *Q* tarp visų būsenų *S* ir veiksmų *A* porų, vadinamą *Q-table*. Reikšmė lentelėje yra numatomas atlygis pasirenkant konkretų veiksmą esamoje būsenoje, kur $Q(S_t, A_t)$ yra dabartinio laiko žingsnio *t* ir jame pasirinkto veiksmo sąryšio reikšmė. *Q-learning* optimizuoja *off-policy control*, tai reiškia, kad jis būsenoj $S_t$ pasirenka veiksmą $A_t$ pagal veiksmų pasirinkimo taisyklę (*behaviour policy*), tačiau atnaujina matricos *Q* reikšmes pagal tikslo siekimo taisyklę (*target policy*). Šiuo atveju tikslo siekimo taisyklė (*target policy*) yra godi (*greedy*), kur pasirenka veiksmą, turintį didžiausią reikšmę būsenoje, o veiksmų pasirinkimo taisyklė (*behaviour policy*) yra *ε*-godi (*ε-greedy*), kur renkasi veiksmą su didžiausia reikšme, bet turi tikimybę *ε* atsitiktinai pasirinkti bet kokį veiksmą (iš galimų veiksmų toje būsenoje). Atnaujinimo formulėje (1) dabartinės būsenos $S_t$ ir veiksmo $A_t$ poros reikšmė $Q(S_t, A_t)$ yra atnaujinama remiantis sekančios būsenos (būsena, kurioje atsirandame atlikus veiksmą $A_t$) ir godaus (*greedy*) veiksmo pora $\max_a Q(S_{t+1}, a)$, γ nurodo nuolaidos faktorių, t.y. kiek norime atsižvelgti į sekančių žingsnių reikšmes. $R_{t+1}$ nusako būsenos po veiksmo $A_t$ atlygį, *α* yra žingsnio dydis dar kitaip vadinamas mokymosi greičiu.

**Sarsa**

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma\, Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (2)$$

*Sarsa*, kitaip nei *Q-learning* yra *on-policy* metodas. Jis taip pat atnaujina tas pačias būsenų *S* ir veiksmų *A* poras $Q(S_t, A_t)$ reikšmes, tačiau tai daro optimizuojant agento veiksmų pasirinkimo taisyklę (*behaviour policy*). Vietoj to, kad atnaujintų *Q* reikšmes pagal godžiai (*greedy*) pasirinktą veiksmą, jas atnaujina pagal sekantį veiksmą $A_{t+1}$ (2), pasirinktą *ε*-godžiai (*ε-greedy*). Kol *Q-learning* mokinasi idealius veiksmus nepaisant to, kad gali atsitiktinai pasirinkti kitą, *Sarsa* tiesiogiai mokinasi pagal tuos veiksmus kuriuos pasirenka.

**Sarsa(λ)**

*Sarsa(λ)* yra *Sarsa* algoritmo išplėtimas, kuriame naudojamas parametras $\lambda$ ($0 \le \lambda \le 1$), leidžiantis kontroliuoti, kiek ankstesnės patirtys turi įtakos būsenos ir veiksmo poros reikšmės atnaujinimui. Algoritmas naudoja žymeklių

masyvą (*eligibility traces*) $z_t$, kuriame kaupiama, kaip seniai viena ar kita būsena buvo aplankyta. Žymeklio reikšmė nustatoma 1 naujausioje būsenoje, ir sumažėja parametro $\lambda$ dydžiu kas žingsnį, tokiu būdu agentas gauna žymeklių „uodegas" (*traces*), kur ankstesnės patirties reikšmė mažėja, kuo ilgiau būsena neaplankyta. Kai $\lambda = 0$, *Sarsa(λ)* tampa paprastu *Sarsa* algoritmu, kuris žiūri tik į sekančio veiksmo numatomą atlygį, o jei 1, tada tampa „*Monte Carlo*" metodu [6]. *Q* matricos elementų (įtakos būsenos ir veiksmo poros reikšmės) perskaičiavimas vykdomas naudojant *Q* matricos aproksimaciją [6], [7], kur esant didelei aplinkai, jos būsenai reprezentuoti naudojamas tam tikras ypatybių vektorius. Kai aplinka yra pakankamai maža, kad galima į būsenų ir veiksmų porų reikšmes žiūrėti individualiai, tokiu atveju siūlome $z_t$ naudoti kaip lentelę, kur kaupiamos visos žymeklių reikšmės (3). Dabartinei būsenai ir veiksmui $z(S_t, A_t)$ priskiriama reikšmė 1, o visos kitos reikšmės pamažėja $\lambda$ dydžiu. Čia matrica $Q_t$ yra visos matricos *Q* reikšmės *t* laiko momentu. Algoritmas veiksmus taip pat rinkosi $\varepsilon$-godžiai (*ε-greedy policy*).

$$\begin{aligned} &z_t \leftarrow \gamma\lambda z_{t-1} \\ &z(S_t, A_t) \leftarrow 1 \\ &Q_t \leftarrow Q_t + \alpha\big[(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))\big]z_t \end{aligned} \tag{3}$$

Mūsų tyrime algoritmai *Q-learning*, *Sarsa*, *Sarsa(λ)* naudojo tokius parametrus: mokymosi žingsnio dydį $\alpha = 0.01$; nuolaidos faktorių $\gamma = 1$, kad pilnai atsižvelgtų į ateities atlygius; aplinkos tyrinėjimo vertę $\varepsilon = 0{,}1$; pradinės *Q* vertes $Q(s, a) = -100$, visiems $s \in S$, $a \in A(s)$. Šios pradinės *Q* reikšmės buvo pasirinktos todėl, kad aplinkoje žingsnių atlygis yra neigiamas. Jei $Q(s, a)$ būtų inicijuota nulinė, tada agentui ilgą laiką aukščiausios vertės veiksmas būtų mažiausiai tyrinėtas, nors ir blogas, veiksmas. *Sarsa(λ)* taip pat naudojo $\lambda = 0{,}95$ parametrą žymekliams, nes tai rodo, kad ankstesnės patirtys turi ganėtinai didelę įtaką, bet vis tiek pamažu į jas mažiau atsižvelgiama.

**Actor-critic**

*Actor-critic* yra *policy* gradient metodas. Vietoj to, kad išmoktų veiksmų vertes ir tada pagal kažkokią veiksmų pasirinkimo taisyklę juos pasirinktų, jis tiesiogiai išmoksta elgseną gradiento metodu. *Actor-critic* užtat skiriasi nuo paprasto *policy* gradient tuo, kad jis taip pat mokosi būsenos vertę $v(S_t)$, kuri nusako, kaip gerai agentui būti tam tikroje būsenoje, tikslu kritiškai vertinti savo elgsenos atnaujinimą. Kaip ir su *Sarsa(λ),* esant didelei aplinkai Q

matricos elementų (įtakos būsenos ir veiksmo poros reikšmės) perskaičiavimas vykdomas naudojant būsenos vertė $\hat{v}$ aproksimacija [4]. Nedidelės aplinkos atvejui siūlome visas būsenų vertes $v(S_t)$ kaupti atskirai (4). Tam, kad algoritmas įgautų *policy* $\pi$ (veiksmų tikimybių paskirstymą) naudojama *softmax* funkcija. Funkcijai duodama reikšmė yra *policy* parametras $\theta$, kuris įvertina kiek stipriai reikia apsvarstyti veiksmą. Šiuo atveju naudojam skirtingus žingsnio dydžius $\alpha$ būsenai $\alpha^v$ ir *policy* parametrui $\alpha^\theta$. Siekiant greičiau išmokti apytikslią būsenos vertę ir pagal ją geriau kritiškai vertinti savo elgseną, būsenos mokymosi greitį nustatome didesniu. $v(S_{t+1})$ (arba $\hat{v}(S_{t+1}, w)$) yra sekančios būsenos reikšmė.

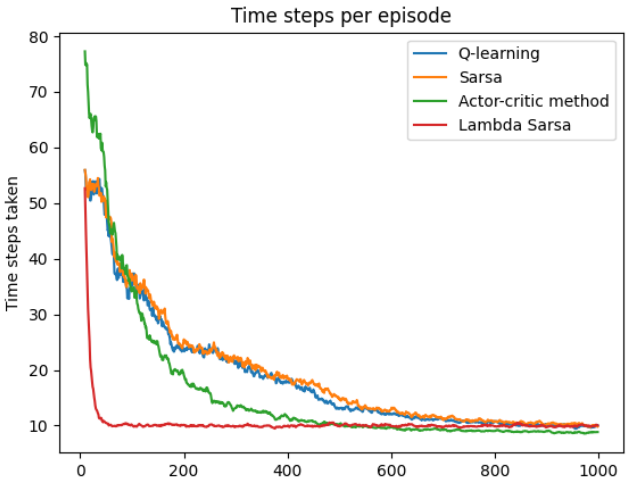
$$\begin{aligned} v(S_t) &\leftarrow v(S_t) + \alpha^v[R_{t+1} + \gamma v(S_{t+1}) - v(S_t)] \\ \theta &\leftarrow \theta + \alpha^\theta[R_{t+1} + \gamma v(S_{t+1}) - v(S_t)]\nabla ln\pi(A|S,\theta) \end{aligned} \tag{4}$$

*Actor-critic* skyrėsi nuo kitų trijų aukščiau pateiktų algoritmų tuo, kad naudojo papildomai $\alpha^v = 0{,}1$ ir $\alpha^\theta = 0{,}005$, kad greičiau galėtų tiksliau kritiškai vertinti savo veiksmus. Vietoj būsenų ir veiksmų porų jis naudojo tik numatomas būsenų vertes, kurios inicializuojamos $v(s) = 0$, visiems $s \in S$. Šios pradinės vertės yra standartinės ir nėra labai svarbios. *Policy* parametrai $\theta$ buvo atsitiktinai sugeneruojami intervale (0, 1) visiems $s \in S$, $a \in A(s)$ kad pradžioj turėtų ne vienodą veiksmų pasirinkimą.

Tyrime lyginame *Q-learning* mokymąsi pagal atskirą tikslo siekimo taisyklę (*target policy*), *Actor-critic* tiesioginį elgsenos mokymąsi bei skirtumą tarp *Sarsa* su žymekliais ir be jų.

## 5 Tyrimo rezultatai

Eksperimentiškai ištirti keturi aukščiau pateikti algoritmai. 3 pav. pateikti duomenys yra 20 mokymosi paleidimų vidurkis, rodantis reikalingų žingsnių skaičių finišui pasiekti, didėjant mokymosi epochų kiekiui. Kaip matome, *Sarsa(λ)* greičiausiai randa optimalų 8 žingsnių kelią ir jį pasiekia apie 60-oj epochoj. Tuo tarpu *Actor-critic* algoritmui prireikia 500 epochų, o *Q-learning* ir paprastam *Sarsa* algoritmui prireikia 800 epochų. Kadangi tiek *Sarsa* algoritmai, tiek *Q-learning* naudoja $\varepsilon$-godų veiksmų pasirinkimą, net ir išmokę optimalų kelią, jie kartais atsitiktinai pasirenka neteisingą veiksmą. Priešingai, *Actor-critic* metodas atnaujina savo elgseną ir palaipsniui mažina aplinkos tyrinėjimą. Tai galima pastebėti nuo 600-osios epochos, kai *Actor-critic* pradeda rečiau rinktis atsitiktinius veiksmus ir artėja prie 8 žingsnių




**3 pav.** Algoritmų palyginimo rezultatai

vidurkio. Paprastas *Sarsa* ir *Q-learning* šiame uždavinyje parodė identiškus rezultatus, reikšmių optimizavimas pagal atskirą elgseną neturėjo didelės įtakos. *Actor-critic* pradeda lėčiausiai, tačiau po 100 epochų aplenkia *Q-learning* ir *Sarsa* ir po 600 epochų aplenkė *Sarsa(λ)*. Tiesiogiai mokydamasis elgseną, šis metodas pasižymi tuo, kad išmoksta lygiaverčius veiksmus ir gali juos pasirinkti su lygia tikimybe. Šioje aplinkoje tai reiškia, kad jis išmoksta, jog tam tikroje situacijoje tiek pat verta važiuoti į viršų, kiek ir į apačią.

## 6 Išvados

Tyrime buvo naudojami keturi skatinamojo mokymosi algoritmai: *Q-learning*, *Sarsa*, *Sarsa(λ)*, *Actor-critic*. Pasiūlytos realizacijos, labiausiai tinkančios sprendžiamam uždaviniui. Remiantis tyrimo rezultatais, galima teigti, kad

1. *Sarsa(λ)* algoritmas buvo efektyviausias randant optimalią maršrutą greitai. Naudodamas ankstesnius savo patyrimus būsenos ir veiksmų poros reikšmių atnaujinimui, jis sugeba daugiau nei aštuonis kartus greičiau išmokti optimalų kelią.
2. *Actor-critic* metodas, nors ir pradeda mokintis lėčiausiai, pasižymėjo tuo, kad gali išmokti lygiaverčius veiksmus bei sumažina būsenų tyrinėjimą laikui einant.

3. *Q-learning* ir paprastas *Sarsa* reikalauja mažiau skaičiavimų ir juos lengviau įgyvendinti, tačiau jie nepasirodo tiek įspūdingai kiek *Sarsa*, jei *Sarsa* naudoja *eligibilty traces*.

Sukurta vizuali aplinka yra sudaryta iš automobilio, kelių ir šviesoforų tinklo, bei galutinio finišo. Ji leidžia stebėti agento mokymosi procesą sulėtintame režime. Mokymasis vyksta siekiant minimizuoti pravažiuotų atkarpų skaičių. Numatyta galimybė pasirinkti skirtingus skatinamojo mokymosi algoritmus šiam uždaviniui spręsti. Aplinka naudinga susipažįstantiems su skatinamuoju mokymusi ir jo principais. Straipsnyje pateikiama nuoroda į aplinkos programos kodą ir instrukcijos, kaip ja pasinaudoti. Tai turėtų išplėsti skatinamojo mokymosi taikymus. Aplinką galima išbandyti ir stebėti, kaip vyksta skatinamojo mokymosi procesas naudojantis šia nuoroda Nacionaliniame atviros prieigos mokslinių tyrimų duomenų archyve MIDAS [8].

## Literatūra


[1] C. Watkins, P. Dayan, Technical Note: Q-Learning, *Machine Learning*, 8, 279-292., 1992.

[2] G. A. Rummery, M. Niranjan, *Online Q-Learning using Connectionist Systems*, Technical Report CUED/F-INFENG/TR 166, 1994.

[3] R. S. Sutton, H. V. Seijen, M. C. Machado, P. M. Pilarski, A. R. Mahmood, *True Online Temporal-Difference Learning*, Journal of Machine Learning Research (JMLR), 17(145):1-40, 2016.

[4] R. S. Sutton, M. White, T. Degris, *Off-Policy Actor-Critic*, https://doi.org/10.48550/arXiv.1205.4839, 2012.

[5] A. Daranda, G. Dzemyda, Reinforcement learning strategies for vessel navigation *Integrated Computer-Aided Engineering*, 30 (1), 53-66, 2023.

[6] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction. Second edition, MIT Press, 281, 303-305, 2018.

[7] H. v. Hasselt, DeepMind x UCL RL Lecture Series, 2021. https://www.youtube.com/watch?v=TCCjZe0y4Qc&list=PLqYmG7hTraZDVH599EItlEWsUOsJbAodm

[8] O. Klimašauskas, G. Dzemyda, „Vairavimo maršruto skaičiavimo, grindžiamo skatinamuoju mokymusi, vizualios aplinkos kūrimas," Nacionalinis atviros prieigos mokslinių tyrimų duomenų archyvas (MIDAS), 2024. https://www.doi.org/10.18279/MIDAS.RLmokymasis.250217