LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

POMDP problēmu risināšana, izmantojot vēsturiskus elementus, un tās optimizācija

BAKALAURA DARBS

Autors: **Elvis Egle**
Studenta apliecības Nr.: ee13008
Darba vadītājs: prof. Jānis Zuters

RĪGA 2017

ANOTĀCIJA

Elvja Egles Bakalaura darba „POMDP problēmu risināšana, izmantojot vēsturiskus elementus, un tās optimizācija" ietvaros tika izpētīti daļēji novērojami Markova lēmuma procesu (POMDP) algoritmi. Balstoties uz Jāņa Zutera zinātniski pētniecisko raksta saturu „Sequence Q-Learning: a Memory-based Method Towards Solving POMDP", kurā ir aprakstīta POMDP risināšanas ideja, tika izstrādāts līdzvērtīgs mašīnmācīšanās algoritms, saskaņā ar rakstā sniegto informāciju. Bakalaura darba ietvaros tika pētīta šī Algoritma efektivitāte, kā arī apskatītas un piedāvātas vairākas iespējas tā tālākai pilnveidei. Programmētais darbs tika pievērsts konkrētai problēmai, kurai bija novērojama labāka ātrdarbība ar nelieliem uzlabojumiem algoritma kvalitātē dažās situācijās.

Atslēgas vārdi: mašīnmācīšanās, stimulētā mācīšanās, POMDP, Sequence Q-Learning

ABSTRACT

Elvis Egle’s Bachelor thesis "POMDP problem solving, using historic elements and its optimization" examines the topic of partially observable Markov decision process (POMDP). Based on the research article "Sequence Q-Learning: a memory-based Method towards Solving POMDP" by Janis Zuters, which describes a proposed idea on how to solve POMDP, an equivalent machine learning algorithm was developed, in line with the information contained in the article. Within the bachelor's thesis the algorithm efficiency was evaluated, improved and a number of ways of how it could be perfected were discussed. The program was aimed at a particular problem, which saw an improved operation speed with minor improvements in the quality of the algorithm in some situations.

Key Words: machine learning, reinforcement learning, POMDP, Sequence Q-Learning

Saturs

1. IEVADS

1.1. Tēmas izvēle

Daļējas novērojamības MDP problēmas ir ļoti specifiska nozare mašīnmācīšanā. Tās ietvaros galvenokārt tiek apskatītas situācijas, kurās aģents jeb dators var novērot problēmu tikai daļēji un arī apmācības procesā netiek dota pilna informācija par vidi, kurā aģents atrodas. Tēmas izvēle tika veikta pēc tematiskās atlases, kad tika apsvērti dažādi darba virzieni Mašīnmācīšanās nozarē un tika izrādīta padziļināta interese saistībā ar POMDP problēmām. Saskaņā ar darba vadītāja ieteikumiem un darba autora izskatīto rakstu satura izvērtējumu, tika apsvērti vairāki temati darbam saistībā ar algoritmiem un to attīstīšanu. Izvēlētais algoritms, salīdzinājumā ar galvenajām pieejām pie POMDP, neizmanto varbūtību, bet gan patur atmiņā iepriekš veiktos lēmumus. Tā kā mūsdienās cilvēki ir tendēti pārnest aiz vien lielāku atbildību uz datoriem, tai skaitā – prast parūpēties par neparastām situācijām, tad ir svarīgi, lai datori pilnvērtīgi spētu pielāgoties nestandarta scenārijiem, neskatoties uz iepriekšējo novērojumu nepilnībām. POMDP ir MDP apakšnozare, kurā tiek pētīti lēmuma procesi ar tieši šādu daļējas vai ierobežotas novērošanas iespējamību.

1.2. Pētāmā problēma un hipotēze

Šajā darbā tiek pētīts piedāvātais „Sequence Q-Learning” algoritms un tās potenciālie uzlabojumi, lai efektīvāk un ātrāk trenētu datoru POMDP problēmu risināšanā. Pētījuma hipotēze: ir iespējams POMDP problēmu risināšanas algoritmā ieviest tādas izmaiņas un labojumus, kā rezultātā problēmas apstrādes laiks vai rezultāta kvalitāte tiks uzlabota. Darba gaitā Algoritms bija sekmīgi realizēts ar vairākām implementācijas modifikācijām. Saskaņā ar veiktajām izmaiņām ir gaidāms, ka algoritms tiks optimizēts problēmu risināšanai vai risināšanas ātrdarbībai.

1.3. Darba mērķis un uzdevumi

Galvenais darba mērķis ir izpētīt daļēji novērojamo Markova lēmuma procesu (POMDP) risinājumus, kā arī izveidot izvirzītā algoritma sekmīgu implementāciju un realizēt tās darbību noteiktas problēmas risināšanā. Pakārtojoties darba mērķim, darbs uzdevumi ir:

1.informācijas avotu analīze pētāmās tēmas kontekstā;

2. attīstīt „Sequence Q-Learning” algoritma veiktspēju;

3.izpētīt potenciālas algoritma pilnveidošanas iespējas un to plānoto ietekmi uz rezultātiem.

1.4. Pētniecības metodes

Šajā darbā tiek pielietotas: Informācijas avotu analīze un Izziņas pētnieciskās metodes. Darba gaitā ir nepieciešams analizēt un sintezēt algoritma darbību, loģiski izsecināt – kādas izmaiņas atstāj pozitīvas vai negatīvas ietekmes uz rezultātiem un izvirzīt vairākas hipotēzes un priekšlikumus par to, kā šis POMDP algoritms varētu tikt attīstīts tālāk.

1.5. Faktoloģiskā materiāla avotu raksturojums

Bakalaura darbā veiktā analīze un pētījums pamatojas uz publikācijām zinātnisko rakstu krājumos un periodikā, saistītajiem pētnieciskajiem datiem, Latvijas un ārvalstu zinātnieku darbiem, internetā pieejamiem materiāliem, kā arī Latvijas Universitātes mācību materiāliem.

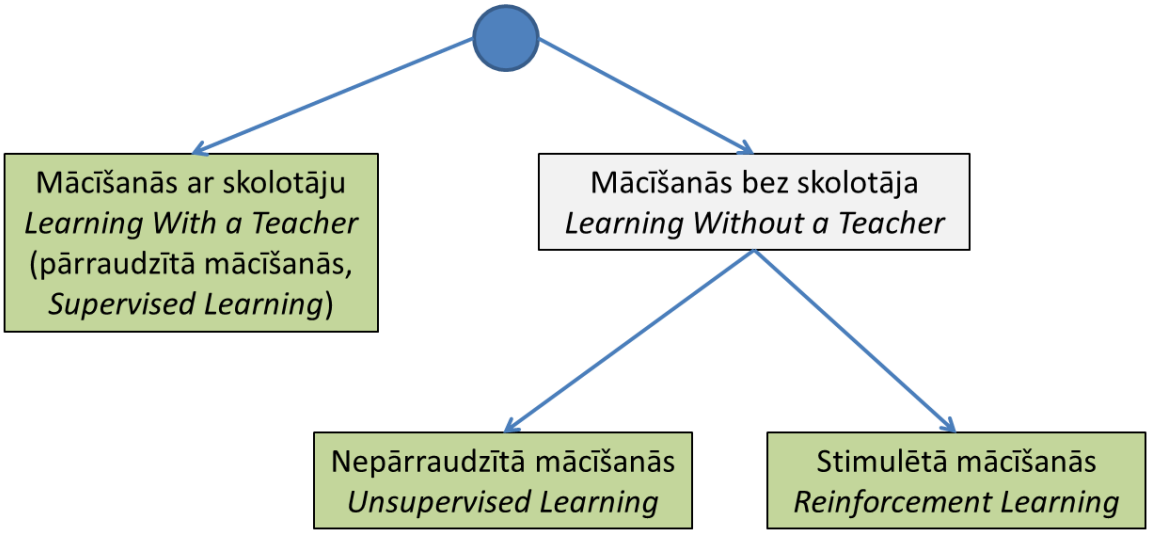
1.6. Darba struktūra

Darba struktūrā vispirms tiek sniegts skaidrojums par pētāmās problēmas nozari, to raksturojot ar piemēriem, lai sniegtu pilnīgāku izpratni par darba šauro nozari. Nākamajā nodaļā tiek sniegts detalizēts izklāsts par piedāvāto netradicionālo POMDP algoritmu „Sequence Q-Learning”, par tā stiprām pusēm, pielietošanas iespējām kā arī trūkumiem. Tas sniegs padziļinātu ieskatu par konkrētā algoritma būtību un pielietošanas prioritātēm kādās konkrētās situācijās. Autora implementācija tiek apskatīta pamatdaļas beigās, kurā tiek aprakstītas ievērojamākās algoritma modifikācijas, kā arī tā turpmāko potenciālo uzlabojumu iespējas.

2. MAŠĪNMĀCĪŠANĀS

2.1. Stimulētā mācīšanās (RL)

Stimulēta mācīšana (Reinforcement Learning)[4] ir Mašīnmācīšanas bez skolotāja paveids. Mašīnmācīšanas bez skolotāja ietvaros arī ietilpst nepārraudzītā mācīšanās (Unsupervised Learning), tomēr bieži vien tas nesniedz tikpat labus rezultātus kā lielākā daļa RL metodes. Pastāv arī pārraudzītā mācīšanas metode, kurā tiek apskatīta pasaule, vākti dati no apkārtējās vides, marķēti savāktie rezultāti un tad notiek mašīnas pārraudzītā mācīšana. Salīdzinājumā ar pārraudzīto mācīšanās paradigmu, kura ir piemērotāka konkrētai situācijai, stimulēta mācīšana atļauj mašīnai pašai iemācīties situāciju nianses, kamēr stimulācija tiek dota no iepriekš noteiktiem objektiem (skat. 2.1.1. attēlu).



2.1.1. att. ***Mašīnmācīšanās shēma***

Ja problēma tiek apskatīta kā „režģa pasaule" (gridworld), tad ir iespējams ērtāk izprast kā tieši strādā stimulētā mācīšanās. Tiek pieņemts, ka eksistē lauks, uz kura ir izvietotas darbību atlīdzības (skat. 2.1.2. attēlu). Katrai lauka zonai tiek piešķirta sava atlīdzība, kur bieži vien šādās problēmās beigu stāvokļi ir ar lielākām atlīdzībām nekā neitrālie stāvokļi. Mašīnai būs tendence tiekties sasniegt pēc iespējas lielāku atlīdzību. Ir pieņemts, ka, lai pārvietotos pa neterminālajiem stāvokļiem, tas prasa darbu, līdz ar ko to atlīdzības ir nedaudz

negatīvas. Terminālie stāvokļi var būt gan ar pozitīvām vērtībām (piemēram, labirinta izeja), gan ar negatīvām (lamatas).

-1	-1	-1	10
-1		-1	-10
-1	-1	-1	-1

2.1.2. att. ***Režģa pasaules vērtības***

Stimulētā mācīšanās sākumā parasti no apkārtējās vides tiek ievākti dati, kuri tiek pārveidoti un pasniegti datoram saprotamās vērtībās. Pareizā darbība iepriekš nav zināma nevienam solim, pieejamas ir tikai stāvokļu vērtības. Nav arī zināma atrašanās vieta vai tās atkarība no citiem stāvokļiem. Kad darbībai nepieciešamie bāzes dati ir iegūti, tiek uzsākts stimulētais mācīšanās process, kura laika tiek izdarīta gājiena izvēle (režģa pasaules gadījumā gājiens uz augšu, uz leju, pa labi vai pa kreisi). Mācīšanās process vienlaicīgi tiek veikta konstatēšana par to, kuri gājieni katrā laukā ved pie vairāk pozitīviem stāvokļiem un kādi tie gājieni ir. (skat. 2.1.3. attēlu)

→	→	→	10
↑		↑	-10
↑	→	↑	←

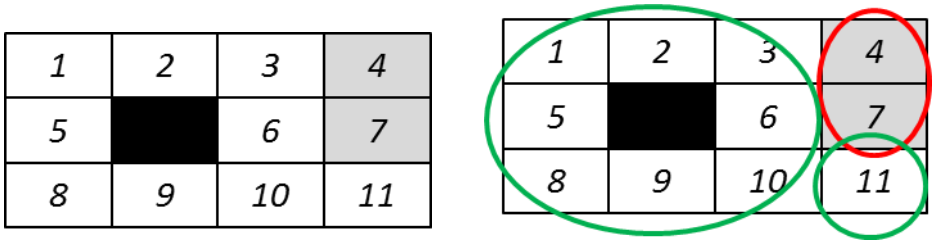
2.1.3. att. ***Režģa pasaules gājienu optimālie virzieni***

Stimulētās mācīšanas tiešsaistē piedāvā iespēju savienot apkārtējās vides izpratnes procesu kopā ar gājiena izvēles gaitu, kas kopumā atļauj mašīnai pakāpeniski nokonstatēt vides īpatnības paralēli ar analīzes procesu. Šāda pieeja problēmas risināšana var būt nepieciešama atkarībā no problēmas situatīvā rakstura.

Kopumā stimulētās mācīšanas ir balstītas uz mēģinājuma-kļūdu metodi (trial-and-error search) ar aizkavētām atlīdzībām, jo kamēr terminālie stāvokļi palēnām ietekmēs iepriekšējos soļus, mašīna jau būs paveikusi vairākas problēmas iterācijas. SM metodes ir vairāk problēmu kopums nekā algoritmu kopums, kur datu punkti (režģa gadījumā: stāvokļi) nav autonomi.

2.2. Markova lēmumu process (MDP)

Markova lēmuma process atbild par vairākām vides īpašībām un faktoriem, kas tālāk interpretāciju sasaista ar mašīnmācīšanos. Tas apraksta, kā tiek uztverti un izskatīti apkārtējās vides un vērojamās problēmas fakti, kā tiek definēti stāvokļi, kā aģents var pārvietoties pa šīs pasaules stāvokļiem un kādas atlīdzības tas sniedz aģentam, kad tiek veiktas darbības. Stāvokļi jeb pozīcija režģa pasaules gadījumā apraksta pasaules izklājumu un izskata dažādās situācijas, kurās aģents var atrasties. Starp šiem stāvokļiem tiek definēti kā sākotnējie, tā arī beigu vai terminālie stāvokļi. Aplūkotajā režģa pasaules gadījumā katrs numurētais lauks ir pasaules stāvoklis, kurā var atrasties aģents. Šajā piemērā zaļie numurētie lauki apzīmē iespējamās sākuma pozīcijas aģentam un sarkanie stāvokļi – terminālie stāvokļi (skat. 2.2.1. attēlu).



2.2.1. att. ***Režģa pasaules stāvokļu izkārtojums***

MDP satur arī darbības, kuras aģents var veikt epizodes laikā. [1] Dotajā problēmā šīs darbības ir kustības pa režģa pasauli (↑,→,↓,←). No šīm MDP īpašībām var izsecināt, ka stāvokļu pārejas (Transitions) argumenti saturēs informāciju par to, ko katra darbība veiks saistībā ar stāvokļu maiņu. Tas tiek aprakstīts, kā $S \times A \rightarrow T$; vektoriālā reizinājuma rezultātā tiek iegūta pārvietošanās matricas dimensijas. Līdzīgi arī tiek aprakstīta apbalvojumu jeb atlīdzības matrica: $S \times A \rightarrow R$. Katra no šīm matricām tiek aizpildīta ar nepieciešamajām vērtībām. Vēl viens atribūts, kas tiek saistīts ar MDP ir γ jeb diskonta faktors, kas nosaka, cik efektīvi vērtībām vajadzētu izplatīties, ejot tālāk prom no konkrētā stāvokļa.

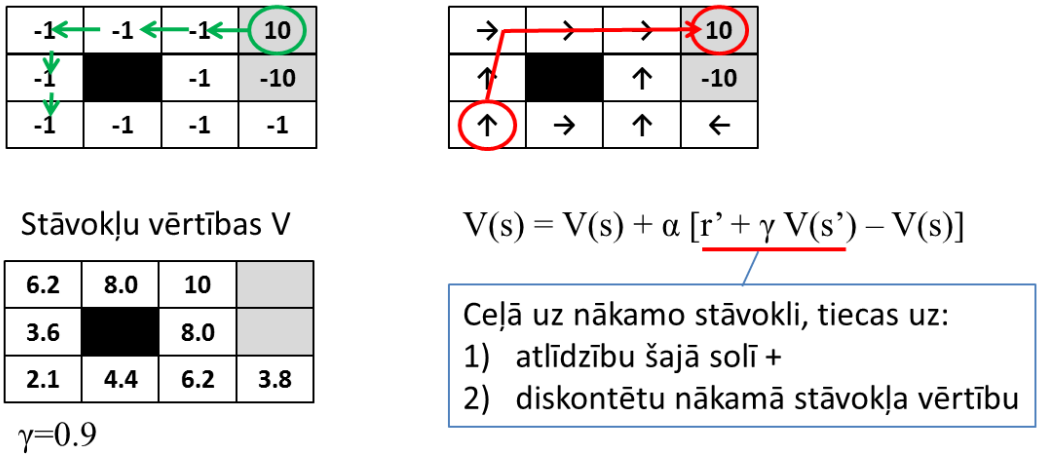
Piemēri:

- Stāvokļu pārejas
  - (5,↑) –› 1
  - (6,←) –› 6
  - (6,→) –› 7
- Atlīdzības par kustībām
  - (5,↑) –› 1

- (6,←) –› 6
- (6,→) –› 7

Vērtību balstītās metodes MDP risināšanā tiek atkārtoti izrēķināta katra stāvokļa vērtība, kad no tā tālāk pāriet uz nākamo stāvokli. Tiek ņemta vērā nākamā stāvokļa vērtība un, izmantojot to, tiek piekoriģēts iepriekšējais stāvoklis. Šādā veidā stāvokļu vērtības, kuru vērtības ir krasi atšķirīgas, izplatās uz blakus laukiem. Ar pietiekoši daudz atkārtotām epizodēm, kuru laikā tiek papildinātas šīs vērtības, tiek atbilstoši arī aprakstītas katra stāvokļa potenciālā vērtība. Balstoties uz šīm vērtībām metode tālāk izvirza labāko ceļu, jeb darbību gaitu.

Zemāk redzamajā piemērā (skat. 2.2.2. attēlu) tiek apskatītas režģu pasaules pozitīvā terminālā stāvokļa vērtības izplatīšanās līdz pašam stūrim. No tām vērtībām tiek arī izsecināts, kurš virziens būtu jāizvēlas katrā stāvoklī. Vērtības bieži vien tiek izrēķinātas pēc piemērā redzamās formulas, kurā tiek izmantota pašreizējā stāvokļa vērtība kopā ar darbības atlīdzību un diskontēta nākamā stāvokļa vērtību.



2.2.2. att. ***Režģa pasaules vērtības risināšanas gaitā***

Vērtību balstītās metodes ir ļoti dažādas un to vidū ir tādas metodes kā[1]:

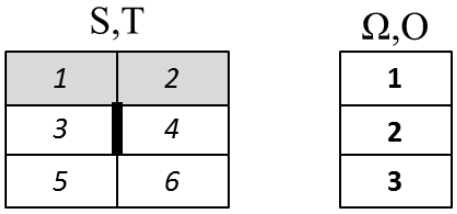
- Dinamiskā programmēšana:
  - vērtību iterācijas
  - politikas iterācijas
- Monte Carlo (MC) metode
- Temporal difference (TD) algoritmi:
  - SARSA
  - Q-learning

Kopumā MDP risināšanas algoritma galvenās komponentes sastāv no vērtību tabulas ar vērtībām katram stāvoklim, vērtību uzstādīšanas formulas un stāvokļu pārstaigāšanas metodes.

2.3. Daļēji novērojams Markova lēmumu process (POMDP)

Daļēji novērojamie Markova lēmuma procesi ir MDP paveids, kas atļauj definēt problēmas, kurās aģentam nav pieejama visa informācija par pasauli. Daļēja novērojamība nozīmē to, ka aģents nevar izšķirties starp diviem vai vairākiem stāvokļiem un ka notiek troksnis stāvokļa novērošanas procesā. Šādu problēmu dabā var sastapt ļoti bieži, it īpaši tad, ja runa iet par mašīnas prasmi orientēties vidē. Stāvokļu atlīdzības un pārvietošanās nemainās, jo tomēr kaut kur ir jāpatur atmiņā šī informācija, lai tā sniegtu savu ieguldījumu turpmākā informatīvās apmācības izvērtēšanā. Pārsvarā problēmās tiek pielikta klāt papildus darbība – novērot, kas sniedz iespēju ar mazāko troksni noorientēties starp stāvokļiem. Galvenā izmaiņas būtība ir tā, ka tiek ieviests novērojumu masīvs, kas sadala stāvokļus pa novērojumu grupām (skat. 2.3.1. attēlu). Aģentam vairs netiek dota stāvokļa informācija, bet gan novērojums. Aģentam, balstoties uz šo informāciju, ir tālāk jākoriģē sava darbība ar varbūtību.

Starp populārākajām POMDP risinājumiem ir divas galvenās pieejas. Ticamības stāvokļi POMDP risināšanā izrēķina varbūtību pēc katra novērojuma, kur tas var atrasties. Ar laiku tiek konstatēta aptuvenā atrašanās vieta ar pietiekoši lielu varbūtību un sekojoši tiek pieņemtas atbilstošās darbības.



2.3.1. att. ***2×3 Režģa pasaules novērojumi pa rindām***

Līdzīgā režģa pasaules problēmā (skat. 2.3.1. attēlu), kura izkārtojums ir 2×3 un ir siena starp vidus stāvokļiem, no sākuma tiek pieņemts, ka aģents var atrasties jebkurā starta laukā. Ja aģents vēlas novērot problēmu, tad vienīgie dati, ko tas var iegūt, ir rindas numurs. Ticamības stāvokļa risinājums izvirza varbūtības katram stāvoklim un ar katru no nākamajiem gājieniem maina tās atbilstoši (skat. 2.3.2. attēlu).

3	4	5	6
0.25	**0.25**	**0.25**	**0.25**

Epizodes sākums, neko nenovēro

3	4	5	6
0	**0**	**0.5**	**0.5**

↓, novēro 3

3	4	5	6
0	**0**	**1**	**0**

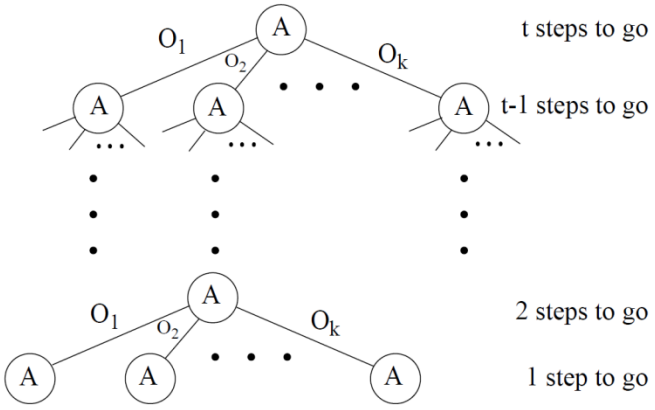
←, novēro 3

2.3.2. att. ***2×3 Režģa pasaules varbūtību vērtības pēc darbībām***

Šīs varbūtības tiek aprēķinātas, izmantojot sekojošu varbūtības formulu (1)[2], un, ja izveidojas situācija ar aģenta atrašanās stāvokļa precizēšanu, tad problēma tiek noreducēta atpakaļ uz standarta MDP problēmu.

$$b_o^a(s') = \frac{Z(s', a, o) \sum_{s \in S} T(s, a, s')\, b(s)}{\Pr(o|a, b)} \tag{1}$$

Otrs problēmas risinājums izmanto politikas koku (skat. 2.3.3. attēlu)[2], lai tiktu pie darbību sekvenču struktūras, kas tiek novērots un atjaunināts ar katru gājienu. Katra koka mezgls ir paveiktā darbība, un no katra mezgla iziet potenciālo novērojumu savienojumi dažādību skaitā uz citiem darbības mezgliem, kuri iepriekš ir noteikti. Izsekojot cauri šim kokam, aģents var potenciāli nonākt līdz terminālam stāvoklim.



2.3.3. att. ***Politikas koka piemērs***

Katrā gājienā tiek pielietota politikas koka formula, kas atjaunina koka struktūru gadījumā, ja ir neparedzēti rezultāti. Darbības tiek papildinātas ar mērķi nonākt pie vērtīgākā stāvokļa.

3. EKSISTĒJOŠAIS SEQUENCE Q-LEARNING APRAKSTS

Izpētot dažādās Mašīnmācīšanās pieejas pie POMDP problēmu risināšanas, tika izrādīta vēlme realizēt un optimizēt kādu no tās algoritmiem. Pētot publiski pieejamos materiālus un konsultējoties ar darba vadītāju, tika arī apskatīta Jāņa Zutera ierosinātā netradicionālā POMDP risināšanas metode. Rakstā „Sequence Q-Learning: a Memory-based Method Towards Solving POMDP"[6] darba autors stāda priekšā par „Sequence Q-Learning" dēvēto metodi, kas paplašina populāri zināmo un plaši pielietojamo „Q-learning" algoritmu.

Vērtību metodes saglabā un uztur vērtību tabulu katram stāvoklim vai katram stāvoklim un darbības pārim, kā rezultātā galvenie divi vērtību tipi ir darbības vērtība un stāvokļa vērtība. „Q-learning" algoritms izmanto darbību vērtības pieeju, un tas atjauno šīs vērtības, izmantojot sekojošu formulu (2)[4]:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma * max_{a'} Q(s',a') - Q(s,a)] \quad (2)$$

kur ar Q tiek apzīmēta darbības vērtību tabula; r – pašreizējā atlīdzība; α – mācīšanās ātruma koeficients; γ – diskonta faktors; s – pašreizējais stāvoklis; s' – nākamais stāvoklis; (šeit tiek izmantots apzīmējums s tāpēc, ka tiek aplūkots parastais Q-learning algoritma MDP risinājums, vēlāk šis tiks aizstāts līdzīgā formulā ar z). Šī formula tiek tālāk sadalīta metodoloģiskiem nolūkiem, lai no tās tiktu izskaitļota mērķa vērtība (target value) D((3),(4)):

$$Q(s,a) = Q(s,a) + \alpha[D(s,a,r) - Q(s,a)] \quad (3)$$

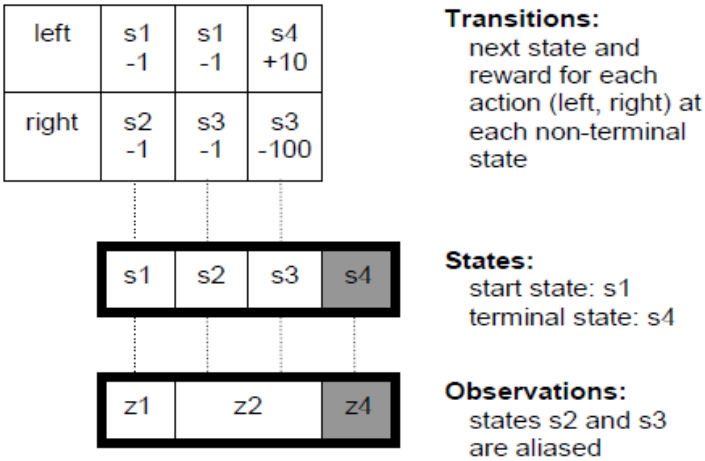
$$D(s,a,r) = r + \gamma * max_{a'} Q(s',a') \quad (4)$$

Ja salīdzinājums ar MDP problēmām, kur Q-learning metode parasti tiek pielietota, tad POMDP aģentam vairs nav pilna informācija par stāvokļiem un tas padara problēmu daudz sarežģītāku.

Jāņa Zutera rakstā[6] tiek piedāvāts atrisināt POMDP problēmas, pārvaldot sekvences jeb darbību virknes darbību vērtību vietā katrā novērojumā. Galvenā piedāvātās metodes ideja ir paplašināt „Q-learning" algoritmu tā, ka pieņemtie stāvokļi tiek potenciāli izlaisti, dodot priekšroku sasaistītās darbības virknes kopšanai un sekošanai (tā vietā, lai izvēlētos darbību katrā stāvokļa solī). Tehniski šī metode aizņemas pāris idejas no NSM (Nearest Sequence Memory)[3] pieejas, kas patur prātā epizodes laikā izdarītos novērojumus, papildinot to ar atmiņas jeb sekvenču pārvaldīšanas mehānismu un savietojot to kopā ar „Q-learning" metodes

struktūru. Piedāvātā metode salīdzinājumā ara NSM saglabās tikai darbības un novērojumus, nevis individuālas atlīdzības.

Klasiskajā „Q-learning" pieejā dators lemj jeb izskaitļo darbību noteiktā stāvoklī, kas POMDP gadījumā pārtop par novērojumu, un šis lēmums tiek veikts tikai vienai darbībai. Šajā piemērā (skat 3.1. attēlu) ir redzams, ka novērojumā z1 ar vienu darbību nepietiek, jo nākošajā novērojumā z2 – ir jāizdara neizšķirams lēmums. Ideja ir – saistībā ar vienu lēmumu izmantot vairāk nekā vienu darbību secībā pēc kārtas, lai pārvarētu divdomīgās situācijas. Tā, piemēram, to var redzēt novērojumā z2, kur ar darbību virkni var secināt, ka labākais lēmums, ko var veikt novērojumā z1 dotajā situācijā ir (→,→,←).



3.1. att. ***POMDP piemēra problēma*** [6]

Salīdzinājumā ar „Q-learning" metodi, kura saglabā tabulu ar darbību vērtībām katrai darbībai katrā stāvoklī, „Sequence Q-learning" metode saglabā un uztur virknes ar darbībām katram novērojumam atsevišķi. Secinājums, pie kura nonāca raksta autors metodes veidošanas laikā – glabāšana ir limitēta

„Sequence Q-learning" metodei ir 10 principi un soļi, uz kuriem tā balstās:

1. Ir nepieciešams mehānisms, kas ģenerētu šādu darbību virknes(5), kas ir galvenā metodes sastāvdaļa. Sekvenču garums tiek limitēts, bet tie var būt dažāda garuma. Arī sekvenču skaitu katram novērojumam ir nepieciešams ierobežot.

$$S = [(z_1, a_1), (z_2, a_2), \dots, (z_n, a_n)] \tag{5}$$

2. Katra sekvence ir raksturota ar sekvences vērtības atribūtu (darbības vērtības vietā).

3. Politika jeb lēmumu pieņemšana ir atspoguļota ar sekvenču sarakstiem un to vērtībām, kuras vienkāršākajā gadījumā ir sekvence ar visaugstāko vērtību.

4. Metodes novērošanas posmā tiek izvēlēta viena „Sekošanas-sekvence" (Follow-sequence), kas ir ekvivalents ar labāko darbību „Q-learning" algoritmā. Pārējās sekvences šajā novērojumā kļūst par „Sekošanas-kandidātiem" (Follow-candidates). Šie divi uzskati sastāda divu līmeņa nepārtrauktības principu (skat. 3.2. attēlu).

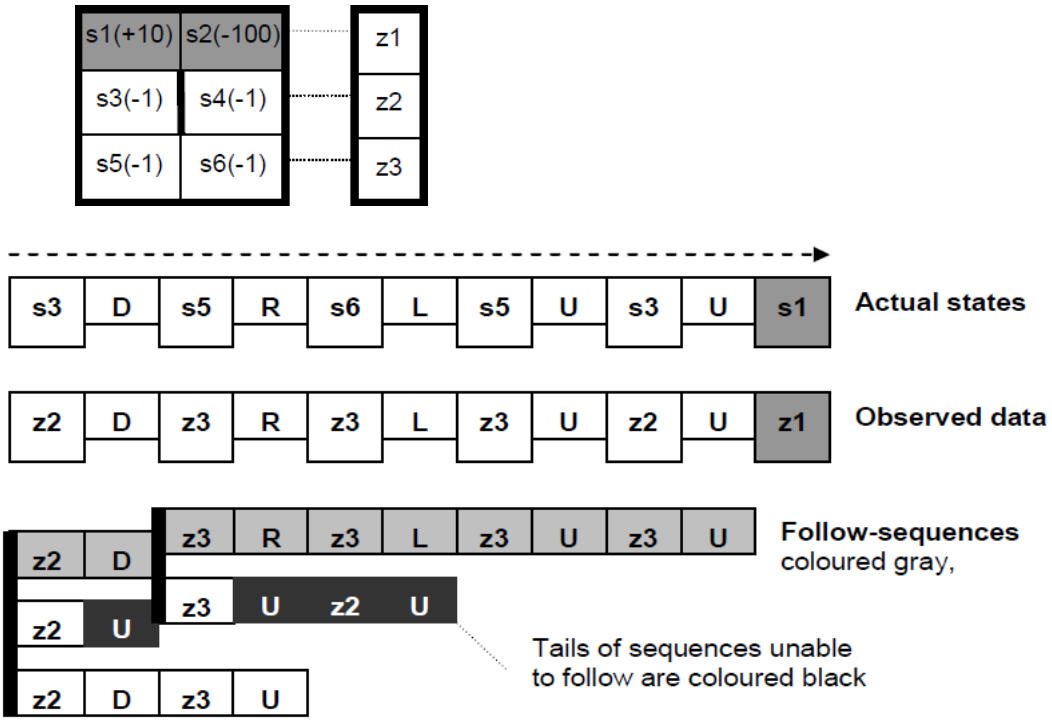
5. Galvenā ideja, kas atļauj metodei turpināt strādāt, ir viena soļa sekošana Sekošanas-sekvencei. Solī p tiek izdarīti sekojoši pārbaudījumi un darbības:

- z_p soļa novērojums par tā sakritību ar pašreizējo vides novērojumu z;
- darbība a_p ir izvēlēta un veikta;
- pasaules stāvoklis tiek mainīts, un nākamais novērojums un atlīdzība ir pieejama.

6. Izvēlētai sekvencei tiek sekots līdz viena no trim situācijām nepārtrauc sekvenci:

- tiek sasniegtas sekvences beigas;
- pasaules stāvoklis ir nonācis terminālā stāvoklī;
- sekošanas-sekvence ir pārtraukta, jo izdarītā soļa sekvences stāvoklis nesakrīt ar pasaules stāvokli.

7. No sekošanas-kandidātu saraksta tiek izslēgtas visas tās sekvences, kurām neizdodas sekmīgi izsekot ar atbilstošajiem novērojumiem un darbībām. (skat. 10. attēlu)



3.2. att. ***Sekošanas-sekvenču un sekošanas-kandidātu piemērs 2×3 režģim*** [6]

8. Pirms sekvence tiek pārlauzta, tai palaiž mācīšanās algoritmu((6),(7)):

$$Q(z,S) = Q(z,S) + \alpha[D(S,p) - Q(z,S)] \quad (6)$$

$$D(S,p) = \sum_{i=1}^{p} \gamma^{i-1} R_i + \gamma^p * max_{S'} Q(z',S') \quad (7)$$

kur R apzīmē kopējo pārvietošanās vērtību summu no sākuma līdz lūzuma punktam p.

9. Pēc tam, kad sekošanas-sekvence ir pabeigta un vērtības atjauninātas, jauna sekošanas-sekvence tiek izvēlēta no sekošanas-kandidātiem. Tikai tajā gadījuma, ja sekošanas-kandidātu saraksts ir tukšs, vēl papildus tiek pievienoti sekošanas-kandidāti no pašreizējās pasaules novērojuma sekvenču saraksta.

10. „Sequence Q-learning" implementācija ir papildus nepieciešams mehānisms, kas ģenerētu jaunus sarakstus un noņemtu sliktākos sarakstus, lai paliktu saskaņā ar saraksta izmēra ierobežojumiem.

Kopumā algoritma mērķis ir paturēt secību. Darbību secība ir svarīga POMDP problēmās un tieši tāpēc „Sequence Q-learning" metode dod priekšroku darbībām ar kontekstu pirms darbībām bez konteksta.

4. ALGORITMA IMPLEMENTĀCIJA UN MODIFIKĀCIJA

4.1. POMDP Algoritma ievadizvade

Rakstā „Sequence Q-Learning: a Memory-based Method Towards Solving POMDP" darba autors Jānis Zuters[6] arī sniedz savu implementācijas pseidokoda (skat. 1. pielikumu) paraugu. „Sequence Q-Learning" algoritms atgriež sarakstu, kura saturā ir katram novērojumam konkrēts sekvenču saraksts ar katras sekvences vērtību. Šī algoritma izvaddati tiek apzīmēti ar Ω. „Sequence Q-Learning", bez tam algoritms saņem arī šādus ievaddatus:

- Pasaule, kas apraksta problēmu tiek apzīmēta ar *mdp*;
- Mācību iterāciju skaits *N*;
- Ar μ_1 apzīmē sekvenču garumu;
- μ_2 nosaka to, cik daudz sekvences tiek uzturētas katram novērojuma.

Kad tiek izstrādāta šī POMDP, ir svarīgi izveidot stabilu datu struktūru, kas varēs saturēt visas sekvences un to vērtības. Sekvenču datu struktūra tiks aktīvi izmantota, jo atkārtoti notiks vērtību izmainīšana un datu papildināšana ar jaunām darbību virknēm.

4.2. Eksperimentā izmantotā POMDP problēma

Ir atsevišķi jāatdala vai jānošķir no pārējiem datiem galvenā sekošanas-sekvence un vairāki sekošanas-kandidāti, kuri tiek aktīvi mainīti, katras epizodes laikā.

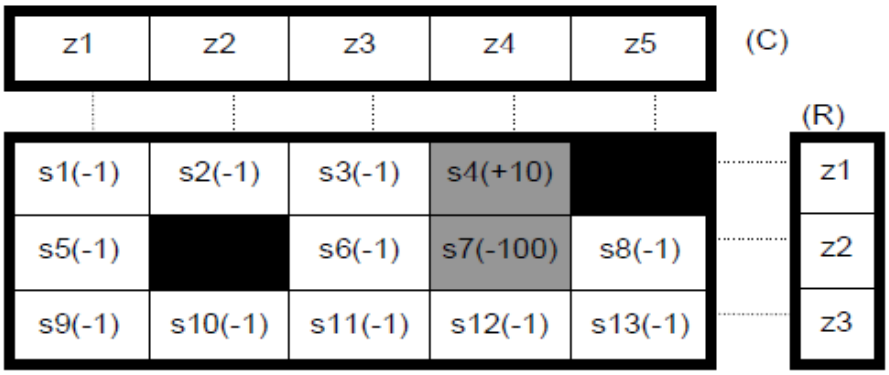
Pirms algoritms tiek tālāk apskatīts, ir jādefinē POMDP problēma, kuru darba autors apskatīs ar šo algoritma implementāciju.

Līdzīga režģu pasaules situācija tiek noteikta. Režģa dimensijas ir nomainītas uz 5×3 ar nedaudz pamainītām atlīdzībām (skat. 4.2.1. attēlu). Redzamajā piemērā atlīdzības ir ierakstītas iekavās un stāvokļu apzīmējumi ir no s1 līdz s13. Atlīdzības par pārvietošanos, ja neskaita jaunos laukus un terminālos stāvokļus, ir tādas pašas kā iepriekš. Negatīvā terminālā stāvokļa vērtība ir palielināta ar nolūku, lai aģents vairāk izvairītos no šī stāvokļa.

s1(-1)	s2(-1)	s3(-1)	s4(+10)	
s5(-1)		s6(-1)	s7(-100)	s8(-1)
s9(-1)	s10(-1)	s11(-1)	s12(-1)	s13(-1)

4.2.1. att. ***Režģu pasaules apskatāmā problēma*** [6]

Tika izvēlēts šis piemērs, jo raksta darba autors arī pielieto šādu problēmas definīciju un ir nepieciešams parūpēties par implementācijai (skat. 2. un 3. pielikumu) līdzīgu problēmas vidi. Lai parūpētos par to, ka problēma pieder pie POMDP definīcijas, ir nepieciešams to apskatīt ar novērojumiem. Dotajai problēmai tiek piedāvātas arī dažādas novērošanas konfigurācijas. Ir iespējams apskatīt problēmu kā no kolonu perspektīvas (C), tā arī no rindu perspektīvas (R) (skat. 4.2.2. attēlu).



4.2.2. att. ***Režģu pasaules apskatāmā problēma ar novērojumu konfigurācijām*** [6]

4.3. „Sequence Q-Learning" implementācijas struktūra

„Sequence Q-Learning" metode tiek iedalīta vairākās daļās, kad runa iet par implementāciju. Rakstā[6] tiek norādīti četri galvenie algoritma moduļi (skat. 1. pielikumu):

1. Darbības izvēle un tās pielietošana:

   Tiek izvēlēta nākamā darbība sekošanas-sekvences sarakstā ar nelielu iespēju samainīt šo darbību uz kādu citu. Ja sekošanas-sekvence nav uzstādīta, tad ar vienlīdzīgu iespēju tiek izvēlēta kāda no stāvokļu pārejām. Pēc darbības veikšanas *mdp* tiek atjaunināta un atbilstošā atlīdzība ar nākamo novērojumu ir pieejami;

2. Pašreizējās iterācijas sekvences apstrāde:

   Visi sekošanas-kandidāti, tai skaitā arī sekošanas-sekvence, tiek virzīti uz nākamo pozīciju. Ja sekošanas-sekvence ir pārtraukta, tad tās vērtība tiek atjaunināta un pēc tam atgriezta atpakaļ kopējā sekvenču sarakstā. Sekošanas-kandidāti arī tiek pārbaudīti uz pārrāvumiem un atgriezti atpakaļ sarakstā;

3. Sekvenču papildus ģenerēšana un vājāko noņemšana:

Tiek ģenerēta sekvence no pēdējām m iterācijām, kur m var būt jebkāda vērtība no 1 līdz μ1. Šī darbību virkne tiek pievienota kopējam sarakstam un ja saraksta garums pārsniedz μ2, tad sekvence ar vismazāko vērtību tiek izņemta no saraksta;

4. Iterācijas pēcapstrādes veikšana:

Ja epizode ir nonākusi terminālā stāvoklī, tad jauna epizode tiek izveidota tā, lai sākuma stāvoklis tiek uzstādīts kā netermināla pozīcija un tiktu iegūts novērojums. Ja sekošanas-kandidātu vairs nav tad tie tiek atjaunināti no sekvenču saraksta atbilstošajam novērojumam. Ja sekošanas-sekvence ir pabeigta, tad labākā no sekvenču-kandidātiem tiek paņemta kā nākošā ar nelielu iespēju izvēlēties nejaušu sekvenci no saraksta.

Realizējot implementāciju, kura ir balstīta šādā struktūras, tika izveidota programma, kura veiksmīgi darbojas ar dotajiem problēmas datiem. Darba autora modifikācijas vēl nav ieviestas.

4.4. Autora piedāvātā „Sequence Q-Learning" modifikācija

Ar nolūku uzlabot metodes darbību tika izvirzīta hipotēze, ka modificējot algoritmu (skat. 2. un 3. pielikumu) ir iespējams uzlabot tās ātrdarbību vai efektivitāti. Tika apskatītas vairākas iespējas mainīt gan datu struktūru, gan pašu algoritmu, kā arī „Sequence Q-Learning" metodes mācīšanās formulas.

Datu struktūra, kaut arī netika detalizēta minētajā rakstā, tika pielāgota metodes prasībām. Bija nepieciešams izveidot tādu datu struktūru, kura individuāli varētu piekļūt visiem elementiem, kā arī paturēt atmiņā to, kuras sekvences tiek izmantotas kā sekošanas-kandidāti. Tika izveidots divdimensiju masīvs, izmantojot vektora bibliotēku, kas atļāva katram masīva elementam būt definētam kā sekvenču saraksts ar tās vērtību. Pirmais masīva indekss atbild par to, kurš novērojums tiek aplūkots un otrais – par to, kura sekvence sarakstā tiek apskatīta. Sekošanas-kandidāti tai skaitā sekošanas-sekvence netiek dubleti. Tā vietā tiem līdzi tiek sekots izmantojot norāžu masīvu, kas tiek uzkopts kopā ar sekvenču masīvu. Ārējām funkcijām tiek padots tikai individuālais elements, lai nebūtu sarežģījumu.

Šādai datu struktūrai salīdzinājumā ar pseidokodā (skat. 1. pielikumu) redzamo mainīgo aprakstu, ir uzlabota gan ātrdarbība, tāpēc ka nav jātērē lieki resursi dublējot datus, gan arī atmiņa, jo nav jāuzglabā šie papildus dati.

Algoritma 1. modulī netiek nekas mainīts, ja neskaita rakstā ieteikto „ε-greedy”[5] izvēles varbūtību, kura tika implementēta ar to pašu funkciju gan gadījumā, kad ir sekošanas-sekvence, gan arī tad, kad tā ir pabeigta.

Otrais modulis arī paliek lielākoties tāds pats, jo tajā veiktās izmaiņas nesniedza nekādus sekmīgus pozitīvus rezultātus. Vērtību mācīšanas formula ir aprakstīta kā atsevišķa funkcija un tā tiek izsaukta šajā sadaļā sekošanas-sekvencei. Pati funkcija tiks apskatīta atsevišķi. Bija mēģinājums modificēt šo sadaļu, pieliekot klāt papildus mācīšanos sekošanas-kandidātiem, taču jau pēc pāris izmēģinājumiem bija skaidrs, ka līdz ar to tiek izmantoti gan papildus resursi, gan arī sekvenču vērtības tiek izrēķinātas neefektīvāk. Ar lielāku laika un resursu patēriņu un sliktākiem rezultātiem šo modifikāciju varēja nosaukt par nesekmīgu un šis algoritma modulis tika atgriezts atpakaļ, kāds tas bija iepriekš.

Trešajā modulī tiek ģenerētas un pievienotas jaunas darbību virknes. Tās tiek iegūtas, ņemot kā piemēru iepriekšējās iterācijas. Raksta pseidokodā (skat. 1. pielikumu) tiek aprakstīts, ka šis virknes tiek izvēlētas ar nejaušu garumu. Tas ir ar nodomu, lai būtu lielāka variācija sekvenču garumā un dažādībā. Pētījuma laikā tika konstatēts, ka, lai arī tas rada iespēju sekvenču sarakstam būt daudzveidīgam, tomēr tas ņem piemērus ārā no konteksta. Tika veikta modifikācija, kuras rezultātā algoritms vienmēr ņems visas pieejamās iepriekšējās iterācijas līdz noteiktajam sekvenču garumam μ1. Izvirzītā hipotēze šai modifikācijai bija, ka ar mazākiem datiem, kuri ir ārpus situācijas konteksta, varēs vairāk resursus veltīt kompleksākas sekvences pilnveidošanai. Modifikācijas rezultātā algoritma vidējais rezultāts palika nemainīgs (nedaudz uzlabojās), kaut arī rezultātu izkliedētība ir samazinājusies salīdzinājumā ar nemodificēto algoritmu, tas nozīmē, ka ir gan mazāk negatīvo, gan arī mazāk pozitīvo rezultātu (skat. 4.4.1. tabulu).

*4.4.1. tabula **Autora ģenerēšanas izmaiņu rezultātu salīdzinājums ar pamatscenāriju***

	Maksimālais rezultāts	Vidējais rezultāts	Minimālais rezultāts
Bez modifikācijas	3.6	-20.1007	-52.2
Ar modifikāciju	-3.66667	-19.008	-51.3333

Ceturtajā modulī nebija daudz iespēju pamainīt metodes pieeju problēmas risināšanā, ja neskaita papildus mainīgo uzkopšanu, kuri tika ieviesti testēšanas un vajadzības gadījumā. Šie papildus mainīgie darba gaitā bija pozīcijas sarakstos, noietās sekvenču daļas atlīdzību summa un terminālā stāvokļa mainīgais, kurš tika uzstādīts pēc darbības veikšanas un atgriezts atpakaļ uz noklusēto vērtību šajā modulī. Šis terminālā stāvokļa mainīgais tika izmantots pārbaudītajās modifikācijas vērtību formulai.

„Sequence Q-Learning" formula arī tika mainīta un modificēta pētījuma gaitā. Tā tika atdalīta no metodes moduļiem kā atsevišķa funkcija, lai to varētu ērtāk pielietos un arī lai to varētu pielietot pēc nepieciešamības vairākās vietās kodā.

Pirmā formulas ieteiktā izmaiņa (8) bija – palielināt atlīdzības vērtību gadījumā, ja epizode nonāk beigu stāvoklī. Izvirzītā hipotēze šādai modifikācijai ir, ja sekvence nonāk terminālā stāvoklī, tad šo sekvenci vajag papildus izšķirt no pārējām neterminālajām virknēm, kas paceltu arī novērojuma maksimālās sekvences vērtību.

$$D(S,p) = \sum_{i=1}^{p} \gamma^{((i-1)*f)} R_i + \gamma^p * max_{S'} Q(z', S') \quad (8)$$

Šī modifikācija pamaina diskonta koeficientu tā, lai tajā gadījumā, kad sekvence ir terminālā stāvoklī – tad viss diskonta koeficients kļūtu par 1. Tas tiek panākts ieviešot mainīgo f, kura noklusējuma v

ērtība ir 1, ja stāvoklis ir neterminmāls un 0 ja tas ir terminmāls. Pieliekot šo mainīgo klāt kā reizinājumam diskonta pakāpi, viss diskonta koeficients vai nu paliks tāds, kāds viņam bija jābūt, vai kļūs par 1 – paaugstinot vai pazeminot sekvences vērtību.

Formulas modifikācijas rezultātā tika iegūta samazināta vidējā vērtība (skat. 4.4.2. tabulu), kas visdrīzāk ir izskaidrojams ar to, ka tiek izmainītas vērtības arī tām sekvencēm, kuras bija vairāk nepieciešamas mācīšanas procesā. Piemēram, ja tiek aplūkota trešā rinda kā viens novērojums, tad viena visizdevīgākā kustība būtu uz augšu, jo tā rezultātā aģents pietuvojas pozitīvajam terminālajam stāvoklim vairumā gadījumos. Šī modifikācija ir bijusi nesekmīga, un tā netiek apsvērta kā formulas uzlabojums.

4.4.2.tabula ***Autora pirmās formulas izmaiņas rezultātu salīdzinājums ar pamatscenāriju***

	Maksimālais rezultāts	Vidējais rezultāts	Minimālais rezultāts
Bez modifikācijas	3.6	-20.1007	-52.2
Ar modifikāciju	2.66667	-22.5413	-54.4667

Otra formulas modifikācija (9), kura tika aplūkota pētījuma laikā, samaina otro diskonta koeficientu pielietojumu pie nākamā novērojuma maksimālās vērtības pret tādu pašu diskonta koeficientu, kāds tiek izmantots pie summām.

$$D(S,p) = \sum_{i=1}^{p} \gamma^{i-1} R_i + \gamma^{p-1} * max_{S'} Q(z', S') \quad (9)$$

Kaut arī sākotnējais nodoms pakāpei p ir samazināt nākamā novērojuma ietekmi uz vērtību, darba autora skatījumā būtu nepieciešams tieši otrādi šo ietekmi palielināt. Modifikācijas rezultātā tiek gaidīts, ka palielināsies aģenta centieni pārvietoties uz labvēlīgākiem novērojumiem.

4.4.3.tabula ***Autora otrās formulas izmaiņas rezultātu salīdzinājums ar pamatscenāriju***

	Maksimālais rezultāts	Vidējais rezultāts	Minimālais rezultāts
Bez modifikācijas	3.6	-20.1007	-52.2
Ar modifikāciju	2.6	-18.4967	-45.0

Šī formulas modifikācija izskatījās sekmīga. Tā ir paaugstinājusi vidējo rezultātu, kā arī samazinājusi minimālo vērtību (skat. 4.4.3. tabulu). Modifikācijas pozitīvie rezultāti liek iedziļināties situācijā. Tiek konstatēts, ka diskonts ir bijis ņemts kā piemērs no „Q-Learning" algoritma. Tā kā pašreizējā pieeja šai problēmai ir daudz savādāka, tad tiek pārbaudīta situācija, kurā vispār diskonta komponente ir noņemta no nākamā novērojuma maksimālās vērtības.

$$D(S,p) = \sum_{i=1}^{p} \gamma^{i-1} R_i + max_{S'} Q(z', S') \quad (10)$$

Otrās formulas modifikācijas tālāko izmaiņu rezultātā (skat. 4.4.4. tabulu) tika secināts, ka atkal ir redzami uzlabojumi, taču šoreiz mazāki nekā iepriekš. Ir konstatēts, ka metodei ir iegūti uzlabojumi no formulas modifikācijas, kaut arī tā rezultātā iekšējās sekvenču vērtības nebūs iespējams salīdzināt ar sākotnējo algoritma metodi. Kad individuālās sekvences tika aplūkotas, bija pamanāmi uzlabojumi, taču to novērtēt ir grūtāk, jo viss ir atkarīgs no individuālās gadījuma situācijas.

4.4.4.tabula ***Autora otrās formulas tālākās izmaiņas rezultātu salīdzinājums ar pamatscenāriju***

	Maksimālais rezultāts	Vidējais rezultāts	Minimālais rezultāts
Bez modifikācijas	3.6	-20.1007	-52.2
Ar modifikāciju	-2.8	-18.078	-43.8667

Vēl viena modifikācija, kura tika apsvērta pētījuma gaitā, bija izmainīt apbalvojumu datus stāvokļiem tā, lai tie mašīnas mācīšanās laikā būtu labvēlīgāki īsākām un pozitīvām sekvencēm. Tika aplūkoti divi gadījumi:

1. Stāvokļu pārejas uz sienām - kuras tiek papildus sodītas;
2. Negatīvā Terminālā stāvokļa vērtība, kura tiek samazināta.

Pirmajā gadījumā visas „sadursmes" ar režģa sienām tiek sodītas ar -2 atlīdzību (skat. 4.4.1. attēlu). Hipotēze ir, ka ar šādām vērtībām sekvences, kuras atkārtoti izdara liekas darbības (kuras pārsvarā ir ieskriešana sienā) tiks vērtētas mazāk, līdz ar ko tās būs mazāk sastopamas starp risinājumiem. Rezultātā tika iegūtas neveiksmīgākas sekvences, kur visām sekvencēm bija līdzvērtīgākas vērtības, jo kaut arī garākas sekvences bieži izgāzās atkārtoti saduroties ar sienām, derīgo īsāko sekvenču vērtības arī tika pasliktinātas. Ja tiek aplūkots z3, kas ir trešās rindas novērojums, tad pusē gadījumos vajadzīgā darbība uz augšu vai pa kreisi netiek attīstīta vērtības ziņā, jo tās ietriecas sienā vai nonāk terminālā stāvoklī.

-2 -2 -1 -1	-2 -1 -1 -2	-2 -1 +10 -1	+10	
-1 -2 -2 -1		-1 -2 -100 -1	-100	-2 -100 -2 -1
-1 -2 -1 -2	-2 -1 -1 -2	-1 -1 -1 -2	-100 -1 -1 -2	-1 -1 -2 -2

4.4.1. att. ***Režģu pasaules pārvietošanās atlīdzības 1. konfigurācija***

Otrā datu modifikācija bija 7. stāvokļa vērtības izmaiņa no -100 uz -10 (skat. 4.4.2. attēlu), tā lai gan negatīvie, gan pozitīvie terminālie stāvokļi būtu vienādi. Paredzamās sekas bija, ka darbības, kuras iet garām 7. stāvoklim nebūtu negatīvi ietekmētas.

-1	-1	-1	10	
-1		-1	-10	-1
-1	-1	-1	-1	-1

4.4.2. att. ***Režģu pasaules pārvietošanās atlīdzības 2. konfigurācija***

Izpētes rezultātā netika manītas nekādas atšķirības algoritma veiktspējā starp šiem datiem un pamatscenāriju datiem. Šāds rezultāts visdrīzāk ir izskaidrojams ar to, ka rezultātu vērtības nav krasi mainījušās būtībā, bet tikai amplitūdā. Tas tika apstiprināt, kad tika izdarīta pārbaude, kur terminālo stāvokļu atlīdzības tika nomainītas uz 1 un -2 atbilstoši. Šī modifikācija nav sniegusi nekādas izmaiņas „Sequence Q-Learning" darbībā.

REZULTĀTI

Pētījuma gaitā tika realizēts darba mērķis – izpētīt daļēji novērojamo Markova lēmuma procesu (POMDP) risinājumus, kā arī izveidot izvirzītā algoritma sekmīgu implementāciju un realizēt tās darbību noteiktas problēmas risināšanā, un mērķim pakārtotie darba uzdevumi:

1.informācijas avotu analīze pētāmās tēmas kontekstā;

2. attīstīt „Sequence Q-Learning" algoritma veiktspēju;

3.izpētīt potenciālas algoritma pilnveidošanas iespējas un to plānoto ietekmi uz rezultātiem.

Kopumā pētījuma laikā tika detalizēti apskatīta „Sequence Q-Learning" metode POMDP problēmu risināšanā, un tur, kur bija redzamas iespējamas pielikt uzlabojumus, tās arī tika veiktas kā modifikācijas un izmaiņas. Daļa hipotētiski izvirzīto modifikāciju nebija sekmīgas un deva tikai negatīvus rezultātus salīdzinājumā ar pamatscenāriju. Tomēr izdevās iegūt arī pozitīvus un neitrālus novērojumus citām izmaiņām:

1. Viena no galvenajām modifikācijām, kuras tika veiktas ar šo metodi, bija datu struktūras uzlabojums, kurš tika izstrādāts jau no paša sākuma. Tas atviegloja darbu datoram un neaizņēma pārāk daudz vietas atmiņā.

2. Pašam algoritmam bija viena sekmīga modifikācija, kura neuzlaboja rezultātu tieši, bet gan samazināja rezultāta dažādību un nedaudz pacēla vidējo iegūto rezultātu. Šī modifikācija izmainīja to, kā tiek ģenerētas jaunas sekvences.

3. Tika izmainīta arī formula uz kā ir balstīta „Sequence Q-Learning". Modifikācijas rezultātā iznāca iegūt uzlabotu rezultātu, kā arī samazināt rezultāta daudzveidību. Negatīvie rezultāti krasi samazinājās, kamēr vidējais palielinājās nedaudz.

4. Bija mēģinājums izmainīt problēmas datus, uz kuriem algoritms strādāja, lai izprastu vai var virzīt algoritmu prom no sienām (atkārtotām nevajadzīgām darbībām) un nesodīt tik būtiski darbības, kuras notiek blakus negatīviem stāvokļiem. Izmaiņas nedeva nekādus pozitīvus rezultātus, tomēr papildināja autora izprašanu, par ievaddatu ietekmi uz algoritmu.

Ir jāpiemin, ka modifikāciju laikā to rezultāti tika salīdzināti, izmantojot vidējo aritmētisko vērtību no 100 mašīnas apmācības mēģinājumiem. Katrā apmācības mēģinājumā no sākuma dators izrēķina vērtības visām sekvencēm, no kurām katram novērojumam tālāk tiek paņemta visaugstāk novērtētā sekvence un pēc tam tie tika pārbaudīti. Pārbaude notika ar aģentu izvietošanu katrā no 11 sākuma stāvokļiem un, sekojot šīm labākajām sekvencēm, tika aplūkots iegūtais rezultāts, kas arī veidoja katras apmācības reizes vērtību.

DISKUSIJA

Kaut arī tika atrasta labāka datu struktūra metodei, tomēr tiek pieļauta iespējamība, ka šajā jomā vēl arvien varētu iegūt uzlabojumus. Pētījuma laikā tika atrasti vairāki alternatīvi varianti datu glabāšanai POMDP metodēs, tai skaitā datu glabāšana koka shēmā. Politikas koks, kurš tiek izmantots alternatīvā POMDP problēmu risināšanas metodē, varētu tikt pielāgots „Sequence Q-Learning" algoritmam. Tas atļautu ieekonomēt vēl vairāk atmiņas, kā arī potenciāli dod iespējas savienot šo algoritmu ar kādu no koka apstrādes metodēm.

Paša algoritma struktūra bija pietiekoši korekta, līdz ar to pētījuma gaitā bija grūtības atrast kaut kādus uzlabojumus kādā no metodes moduļiem. Vienīgais uzlabojums bija sekvenču ģenerēšanā, kur veiktās izmaiņas nedaudz uzlaboja vidējo rādītāju. Tomēr šī modifikācija arī samazināja pozitīvākos rezultātus, kas ne visos gadījumos varētu būt izdevīgi. Algoritma tālākai pilnveidei ir ieteicams apskatīt vēl citas metodes jaunu sekvenču ģenerēšanā, kā piemēram, ņemot tikai tos soļu kopumus no iepriekšējām iterācijām, kuri sniedza vislielāko vērtības izmaiņu. Tas atļautu algoritmam sadalīt iepriekšējās sekvences vairākos noderīgākos gabalos.

Apskatot „Sequence Q-Learning" formulu, tika konstatēts, ka tai varēja palikt kaut kādi iespaidi no „Q-Learning" algoritma, kuri šajā situācijā nestrādātu. Diskonta komponente ir domāta tam, lai atšķirtu pašreizējo stāvokļa atlīdzību no nākamajiem stāvokļiem, kā rezultātā arī radās ideja izmēģināt šo mainīgo pavariēt. Ir ļoti iespējams, ka šāda konfigurācija radīja uzlabojumus tikai šai problēmai, tāpēc ir ieteicams paeksperimentēt arī ar šo konstanti. Visdrīzāk sākotnējā „Sequence Q-Learning" formula bija pareiza, bet konstanti vajag atbilstošāk pielāgot (diskonta faktors var būt no 0 līdz 1) POMDP problēmām, kuras tiek risinātas.

NOBEIGUMS

Darba gaitā ir sekmīgi sanācis izpētīt vairākus daļēji novērojamo Markova lēmuma procesu risinājumus. No izpētītajām metodēm detalizētāk tika apskatīta „Sequence Q-Learning" metode, kura tika analizēta un īstenota. Sākotnējā implementācija bija izveidota pēc raksta dotās informācijas. Kad tā veiksmīgi bija darbināta noteiktas POMDP problēmas kontekstā, darba autors varēja sākt to modificēt un izmainīt, skatoties kur potenciāli varētu iegūt uzlabojumu algoritmā.

Metodes sekmīgās modifikācijas atļāva algoritmam izmantot mazāk atmiņas, kā arī ātrāk un efektīvāk apstrādāt POMDP problēmas. Ne visi izmaiņu mēģinājumi bija sekmīgi un daži nesniedza nekādas ievērojamas izmaiņas, tomēr pozitīvās izmaiņas un tālākā algoritma potenciālā attīstība bija izmeklētas.

Metodes modifikācijas laikā tika izvirzītas vairāki pētījuma pieņēmumi katrai modifikācijai.

Izvirzītais darba mērķis un tam pakārtotie uzdevumi ir paveikti sekmīgi un, kopā ar praktisko izklāstu, var secināt, ka darbs ir bijis veikts pietiekami pilnvērtīgi. Darba izpildes laikā bija cieša, atbalstoša un radoša sadarbība ar darba vadītāju, kura deva jūtamu dinamiku darba virzībā un potenciālo materiālu atlasē.

Datu struktūras uzlabošanas, kurā vairāk tiek izmantotas norādes uz sekvencēm, pieņēmums bija pareizs, jo tas aizņēma mazāk vietas atmiņā un netērēja datora resursus elementu dublicēšanā. Pieņēmums, ka ar papildus mācīšana katrai sekvencei sekošanas-kandidātos dotu uzlabojumus bija aplams, līdz ar ko netiek ietikts kā uzlabojums. Sekvenču ģenerēšanas modifikācija, tika izvirzīta no pieņēmuma, ka, ja tiek izņemtas darbības ārā, no konteksta tās var samazināt resursus, kuri varētu būt izmantoti uz garāku sekvenču attīstīšanu. Šis pieņēmums deva neitrālus rezultātus, kaut arī bija palielinājies garāku sekvenču skaits un vidējais rezultāts bija nedaudz paaugstinājies, maksimālā vērtība bija kritusies. Pirmais formulas pētījuma pieņēmums bija aplams un deva negatīvus rezultātus, bet otrais pieņēmums, ka diskonta koeficienta izmaiņa pie maksimālās nākamās sekvenču vērtības dotu, pozitīvus rezultātus tika apstiprināts. Otrā modifikācijai ir vēl iespēja attīstīties, jo ir iespējams, ka šo vajadzētu precīzāk pielāgot problēmai. Pieņēmums, ka ievaddatu izmaiņas uzlabotu algoritma darbību ar šo problēmu, izrādījās nepatiess šajā situācijā.

IZMANTOTĀ LITERATŪRA

1. Latvijas Universitātes kursa DatZ4057: Mašīnmācīšanās izvēlētas nodaļas palīgmateriāli [tiešsaiste; apskatīts 05.2017.] Pieejams: http://home.lu.lv/~janiszu/mlearning/mlearning_lab.pdf
2. Kaelbling L.P. Planning and acting in partially observable stochastic domains [tiešsaiste apskatīts 05.2017.] http://people.csail.mit.edu/lpk/papers/aij98-pomdp.pdf
3. McCallum A.R. Instance-Based State Identification for Reinforcement Learning [tiešsaiste apskatīts 05.2017.] https://papers.nips.cc/paper/932-instance-based-state-identification-for-reinforcement-learning.pdf
4. Sutton Barto. Reinforcement Learning: An Introduction. 2012 [tiešsaiste apskatīts 05.2017.] http://people.inf.elte.hu/lorincz/Files/RL_2006/SuttonBook.pdf
5. Tokic M. Adaptive ε-greedy Exploration in Reinforcement Learning Based on Value Differences [tiešsaiste apskatīts 05.2017.] http://tokic.com/www/tokicm/publikationen/papers/AdaptiveEpsilonGreedyExploration.pdf
6. Zuters J. (2015) Sequence Q-Learning: a Memory-Based Method Towards Solving POMDP. *Proceedings of the 20th International Conference on Methods and Models in Automation and Robotics* (MMAR 2015), Miedzyzdroje, Poland, August 24-27, 2015, pp. 495-500, ISBN 978-1-4799-8701-6

PIELIKUMS

1. Piedāvātais „Sequence Q-Learning" algoritma implementācijas pseidokods[6]

```
algorithm sequence_qlearning (mdp, N, μ1, μ2) returns Ω:
    mdp – the environment representing the problem;
    N – number of iterations to process;
    μ1 – maximum sequence length
    μ2 – maximum number of sequences per observation
    Set internal state of mdp to some non-terminal state, perceive
        observation z
    let φ = None  # the main "follow-sequence"
    let Φ = {}  # "follow-candidates"
    let Ω = [{},{},....,{}] # All stored sequences by observations
    repeat N times:
      # (1/4) choose action and apply it
      a = get action from φ at current position (ε-greedy)
          (if φ is not None, otherwise select action at random)
      apply action a to mdp, perceive observation z and reward r
      # (2/4) process current iteration of sequences
      Advance all sequences of Φ to the next position
      if φ is unmatched or finished:
          Apply learning to φ (Eqs. (5),(6))
          let φ = None
      Remove all unmatched or finished sequences from Φ
      # (3/4) generate and remove sequences
      Let G[(z1,a1), (z2,a2),..., (zm,am)] is sequence of last iterations:
          m is random value between 1 and μ1:
      Add sequence G to Ω[z1] if already not present
      if length (Ω[z1]) > μ2:
          remove the worst sequence from Ω[z1]
      # (4/4) do post-processing of the iteration
      if mdp reached terminal state:
          Set internal state of mdp to some non-terminal state,
              perceive observation z
      if Φ == {}: let all sequences of Ω[z] are added to Φ:
      if φ == None: let φ = the best sequence from Φ (ε-greedy)
```

2. Darba autora formulas implementācija ar modifikācijām

```
void learning(float alpha, float d_f, s_v* p, int pos, episode mdp, vector<
vector<s_v> > omega, float R_sum, int finish)
{
    float max_val=0;
    if(omega[mdp.z].size()!=0){
        max_val=omega[mdp.z][0].value;
        for (int i=1; i<omega[mdp.z].size();i++)
        {
            if (max_val<omega[mdp.z][i].value)
            {
                max_val=omega[mdp.z][i].value;
            }
        }
    }
    float D = pow(d_f,(pos/* *finish */))*R_sum + pow(d_f,pos/*-1*/)*max_val;
    p->value+= alpha*(D-(p->value));
}
```

```
vector< vector<s_v> > sequence_qlearning(int N, int u1, int u2)
{
    int phi = -1;
    int phi_pos=0;
    vector<s_v *> PHI;
    vector<int> PHI_it;
    vector<o_a> G;
    vector<float> G_v;
    vector< vector<s_v> > omega;
    omega.resize(3);
    omega[0].reserve(u2+1);
    omega[1].reserve(u2+1);
    omega[2].reserve(u2+1);
    episode mdp = episode();
    float R_sum = 0.0;
    do
    {
        mdp.s = rand()%15;
    } while (!S[mdp.s]);
    mdp.observe();

    int finish_state_reached = 0;

    for (int i=0; i<N; i++)
    {
        // PIRMAIS MODULIS
        o_a current;
        if (phi != -1)
        {
            current.action = PHI[phi]->sequence[phi_pos].action;
            phi_pos++;
            current.observation=mdp.z;
            mdp = next_state(mdp, eps, current.action);
            current.action = mdp.a;
            R_sum+=mdp.r;
        }
        else
        {
            current.action = 3;
            current.observation=mdp.z;
            mdp = next_state(mdp, 0.25, current.action);
            current.action = mdp.a;
        }

        if (F[mdp.s])
        {
            finish_state_reached = 1;
        }
```

```
        // OTRAIS MODULIS
        for (int j=0; j<PHI_it.size(); j++)
        {
            PHI_it[j]++;
        }
        if (phi != -1)
        {
            if ((phi_pos >= PHI[phi]->sequence.size()) || (PHI[phi]-
>sequence[phi_pos].observation!=mdp.z) || (PHI[phi]->sequence[phi_pos-
1].action!=current.action))
            {
                //Apply learning to phi (Eqs. (5),(6))

learning(alp,discount_rate,PHI[phi],phi_pos,mdp,omega,R_sum,finish_state_reac
hed);
                phi = -1;
                phi_pos = 0;
            }
        }
        for (int j=0; j<PHI_it.size(); j++)
        {
            if ((PHI_it[j]>=PHI[j]->sequence.size()) || (PHI[j]-
>sequence[PHI_it[j]].observation!=mdp.z) || (PHI[j]->sequence[PHI_it[j]-
1].action!=current.action))
            {

//learning(alp,discount_rate,PHI[j],PHI_it[j],mdp,omega,R_sum,finish_state_re
ached);
                PHI_it.erase(PHI_it.begin()+j);
                PHI.erase(PHI.begin()+j);
                if (phi>j) {phi--;}
                else if (phi==j)
                {
                    phi = -1; phi_pos=0;
                }
                j--;
            }

        }


        // TRESAIS MODULIS
        G.push_back(current);
        G_v.push_back(mdp.r);
        if (G.size()>u1)
        {
            G.erase(G.begin()+0);
            G_v.erase(G_v.begin()+0);
        }
        int n=0;
        //n = rand()%G.size();
        vector<o_a>::const_iterator first = G.begin()+n;
        vector<o_a>::const_iterator last = G.end();
        vector<o_a> subG(first, last);
```

```
        for(int j=0; j<=omega[G[0+n].observation].size(); j++)
        {
            if (j==omega[G[0+n].observation].size())
            {
                s_v tmp;
                tmp.sequence = subG;
                float sum = accumulate(G_v.begin()+n, G_v.end(), 0.0)/(n+1);
                tmp.value = 0;
                learning(alp,discount_rate,&tmp,G.size()-1-
n,mdp,omega,sum,finish_state_reached);
                omega[G[0+n].observation].push_back(tmp);
                break;
            }
            else if (compare_sequences(omega[G[0+n].observation][j].sequence,
subG))
            {
                break;
            }

        }
        if (omega[G[0+n].observation].size()>u2)
        {
            int min_pos=0;
            for (int j=1; j<=u2; j++)
            {
                if (omega[G[0+n].observation][min_pos].value >=
omega[G[0+n].observation][j].value)
                {
                    if ( (omega[G[0+n].observation][min_pos].value ==
omega[G[0+n].observation][j].value)
                        &&
(omega[G[0+n].observation][min_pos].sequence.size() >
omega[G[0+n].observation][j].sequence.size()) )
                    {
                        continue;
                    }
                    else
                    {
                        min_pos=j;
                    }
                }
            }
            for (int j=0; j<PHI.size(); j++)
            {
                if (PHI[j] == &omega[G[0+n].observation][min_pos])
                {
                    PHI_it.erase(PHI_it.begin() + j);
                    PHI.erase(PHI.begin() + j);
                    if (phi>j) {phi--;}
                    else if (phi==j) {phi = -1; phi_pos=0;}
                    break;
                }
            }
            omega[G[0+n].observation].erase(omega[G[0+n].observation].begin()
+ min_pos);
        }
```

```
        // CETURTAIS MODULIS
        if (F[mdp.s])
        {
            mdp = episode();
            do
            {
                mdp.s = rand()%15;
            }while (!S[mdp.s]);
            mdp.observe();
            G.clear();
            G_v.clear();
            PHI.clear();
            PHI_it.clear();
            phi = -1;
            phi_pos = 0;
            finish_state_reached = 0;
            R_sum = 0.0;
        }
        if (PHI.size()==0)
        {
            R_sum = 0.0;
            for (int j=0;j<omega[mdp.z].size();j++)
            {
                s_v * tmp = &(omega[mdp.z][j]);
                PHI.push_back(tmp);
                PHI_it.push_back(0);
            }
        }
        if ((phi == -1) && (PHI.size()>0))
        {
            phi = next_sequence(PHI, eps);
            phi_pos = PHI_it[phi];
            R_sum = 0.0;
        }
    }
    return omega;
}
```

Prasībām noslēguma darbu
izstrādāšanai un aizstāvēšanai LU

Bakalaura darbs „POMDP problēmu risināšana, izmantojot vēsturiskus elementus, un tās optimizācija” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Elvis Egle

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītāja: asociētais profesors Dr. dat. Jānis Zuters _____ 29.05.2017.

Recenzents: profesors Dr. dat. Juris Smotrovs

Darbs iesniegts Datorikas fakultātē 29.05.2017.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

Komisijas sekretāre: