

Vilniaus universitetas
Fizikos fakultetas
Dirbtinis Intelektas

ZUIKIO KLAJONĖS

Teorinė fizika ir astrofizika



Ruošė: Paulius Juodsukis
Ieva Gugaitė

Turinys

1	Programa	3
1.1	Idiegimas	3
1.2	Struktūra	3
1.3	Paleidimas	4
2	Veikimas	6
2.1	Pasaulis	6
2.2	Zuikio būsenos	7
2.3	Mokymasis	7
2.4	Sprendimas	8
3	Rezultatai	8
4	Išvados	11

Bibleoteka	Versija	Komanda
tkinter	naujausia	<code>pip install tkinter</code>
pygame	$\leq 1.9.6$	<code>pip install -Iv pygame==1.9.6</code>
pillow	naujausia	<code>pip install pillow</code>

1 lentelė. Išorinės bibleotekos

1 Programa

1.1 Idiegimas

Projektas naudoja tris išorines Python bibleotekas: tkinter, pillow ir pygame. Žiurėkite 1. Pygame bibleotekoje naudojamas funkcionalumas kuris leidžia sujungti šių dviejų bibleotekų veikimą viename lange. Naujausiose pygame atnaujinimuose jis yra pašalintas, todėl naudojama truputį senesnė 1.9.6 versija. Pillow bibleoteka naudojama lengvam paveiksleliu atvaizdavime vartotojo grafinės sasajos elementuose.

1.2 Struktūra

Projektas sudarytas iš eilės Python failų:

- adventure.py
- configurations.py
- program.py
- qsolver.py
- sprites.py
- utils.py
- window.py
- zuikis_state.py

Ir vienos direktoriujos

- pics

Šioje direktoriijoje yra saugoma paveikslėliai skirti zuikio klajonių atvaizdavimui. Programos failai sukirstyti į dvi dalis, pagrindinius, kurie sprendžia uždavinj:configurations, adventure, qsolver, zuikis_state ir pagalbinius: window, utils, sprites ir program. Žiurėkite antrają lentelę.

Konfiguracijų faile yra pradinių sąlygų aprašymas. Klasė Field pilnai aprašo viso klajonių lauko būseną klajonės pradžioje. Joje apsirašo pradinė zuikio vieta, vilkų vietas ir morkų vietas. Taip pat kiek naudingos morkos ir faile aprašyta jų generavimo įpatumai tikimybiniu požiuriu.

Failo pavadinimas	Paskirtis
configurations.py	Konkrečių uždaviniių ir pradinių sąlygų aprašymas
adventure.py	Abstrakcios zuikio orientavimosi pasaulyje sistemos
zuikis_state.py	Zuikio matymo lauko aprašymas
qsoler.py	Uždavinio sprendiklis naudojantis kokybės(Q) funkcija
utils.py	Pagalbinės funkcijos
window.py	Zuikio klajonių vizualizavimo aprašymas
sprites.py	Vizualių elementų aprašymas
program.py	Programos veikimo pavyzdžių leidiklis

2 lentelė. Programos failai ir jų paskirtis

Nuotykių faile yra aprašyta klajonių sistemos dinamika. Action klasėje, arba jos įpėdinyje Action2 klasėje yra aprašytas sistemos atsakas į zuiki ir pačios sistemos evoliucija zuikiui bandant keiliauti. Metodas interactions atlieka pagrindinį vaidmenį. Jis prijima visa sistemos būseną ir norimą zuikio judejimo kryptį. Atlieka sistemos dinamika ir gražina pasikeituses vertes. Šiu klasių objektai taip, pat įsimena paskutinius zuikio būsenas ir jas gražina kai yra pakviečiamas rabbit_vision metodas. Faile taip pat yra story klasė skirta atvaizduoti vienam episodui arba vienai zuikio kelioniui.

Zuikio būsenų faile yra aprašomos zuikio būsenos. Zuikis mato aplink save keturių Manhattan vienetų atstumu. Jo matymo laukas yra sudedamas į šios klasės objektus ir tada šifruojamas(hashing) patogiam naudojimui. Šie objektai taip pat turi galimybę bendrauti su sprendikliu ir jam perduoti reikalingą, papildoma informaciją reikalingą sprendimo metu.

Sprenklio failas sudarytas iš Q funkcijos sprendklio klasės. Šios klasės objektai sprendžia zuikio klajonių uždavinį. Naudodami juos galime išmokyti kokybės Q funkciją ir jos pagalba atlirkti vizualizuotiną zuikio kelionę.

Pagalbinių funkcijų faile yra laiko sekimo funkcijos. Pagrindinama programos faile yra leidiklis skirtas paleisti įvairius programos veikimo pavyzdžius.

Lango faile yra programos dalis skirta vizualizuoti zuikio klajones grafiškai naudojantis klajonių kelio aprašymu. Atvaizdavimas naudoja visualius elementus aprašytus sprites.py faile.

1.3 Paleidimas

Pagrindinis Python failas yra program.py. Ji paleidus automatiškai pasileidžia leidiklio objekto kuris apmoko kokybės funkcija Q ir paleidžia parinktus pavyzdžius. Pagrindiniai nustatymai gali būti keičiami ir randasi leidiklio konstruktoriuje. Funkcijos mokymas yra aprašomas learn metode. Metode launch yra aprašomas pavyzdžių sprendimas ir grafinis atvaizdavimas.

Norint atvaizduoti kaskoki specifinį atveji užtenka pasirinkti viena iš esamų arba susikurti savo pradinių sąlygų(configurations.Field(...)) konfiguracija ir paleisti su ją launch_example metodu. Paleidus programą, ji pradės mokytis ir po kažkurių laiko (~30s) pradės rodyti zuikio klajonių grafinę sąsają. Sąsajoje galima matyti: viso klajonių lauko atvaizdavimą, zuikio būsenos atvaizdavimą, visu pavyzdžio, sprendimo ėjimų skaičių, dabartinį ėjimo numerį, dabartinę energiją, būsenos parinkimo slankiklį, grojimo mygtuką, grojimo greičio slankiklį, zuikio būsenos aprašymą ir galiausiai ējimo į priekį ar atgal mygtukus. Paspaudus ējimo į priekį mygtuką būsena pasikeis į sekančią zui-

kio klajonėje. Būsena atvaizduojama visame lauke ir konkrečioje zuikio būsenoje. Klajonių lauko sienas, zuiki, vilkus ir morkas vaizduoja tam tikros ikonos arba spalvos. Norint paleisti automatini kelionės grojimą užtenka paspausti grojimo mygtuką. Grojimo greiti galima kontroliuoti su tam skirtu slankikliu.

Norint pažiurėti į konkrečias išmoktas kokybės funkcijos būsenas užtenka paleisti launch_q metoda. Atsivers įdentiškas langas rodantis bendrają ir konkrečiaja zuikio būsenas. Šykarta bendroji lauko būsena nesikeičianti ir fyktivi, tačiau galima keisti zuikio būsena ir pamatyti šalia esantį kokybės funkcijos aprašą.

1	2	3
8	Z	4
7	6	5

3 lentelė. Krypčių atvaizdavimas programoje

7	6	5
8	Z	4
1	2	3

4 lentelė. Krypčių atvaizdavimas grafikos lange

2 Veikimas

Aprašome kaip veikia tam tikri programos elementai.

2.1 Pasaulis

Pasaulis kiekvienu momentu yra aprašomas visų elementų: zuikio, vilko, morkų vietomis, vilkų kryptimis ir zuikio energija. Pradinės šių verčių reikšmės yra užrašomos Fields klasėje, tada naujodanties Story klase pasaulio būsena yra keičiama iki kol nepasibaigia zuikio energija. Pati story klasė tik seka dabartines pasaulio koordinates ir naudojasi Actions2 klase atliki bet kokius judėjimus šiame pasaulyje. Actions2 klasė turi metoda interactions kuriam perdavus visas koordinates ir norima zuikio judejimo krypti jis atlieka visus reikiamus veiksmus ir santykių detekcijas.

Kryptis programoje užsirašo aštuonais skaičiais 1, 2, 3, 4, 5, 6, 7, 8 žiurėti lent 3. Šiaurė yra 2, pietus 6 ir t.t. Kadangi grafikos lange ordinatė yra apversta, todėl apsiverčia ir kryptys. Žiurėkite lent. 4

Saveikavimo metode pirmiausia tikrinama ar vilkas ir zuikis yra vienoje vietoje. Jei taip, tada zuikis praranda energijos ir pajuda centro link. Jei ne, bandoma judėti nurodyta kryptimi, nepasi-sekant tik tada jei judama į sieną. Po šių veiksmų atliekami lauko vilkų veiksmai. Pirmiausia visi vilkai patikrina ar jie mato zuiki. Patikrinimas vyksta apskaičiuojant zuikio Manhattan atstuma iki vilko ir jo reletyvų vektorių. Atliekamas skaliarinis produktas šio vektoriaus su vilko krypties vektoriumi ir jei jis yra ≥ 0 ir atstumas yra atitinkamas tada skaitoma, kad vilkas mato zuiki. Jei vilkas mato, tada bando judeti lygiai per 2 langelius jo link. Judėjimo kryptis nustatoma pagal artimiausią kryptį zuikiui naudojant atan2 metodą. Jei ne juda normaliai. Jei atsimuša į sieną pasikeičia jo kryptis.

Galiausia jei zuikis atsidurė ant morkos jis ją suvalgo. Jo energija padidėja nustatyta reikšme ir morka yra sunaikinama ir paddedama kažkur kitur. Morkos generuojamos atsitiktinai pagal tolyginius pasiskirstymą. Svarbiausias parametras generuojant morkas yra jų skaičius kuris nustatomas pagal lauko ploti ir morkos energija:

$$n_{morkos} = \text{round} \left(\frac{d^2}{\pi r^2 M^2} \right)$$

Kur r skaičius mažesnis už 1.

2.2 Zuikio būsenos

Zuikio būsenos yra skaidomos į tris tipus: kai jis nieko nemato, kai jis mato tik morkas ir kai mato/nemato morku ir mato vilkus ir/arba sienas. Pirmuoju atveju pridedama devintoji 9 kryptis kuri atitinka atsitiktinį ejimą į betkuria iš aštuonių krypčių. Antruojų atvejų kryptys yra visiškai pakeičiamos į kelias kryptis kurių skaičius atitinka morkų skaičių. Šios kryptis, pradedant nuo 0 atitinka kryptis iki arčiausios morkos. Tada pirmoji 0 kryptis reiškia ejima link arčiausios, antrojo 1 link antros arčiausios ir taip toliau.

Kadangi zuikiui yra svarbus atstumas iki vilkų arba sienų, jei jis mato nors viena iš šių jo kryptis nepasikeičia.

Zuikio būsenos yra šifruojamos, siekiant jas lengvai atpažinti ir saugoti naudojant python dict() tipo ir siekiant sumažinti galimų būsenų skaičių. Pirmai būsenai yra šifruojama savo tipu t.y. kad zuikis mato nulį visų išorės elementų. Antroji šifruojama morkų krypčių ir atstumu iki jų aibe, be pasikartojimų. Trečiasis tipas irgi saugomas krypčių ir atstumu iki elementų unikalia aibe. Tokiu budu šifruojant dauk būsenų atitinka tą patį šifrą. Tačiau su kiekviena būsena sprendiklis dirba tik vienu laiko momentu, todėl jis kreipiasi į pačią būseną, kad gautu jam reikama informaciją. Pavyzdžiu iuri kuri kryptis ištikro yra arčiausios morkos kryptis. Ši informacija yra saugoma pačios būsenos aprašyme.

2.3 Mokymasis

Sprendiklis naudoja kokybės funkcijos Q metoda išmokti strategijai. Kokybės funkcija yra būsenos s ir veiksmo a porų rinkinys $Q(s, a)$ kiekvienai būsenai s . Tuomet strategija π kiekvienai busenai yra:

$$\pi(s) = \arg \max_a Q(s, a)$$

Pradžioje kokybės funkcija Q inicijuojama į nulius, priklausomai kokios kryptis priklauso tyri-nėjamai būsenai. Kiekviename episode vyksta perjimas iš vienos būsenos s į sekančią būseną s' naudojant veiksmą a . Kiekvieno perėjimo metu kokybės funkcija pakeičiama pagal sąryšį:

$$Q(s, a) = Q(s, a) + \alpha(N(s, a)) \left(R + \gamma \arg \max_{a'} Q(s', a') - Q(s, a) \right)$$

Čia α yra mokymosi sparta, $N(s, a)$ yra apsilankymu saicius, R vertės pokytis einant išvienos būsenos į kita ir γ skaičius. Agentas taip keičia būsenas tam tikra skaičių ejimą. Šis skaičius gali būti skaičius iki klajonės pabaigos(kai pasibaigia zuikio energija) arba nustatytas maksimumas. Tokia, viena kelionė vadinama episodu. Agentas mokosi kokybės funkcijos Q per daudžius episodus. Galiausia vilimasi, kad vertės kiekvienoje būsenos ir veiksmo poroje nusistovės tam tikroje vertėje. Tą užtikrina mokymosi spartos pastovus gesimas:

$$\alpha(N(s, a)) = \frac{N_{cut}}{N_{cut} - 1 + N(s, a)}$$

kur N_{cut} yra išorinis parametras.

Sekančios būsenos yra pasirenkamos naudojant tyrinėjimo išnaudojimo strategija nulemta kiek-vienos būsenos veiksmo aplankymo skaičiaus. Naudojama tyrinėjimo funkcija:

$$f(Q(s, a), N(s, a)) = \begin{cases} R_+ & N(s, a) \leq N_{min} \\ Q(s, a) & N(s, a) > N_{min} \end{cases}$$

čia R_+ ir N_{min} yra išoriniai parametrai. Ši funkcija naudojama nustatyti sekančius veiksmus leidžia ištyrinėti beveik visas būsenas.

2.4 Sprendimas

Sprendikliui išmokus kokybės Q funkcija jis iš jos nusistato strategiją ir iš jos gali bandyti spresti pavyzdžius. Sprendimas vyksta naudojantis strategija ir Story klase, labai panašiai kaip vyko mokymo metu. Tik dabar Q funkcija nebéra mokomasi ir tik naudojama rasta strategija „stumdyti“ zuiki lauke kol nepasibaiks jo klajonė. Kartais vedant zuiki jis nueina į tokia būsena kurios nėra Q funkcijos ir tuo labiau strategijos aprašyme. Tokiu atveju zuikis atlieka atsitiktinį éjimą į betkurią kryptį savo būsenoje.

Kitais atvejais zuikis labai mēgsta įstrigtį į uždarus ciklus eidamas į sieną arba keliaudamas pastoviai per tuos pačius kelis langelius. Tokiu atveju yra sekamos zuikio praeitos būsenos ir tikrinama ar jis neperéjo per ta pačia būsena visai nesenai. Jei peréjo jis vėl atlieka atsitiktini éjimą. Taip zuikis keliauja pusiau atsitiktinai, bet kai pamato morka jo strategija „atgyja“ ir leidžia jam jas susirinkti.

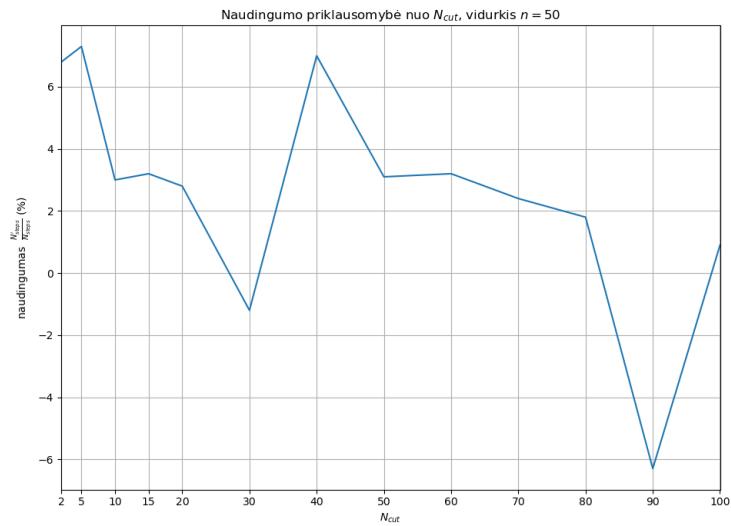
3 Rezultatai

Programoje yra keli pagrindiniai parametrai:

- max_iter - maksimalus iteracijus/episodu skaičius
- max_step - maksimalus žyngsių skaičius episode
- n_cut - N_{cut}
- n_min - N_{min}
- r_plus - R_+
- gamma - γ

Mes pasirinkome max_iter = 400 ir max_step = 500. Parametra n_min pasirinkome 10. Mokome sprendiklį išviso 4 kartus su tuo pačiu lauku, tik pasukti kiekviena karta po 90° neskaitant vilko.

1 pav. N_{cut} sąryšis mokymosi naudingumu $\frac{N'_{path}}{N_{path}}$. Vertės yra kelių vidurkiai su $n = 50$

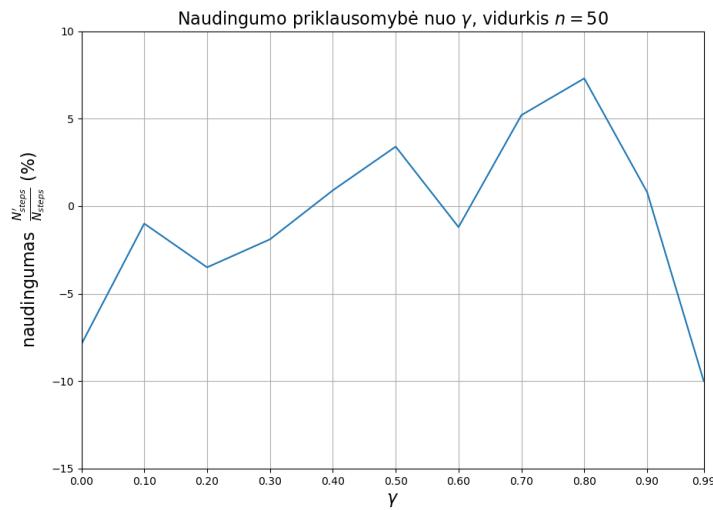


Pirmiausia pabandėme nustatyti N_{cut} vertę, kaitaliodami parametrus. Žiurėti pav. 1. Kaip matosi iš eksperimentų geriausia naudoti vertes $N_{cut} = 5$ arba $N_{cut} = 40$. Mes pasirinkome pirmajį variantą $N_{cut} = 5$. Ateities spėjimo parametras γ yra svarbus nes nuo jo stipriai priklauso mokymosi rezultatai. Jį pakaitaliojome ir gavome geriausia vertę $\gamma = 0.8$. Žiurėti pav. 2

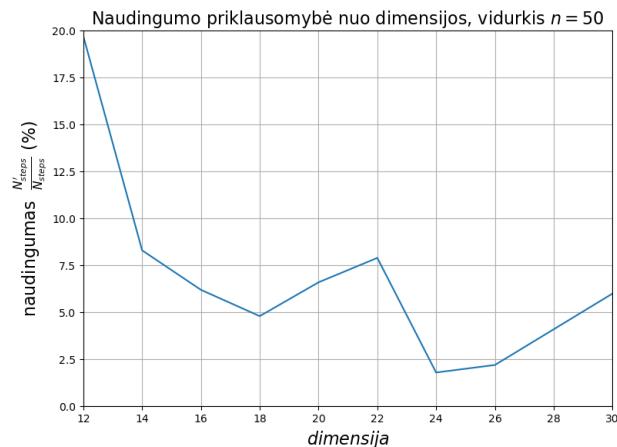
Pabandėme patikrinti priklausomybę nuo lauko dimensijos d , žiurėti pav. 3. Kaip matome, naudingumas mažėja su lauko dydžiu. Tai galima paaiškinti tuom, kad besielgdamas atsitiktinai didele dalį laiko gali savo matymo lauke aptikti daugiau morkų mažesniuose laukuose. Kadangi atsitiktinio éjimo vidurkis yra 0 jam sunku yra nueiti didesnius atstumus atsitiktinai.

Galiausa patikrinome parametra M, žiurėti pav. 4. Matome, kad naudingumas yra ganétinai įvairus. Bet daugiausiai naudos duoda mažesnės vertės. Taip yra dël to, kad nuo parametro M priklauso morkų skaičius. Kuo jis didesnis tuo morkų mažiau(pagal uždavinio sąlygą). Kadangi, vél, zuikiui judant didelius plotus atsitiktinai yra sunku jis negali ieskoti kelių morkų per visa lauką efektyviai.

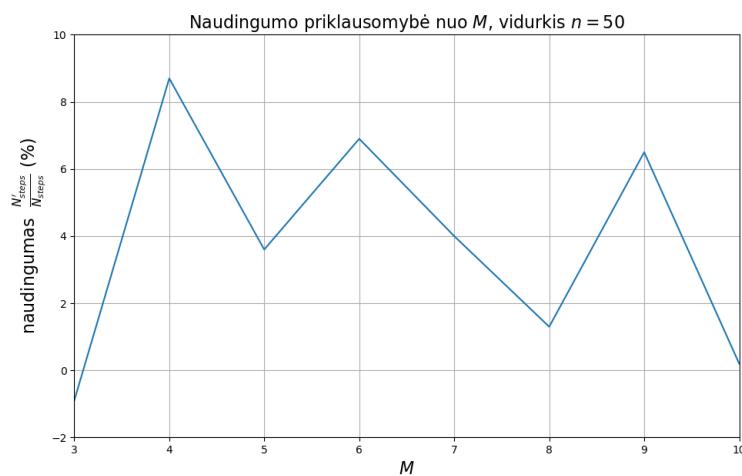
2 pav. γ sąryšis mokymosi naudingumu $\frac{N'_{path}}{N_{path}}$. Vertės yra kelių vidurkiai su $n = 50$



3 pav. d sąryšis mokymosi naudingumu $\frac{N'_{path}}{N_{path}}$. Vertės yra kelių vidurkiai su $n = 50$



4 pav. M sąryšis mokymosi naudingumu $\frac{N'_{path}}{N_{path}}$. Vertės yra kelių vidurkiai su $n = 50$



4 Išvados

- Dalinai atsitiktinai vaikščiojančiam zuikiui geriausia yra mažesnis klajojimo laukas ir mažesnės M vertės. Patobulinus atsitiktinį elementą į geresnį būtų galima pataisyti ši tašką.
- Efektyviam mokymuisi reikia gerai nustatyti hyper parametrus kaip γ ar N_{cut} .
- Esant laiko dimensijai $d = 16$ vienu vilku zuikis susidoroja pakankamai efektyviai, su dviems truputi prasčiau ir su daugiau jo efektyvumas krenta labai stipriai.