

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Računalništvo in matematika – 2. stopnja

Anže Marinko

**VODENJE PSA OVČARJA Z UPORABO
UMETNE INTELIGENCE**

Magistrsko delo

Mentor: izr. prof. dr. Iztok Lebar Bajec

Somentor: doc. dr. Jure Demšar

Ljubljana, 2021

Zahvala

Hvala mentorju izr. prof. dr. Iztoku Lebarju Bajcu in somentorju doc. dr. Juretu Demšarju za usmerjanje in pomoč pri raziskovanju in pisanju. Zahvaljujem se tudi vsem ostalim profesorjem, ki so me v teh letih pripravljali in me navduševali nad matematičnimi in računalniškimi tematikami, ki mi bodo koristile v življenju.

Zahvaljujem se svojim staršem, Tonetu in Mateji, bratom Luku, Janezu, Denisu, Matevžu in Alenu ter sestrama Sari in Ani, za zgled in podporo pri učenju ter študijskemu delu vsa ta leta. Hvala tudi moji zaročenki Karolini za spodbudne besede in da mi v vsem stoji ob strani. Hvala vsem prijateljem, soskavtom in sopevcem za navdihajoče okolje.

Hvala Ani Kepic za slovnični pregled dela.

Kazalo

Program dela	ix
1 Uvod	1
1.1 Organizacija dela	2
2 Obnašanje ovc	3
2.1 Strömbomov model	3
2.1.1 Odbojna sila	4
2.1.2 Sila združevanja	4
2.1.3 Sila bega pred ovčarjem	5
2.1.4 Izračun premika	6
2.1.5 Izogibanje ograji	7
2.2 Popravljen Strömbomov model	7
2.3 Ginellijev model	9
2.3.1 Izračun smeri	9
2.3.2 Sprememba stanja obnašanja	10
2.3.3 Beg pred ovčarjem	11
3 Ročno razviti model vodenja psa ovčarja	13
3.1 Stanja obnašanja ovčarja	13
3.1.1 Stanje zbiranja črede in stanje vodenja črede	13
3.1.2 Ignoriranje dela črede	15
3.1.3 Stanje naključnega premika	15
3.1.4 Stanje premika za čredo	15
3.1.5 Izbira hitrosti gibanja	16
3.2 Sodelovanje skupine ovčarjev	16
3.3 Premik ovčarja	17
4 Model vodenja razvit z genetskim algoritmom	21
4.1 Gen in evalvacija uspešnosti gena	21
4.2 Nova generacija	22
4.2.1 Parjenje	22
4.2.2 Križanje	24
4.2.3 Mutacija	24
4.2.4 Državno tekmovanje	24
4.3 Naša implementacija genetskega algoritma	25
5 Model vodenja z adaptivnim genom	27
5.1 Spodbujevano učenje	27
5.2 Optimizacija z bližnjo strategijo	28
5.3 Adaptivni gen	30
5.4 Naša implementacija modela	31

5.4.1	Nagrajevanje	31
5.4.2	Opozovanje vrednosti	31
6	Program iOvčar IZIDOR	33
7	Rezultati	35
7.1	Iskanje optimalnega gena	36
7.2	Model razvit z genetskim algoritmom	40
7.3	Model z adaptivnim genom	42
7.4	Primerjava modelov vodenja	43
7.5	Razprava	43
8	Zaključek	47
	Literatura	49

Kazalo slik

1	Območja simulacijskega okolja	3
2	Območja zaznave	4
3	Odbojna sila	5
4	Sila združevanja	5
5	Sila bega pred ovčarjem	6
6	Struktura črede po vseh treh modelih	8
7	Možni prehodi med stanji gibanja	11
8	Stanje zbiranja in vodenja črede	14
9	Stanje premika za čredo	16
10	Razdelitev črede in nalog	17
11	Izogibanje ograji, zaokrožanje okrog ovce in okrog črede	18
12	Razmnoževanje uspešnejših genov	23
13	Verjetnost parjenja	23
14	Agent in okolje pri spodbujevanem učenju	27
15	Predstavitev prostora z žarki	32
16	Pogledi med simulacijo	33
17	Meni programa	33
18	Delež ovc na pašniku skozi čas	35
19	Uspešnost skozi generacije	37
20	Trajanje simulacije skozi generacije	38
21	Delež ovc v staji ob koncu skozi generacije	39
22	Vrednost genotipa za parameter r_a skozi generacije	40
23	Naučene vrednosti prvih parametrov	41
24	Poti agentov med simulacijo	42
25	Ocenjevanje dobljene nagrade in dejanska dobljena nagrada	43
26	Delež ovc v staji za različne črede in število ovčarjev	44
27	Povprečen čas in delež zaključenih simulacij	45
28	Primerjava modelov vodenja enega ovčarja	45

Kazalo tabel

1	Parametri Strömbomovega modela gibanja ovc	7
2	Parametri Ginellijevega modela gibanja ovc	10
3	Parametri vodenja psa ovčarja	19
4	Parametri genetskega algoritma	25

Program dela

V magistrski nalogi bo kandidat preučeval smiselnost uporabe metod umetne intelligence za učenje vodenja psa ovčarja. V danem problemu je cilj psa ovčarja, da skupino ovc čim hitreje privede v hlev.

V prvi fazji naloge bo kandidat pridobil oziroma osvežil potrebna znanja. Simulacije bo sprogramiral v orodju Unity. Okolje se primarno uporablja za razvoj računalniških iger, a je več kot primerno tudi za izdelavo raznih vizualnih simulacij [4]. V sklopu učenja orodja Unity, se bo kandidat moral naučiti tudi programskega jezika C#.

Za učenje vodenja psa bomo uporabili dva različna pristopa. Prvi pristop so genetski algoritmi, pri njih se entitete učijo oziroma napredujejo skozi poenostavljeni simuliranje naravne evolucije [16]. Za drugi pristop smo izbrali bolj moderen in kompleksen programski paket imenovan ML Agents. Metoda je osnovana na spodbujevanem učenju in je integrirana v orodje Unity.

Naučeno vodenje psa bomo temeljito evalvirali. Uspešnost naučenega vodenja v obeh primerih (genetski algoritmi in ML Agents) bomo primerjali z ročno sprogramanim vodenjem [17]. Primerjavi bomo izvedli na dveh različnih modelih vedenja ovc, na enostavnem modelu [17] ter na modelu zgrajenem s pomočjo dejanskega gibanja ovc v naravi [6].

Osnovna literatura

- [17] D. Strömbom in dr., *Solving the shepherding problem: heuristics for herding autonomous, interacting agents*, Journal of the royal society interface **11**(100) (2014) 20140719
- [6] F. Ginelli in dr., *Intermittent collective dynamics emerge from conflicting imperatives in sheep herds*, Proceedings of the National Academy of Sciences **112**(41) (2015) 12729–12734
- [16] D. Shiffman, S. Fry in Z. Marsh, *The nature of code*, 2012
- [4] J. Demšar, W. Blewitt in I. Lebar Bajec, *A hybrid model for simulating grazing herds in real time*, Computer Animation and Virtual Worlds **31**(1) (2020) e1914

Podpis mentorja:



Podpis somentorja:



Vodenje psa ovčarja z uporabo umetne inteligence

POVZETEK

V naravi lahko en sam ovčar zbere in vodi čredo več sto ovc, ljudje pa si želimo zbirati, voditi ali preusmerjati tudi druge vrste živali. V bližini letališč imajo letala pogosto težave z jatami ptic, ki letalom med drugim povzročajo škodo na motorjih ali celo strmoglavljanja. V ta namen so že mnogi raziskovali različne modele vodenja s pomočjo umetne inteligence, da bi se podobnih problemov lahko lotili z roboti ali droni. Uporaba pametnih ovčarjev bi lahko prav prišla tudi pri zbiranju razlite nafte na vodi ali pri vodenju panične množice ljudi na varno v kriznih situacijah.

V tem delu študiramo različne modele gibanja ovc, razvijamo model vodenja in ga še izboljšamo z metodami umetne inteligence. Pri tem predlagamo rešitve za različne težave, ki se pri učenju pojavljajo. Razviti želimo namreč model, po katerem ovčarji kar se da hitro v stajo prepeljali celotno čredo. Osnovnemu modelu vodenja najprej dodamo sodelovanje več ovčarjev, nato najdemo dobre parametre modela za izbrane velikosti črede in število ovčarjev. Na koncu razvijemo model, pri katerem se na podlagi izbranih informacij ovčar sproti odloča o vrednostih parametrov.

Guide a sheepdog using artificial intelligence

ABSTRACT

In the wild, a single shepherd can gather and lead a herd of hundreds of sheep, and people want to collect, lead or divert other species of animals as well. In the vicinity of airports, aircraft often have problems with flocks of birds, causing them damage to their engines or even crashes. Many have already researched various models of guidance using artificial intelligence in order to be able to tackle similar problems with robots or drones. The use of smart shepherds could also come in handy when collecting spilled oil on the water or when leading a panicked crowd of people to safety in crisis situations.

In this part, we study different models of sheep movement, develop a management model and further improve it with artificial intelligence methods. We propose solutions to various problems that arise in learning. Namely, we want to develop a model according to which shepherds transport the entire herd to the stable as quickly as possible. We first add the participation of several shepherds to the basic management model, then we find good model parameters for the selected herd sizes and number of shepherds. Finally, we develop a model in which the shepherd decides on the values of parameters on the basis of selected information.

Math. Subj. Class. (2020): 68T20, 68T42, 70E55, 70E60

Ključne besede: umetna inteligenca, problem vodenja ovčarja, sodelovanje, genetski algoritmi, spodbujevano učenje, agenti

Keywords: artificial intelligence, shepherding problem, cooperation, genetic algorithms, reinforcement learning, agents

1 Uvod

Živali se pogosto zbirajo in gibljejo v bolj ali manj strnjeneh skupinah. Skupini sesalcev iste vrste, posebej iz reda kopitarjev, rečemo čreda, skupina ptic ali rib je jata. Sicer pa se živali zbirajo tudi v tako imenovane trope, roje ipd. Z zbiranjem v skupino je posamezna žival bolj varna pred plenilci in se tudi lažje pari za ceno večje opaznosti skupine v primerjavi s posamezno živaljo, hitrejšim prenosom bolezni znotraj skupine ter za ceno manjše količine hrane na posamezno žival v njeni okolini.

Človek je za varovanje, zbiranje in vodenje črede ovc, da se je ta varno in hitreje premikala po pašniku ter tako varno popasla večje območje, udomačil pse ovčarje. Ovčarjevo obnašanje je pravzaprav prevzgrojeno plenilsko obnašanje. V Sloveniji je edina avtohtona pasma kraški ovčar ali kraševec, ki ga je omenil že Janez Vajkard Valvasor v knjigi Slava Vojvodine Kranjske. Je pa kraški ovčar po značaju bolj pastirski pes, kar pomeni, da je umirjen, a ves čas pozoren na nevarnosti, saj bolj kot zbira čredo varuje¹. Ovčarji so sami sposobni zbrati in voditi veliko čredo, lahko celo več sto ovc, saj jim pri tem ključno pomaga močan čredni nagon pri ovcah in strah ovc pred njimi. V Novi Zelandiji so leta 2020 Boston Dynamics poskusili voditi ovce s pomočjo robotskega psa² in ugotovili, da se ovce podobno kot na psa odzivajo tudi na njihovega robota. Ta robot deluje brez človeškega sodelovanja, ampak ga za vodenje črede niso posebno izurili, mi pa si želimo najti uspešen algoritem, ki bo sposoben učinkovito pripeljati vse ovce v stajo v čim krajšem času.

Podobno situacijo, kjer se skupina živali, ljudi ali drugih delcev (z eno besedo agentov) giblje v podobni smeri, opazimo tudi drugje v naravi in včasih bi nam koristilo, da bi imeli udomačeno žival tudi za druge podobne probleme. Ideja je, da bi vlogo ovčarjev lahko nadomestili z uporabo robotov ali dronov. Zgledovali bi se po psih ovčarjih. Uporaba pametnih ovčarjev robotov, razvith na podoben način, bi lahko prav prišla pri preusmerjanju jate ptic v okolici letališč, zbiranju razlitr nafte na vodi ali pri vodenju panične množice ljudi na varno v kriznih situacijah.

Na letališčih že uporablajo različne načine za preusmerjanje jate ptic v okolici letališč. Jata ptic namreč lahko ob trku poškoduje letalo, kar se večinoma zgodi ob vzletanju ali pristajanju, saj večina ptic leti nizko, kar lahko vodi celo do strmoglavljenja letal. V ZDA naj bi bilo med letoma 1990 in 2010 že samo prijavljenih trkov s pticami kar 121 000, na svetu pa naj bi bilo med letoma 1988 in 2012 najmanj 200 smrti zaradi tovrstnih nesreč³. Leta 2019 je v Etiopiji zaradi ptic v eni sami letalski nesreči umrlo 157 ljudi⁴. Najbolj znana nesreča s srečnim izidom pa je bila

¹O kraševcu, dostopno na <http://www.dlvkos.si/o-krasevcu/> [ogled 1. 2. 2021]

²A robot sheepdog? ‘No one wants this,’ says one shepherd, dostopno na <https://www.theverge.com/2020/5/22/21267379/robot-dog-rocos-boston-dynamics-video-spot-shepherd-reaction> [ogled 1. 2. 2021]

³Da ptica ne bi sklatila jeklene ptice, dostopno na <https://www.24ur.com/novice/slovenija/da-ptica-ne-bi-sklatila-jeklene-ptice.html> [ogled 1. 2. 2021]

⁴Preiskava razkrila: Nesrečo Boeinga 737, v kateri je umrlo 157 ljudi, povzročile ptice!, dostopno na <https://nova24tv.si/svet/preiskava-razkrila-nesreco-boeinga-737-v-kateri-je-umrlo-157-ljudi-povzrocile-ptice/> [ogled 1. 2. 2021]

Čudež na Hudsonu v New Yorku leta 2009, ko je v nesreči dobro minuto po vzletu prišlo do trka z jato ptic. Preživelih je vseh 155 potnikov⁵. Pilot je kljub odpovedi obeh motorjev pristal in to kar na reki Hudson. O tem dogodku je posnet tudi film Sully,⁶ imenovan po tem pilotu.

Da bi ptice vodili brez sodelovanja človeka, je korejska raziskovalna skupina že razvila algoritem za preusmerjanje jate ven iz območja letališča⁷ ⁸ [10]. Mi pa se bomo v tem delu osredotočili na vodenje črede ovc v stajo z uporabo umetne inteligence. Z uporabo umetne inteligence bi se ovčar robot lahko učil tudi tekom dela z vrsto živali, ki se obnaša kako drugače.

1.1 Organizacija dela

Različni avtorji so se že lotili modeliranja gibanja ovc in psa ovčarja kot agentov. V tem delu si bomo najprej ogledali dva modela gibanja ovc, ki so ju predstavili avtorja Strömbom s sodelavci [17] in Ginelli s sodelavci [6]. Prvi model bomo tudi nekoliko spremenili, da dobimo bolj naravno gibanje in tako skupno tri modele gibanja ovc. Obnašanje agentov je po vseh treh modelih dovolj različno, da bomo s tem lahko preizkusili robustnost algoritmov vodenja psa ovčarja.

Ogledali si bomo model gibanja psa ovčarja, ki so ga predlagali Strömbom in sodelavci [17]. Modelu smo dodali tudi nekaj izboljšav in sodelovanje več psov z deljenjem črede glede na Voronoieve celice. Da bi uspeh algoritma še izboljšali v smislu hitrosti in uspešnosti vodenja ovc v stajo, smo parametre modela nastavili s pomočjo genetskega algoritma, ki temelji na ideji evolucije. Parametre smo nastavili v odvisnosti od velikosti črede, modela gibanja ovc in števila psov ovčarjev.

Ker pa se število ovc na pašniku sčasoma spreminja, ko ovce pridejo v stajo, želimo parametre prilagajati glede na trenutno in ne le glede na začetno stanje. Z uporabo spodbujevanega učenja (umetna inteligenco), natančneje PPO algoritma z nevronsko mrežo, si želimo dobljeni algoritem še izboljšati.

Na koncu pa si bomo ogledali še program *iOvčar IZIDOR* za izvajanje simulacij in učenja, ki smo ga naredili v programu Unity. Za potrebe magistrske naloge smo uporabili tudi paket ML Agents, ki v Unityu skrbi za učenje in optimizacijo vedenja različnih entitet. Program je napisan v programskem jeziku C#. Predstavitev programa sledi še pregled rezultatov in primerjava modelov vodenja.

⁵Plane crashes in Hudson river in New York, dostopno na <https://www.theguardian.com/world/2009/jan/15/us-airways-plane-crash-new-york-hudson-river> [ogled 1. 2. 2021]

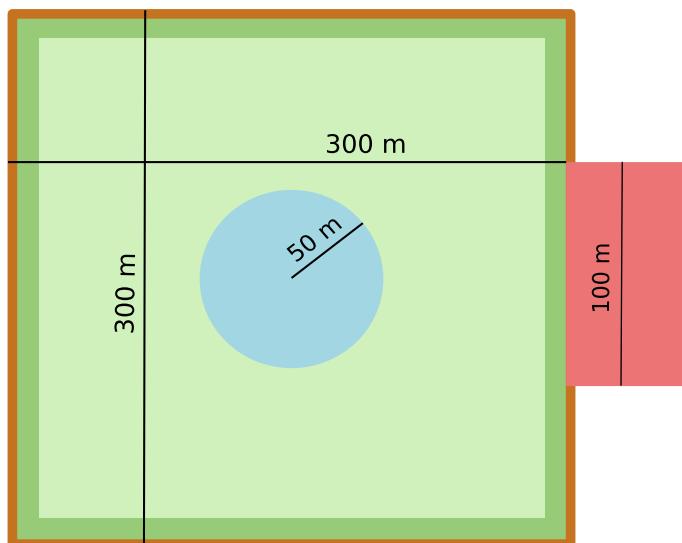
⁶Sully, dostopno na <https://www.warnerbros.com/movies/sully> [ogled 1. 2. 2021]

⁷'Sheepdog robot' herds birds away from flight paths, dostopno na <https://www.imperial.ac.uk/news/187630/sheepdog-robot-\herds-birds-away-from/> [ogled 1. 2. 2021]

⁸Drone-Based Bird-Herding Is Taking Off, dostopno na <https://insideunmannedsystems.com/drone-based-bird-herding-\is-taking-off/> [ogled 1. 2. 2021]

2 Obnašanje ovc

V tem poglavju si bomo ogledali dva modela obnašanja ovc avtorjev Strömboma s sodelavci [17] in Ginellija s sodelavci [6]. Prvi model bomo nekoliko predelali in s tem naredili še tretji model. Za učenje in testiranje ovčarjev bomo uporabili ovce po vseh treh modelih in tako preverili robustnost metode vodenja za različne vrste agentov. Ovce so na začetku naključno enakomerno neodvisno porazdeljene v krog s polmerom 50 m na sredini kvadratnega pašnika s stranico dolgo 300 m. Na sredini ene stranice je 100 m širok vhod na cilj, ki mu bomo rekli staja. Psi ovčarji bodo v simulacijah postavljeni naključno enakomerno neodvisno kamor koli na pašnik vsaj 5 m od ograje. Na sliki 1 si lahko ogledamo opisano postavitev na začetku simulacije. Ovce so porazdeljene enakomerno na modrem območju in ovčarji enakomerno na svetlo zelenem območju vsaj 5 m od ograje z namenom, da na ovčarjevo gibanje ograja ne vpliva že takoj na začetku simulacije.



Slika 1: Prikaz območij simulacijskega okolja s pašnikom, stajo in območji porazdelitve agentov.

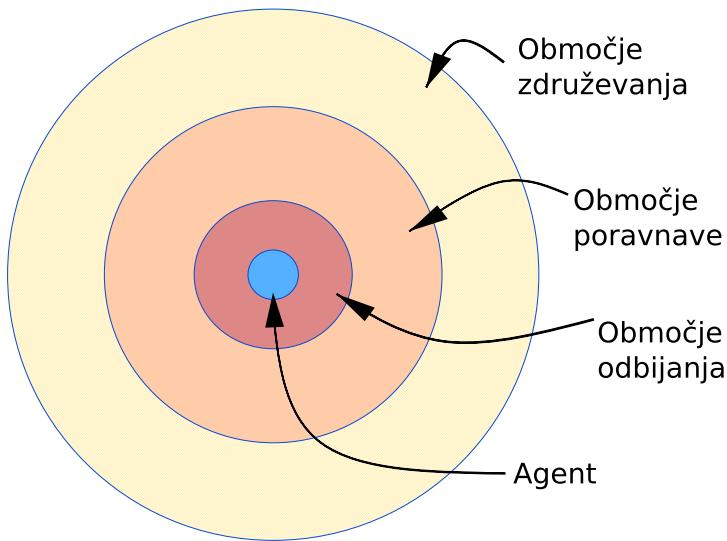
2.1 Strömbomov model

Prvi in najenostavnnejši model gibanja ovc, ki si ga bomo ogledali, temelji na modelu, pri katerem ima vsak agent tri območja, ki ustrezajo trem osnovnim težnjam agenta kot predlagano v [13] (slika 2). Osnovne težnje agenta so:

1. Agent se od ostalih agentov, ki so mu preblizu, odmika za preprečevanje trkov. Območje odbijanja je izmed vseh najmanjše.
2. Agent poravna svojo smer gibanja z drugimi agenti, ki so mu dovolj blizu. Območje poravnave je večje od območja odbijanja.

3. Agent se približuje agentom, ki so v njegovem zaznavnem ali vidnem polju. Območje združevanja je največje.

Od vrste živali je odvisna velikost posameznega območja in moč posamezne težnje. Model je zasnovan za ptice, a ga lahko uporabljam tudi v ravnini. Strömbom je s sodelavci modelu dodal še območje strahu za beg pred grožnjo oziroma ovčarjem in zanemaril območje poravnave. Območje strahu je celo večje od vidnega polja, saj ovca že na daleč psa zazna tudi zaradi njegovega oglašanja, čeprav ta še ni v njenem vidnem polju.



Slika 2: Prikaz območij modela po [13].

2.1.1 Odbojna sila

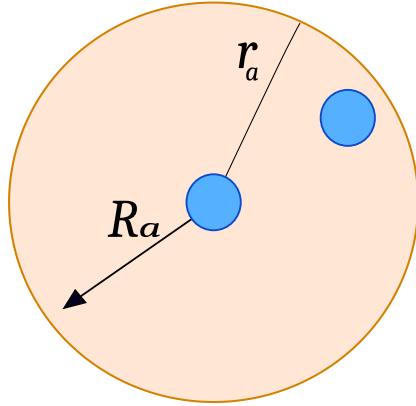
Odbojna sila služi preprečevanju trkov med agenti. Za vsako ovco, ki ima v območju odbijanja vsaj eno ovco, izračunamo odbojno silo \mathbf{R}_i^a , ki je enaka vsoti normaliziranih vektorjev stran od vsake izmed ovc v območju odbijanja z radijem r_a .

$$\mathbf{R}_i^a = \sum_{j \neq i, \|\mathbf{A}_i - \mathbf{A}_j\| < r_a} \frac{\mathbf{A}_i - \mathbf{A}_j}{\|\mathbf{A}_i - \mathbf{A}_j\|}, \quad (2.1)$$

kjer je \mathbf{A}_j lokacija j -te ovce. Primer si lahko ogledamo na sliki 3.

2.1.2 Sila združevanja

Ovca želi biti vedno blizu črede, saj se tako počuti varnejše. Sila združevanja je kar enaka normaliziranemu vektorju usmerjenemu proti središču dela črede, ki je v njenem vidnem polju oziroma v območju združevanja s polmerom r_d , kot je predlagal avtor diplomskega dela [15], da bi se izognil situaciji, ko bi se ovce žezele približevati



Slika 3: Odbojna sila od ovce v območju odbijanja.

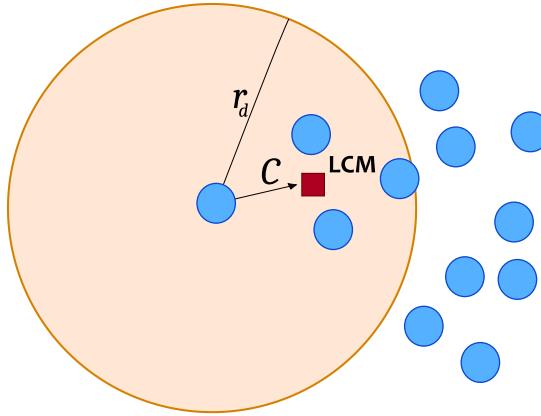
ovcam, ki niso v njihovem vidnem polju, saj so avtorji lokalni center izračunavali na podlagi najbližjih k sosedov za nek izbran k . V našem primeru pa se lahko zgodi, da ovca vidi vse ali celo nobene druge ovce. Lokalni center mase (LCM) ali lokalno težišče tako izračunamo kot

$$\text{LCM}_i = \frac{1}{k} \sum_{\|\mathbf{A}_i - \mathbf{A}_j\| < r_d} \mathbf{A}_j, \quad (2.2)$$

kjer je k število ovc v območju združevanja ovce i vključno z njo samo in se k skozi čas spreminja. Tedaj je smer sile združevanja enaka

$$\mathbf{C}_i = \text{LCM}_i - \mathbf{A}_i. \quad (2.3)$$

Shematski prikaz sile združevanja si lahko ogledamo na sliki 4.



Slika 4: Sila združevanja v smeri proti lokalnemu centru mase ovc v območju združevanja.

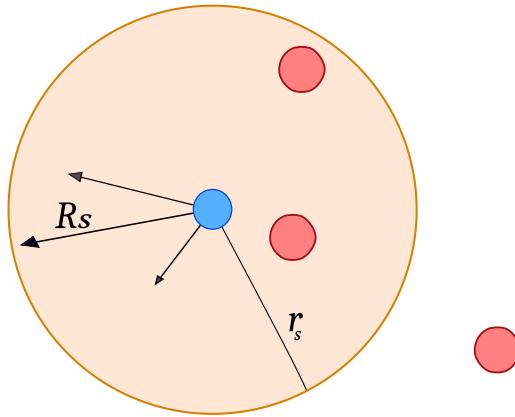
2.1.3 Sila bega pred ovčarjem

Ko ovca zazna grožnjo se želi od nje umakniti. Ovca beži od vseh ovčarjev, ki so v njenem območju bega s polmerom r_s . Smer bega izračunamo kot vsoto norma-

liziranih smeri stran od ovčarjev pomnoženih s faktorjem strahu pred posameznim ovčarjem

$$\mathbf{R}_s = \sum_{\|\mathbf{A}_i - \mathbf{S}_j\| < r_s} \frac{\mathbf{A}_i - \mathbf{S}_j}{\|\mathbf{A}_i - \mathbf{S}_j\|} \left(\frac{\|\mathbf{A}_i - \mathbf{S}_j\| - r_s}{r_s} \right)^2, \quad (2.4)$$

kjer je \mathbf{S}_j lokacija j -tega psa ovčarja. Faktor strahu pred posameznim ovčarjem služi temu, da se ovca bolj boji ovčarja, ki ji je bližje. Za prikaz si oglejmo sliko 5.



Slika 5: Sila bega pred dvema ovčarjema v območju strahu. Tretji ovčar je ne ogrožen.

2.1.4 Izračun premika

Smer gibanja \mathbf{H}'_i izračunamo kot uteženo vsoto vseh izračunanih normaliziranih vektorjev izračunanih po formulah (2.1), (2.3), (2.4), normalizirane smeri v prejšnjem koraku $\hat{\mathbf{H}}_i$ in naključnega enotskega vektorja ϵ po formuli

$$\mathbf{H}'_i = h\hat{\mathbf{H}}_i + c\hat{\mathbf{C}}_i + \rho_a \hat{\mathbf{R}}_i^a + \rho_s \hat{\mathbf{R}}_i^s + e\epsilon_i, \quad (2.5)$$

pri čemer izbrane uteži zadoščajo pogoju $\rho_a > c > \rho_s > h > e$, da ovce ostanejo na razdalji in hkrati skupaj kljub grožnji in šumu. Ničelnih vektorjev ne normaliziramo, da se med drugim izognemo begu, ko noben ovčar ni v območju strahu. Stara smer je uporabljena z namenom bolj naravnega gladkega gibanja brez ostrih zavojev. Šum z velikostjo e je prisoten le z verjetnostjo p . Agent se ves čas giblje s konstantno hitrostjo v . Tako se ovca premakne na točko $\mathbf{A}'_i = \mathbf{A}_i + \Delta t v \hat{\mathbf{H}}'_i$, kjer je Δt spremembra časa od prejšnjega koraka in $\hat{\mathbf{H}}'_i$ normalizirana smer izračunana v (2.5), prilagojena za izogibanje ograji na način, opisan v naslednjem razdelku.

Vrednosti parametrov, ki smo jih uporabili, si lahko ogledamo v tabeli 1. Večino vrednosti smo povzeli po Strömbomu in sodelavcih [17] ter Demšarju in sodelavcih [4]. Zadnja dva parametra nastopata v popravljenem modelu, opisanem v nadaljevanju.

Oznaka	Opis parametra	Uporabljena vrednost
r_a	Polmer območja odbijanja	2 m
r_s	Polmer območja združevanja	30 m
r_d	Polmer območja strahu	15 m
ρ_a	Relativna moč odbojne sile	2
c	Relativna moč sile združevanja	1,05
ρ_s	Relativna moč sile bega pred ovčarjem	1
h	Relativna moč prejšnje smeri	0,5
e	Relativna moč šuma	0,3
p	Verjetnost nastanka šuma	0,05
v	Hitrost premikanja	5 m/s
r_f	Razdalja za izogibanje ograji	20 m
γ	Stopnja izogibanja ograji	0,01
v_p	Hitrost premikanja v odsotnosti grožnje	0,5 m/s
ϕ	Faktor dovoljene spremembe smeri	$\pi/20$ rad

Tabela 1: Parametri Strömbomovega modela. Spodnja dva parametra potrebujemo samo za popravljen Strömbomov model opisan v poglavju 2.2.

2.1.5 Izogibanje ograji

Ovca se od vsakega dela ograje, ki ji je bliže od r_f , želi odmakniti. Smeri $\hat{\mathbf{H}}'_i$ zato prištejemo odbojno silo v smeri stran od ograje z velikostjo $\gamma(r_f - d_{i,j})/r_f$, kjer je γ stopnja izogibanja ograji in $d_{i,j}$ oddaljenost i -te ovce do j -tega dela ograje, kakor so predlagali Demšar in sodelavci [4]. Smer gibanja ponovno normaliziramo.

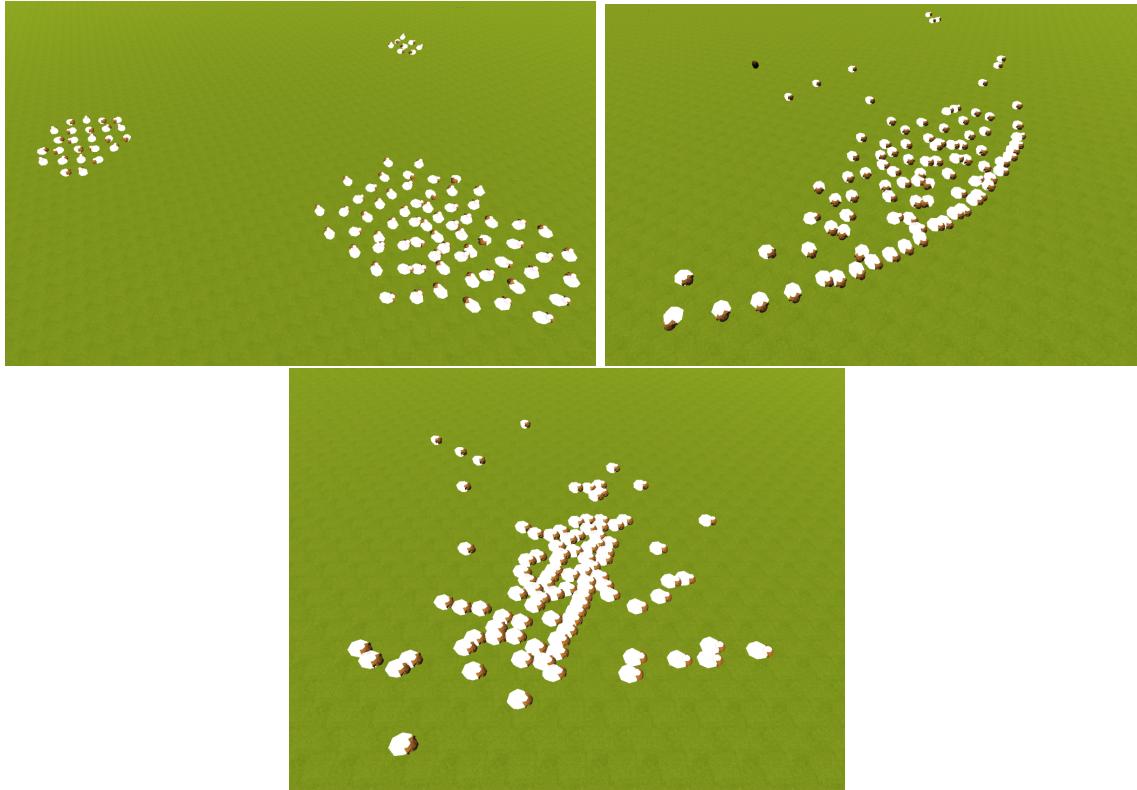
2.2 Popravljen Strömbomov model

Ovce se po opisanem Strömbomovem modelu ne gibljejo najbolj naravno. Oblikujejo se manjše črede, ki se združijo, ko so dovolj blizu in izgledajo bolj podobne kapljami olja na vodi kakor pa čredi ovc (slika 6). Čreda ima kristalno in pretirano strukturirano obliko, saj ovce striktno ohranjajo določeno medsebojno razdaljo, ko so si enkrat dovolj blizu. Ker pa se ne bi smelete premikati za ohranjanje te razdalje, se le tresejo okrog svoje osi. Ta problem smo rešili z določanjem največjega dovoljenega kota spremembe smeri v časovnem koraku. Največji dovoljen kot definiramo kot $\phi_v = \phi/\sqrt{v_t}$, kjer je ϕ faktor največje dovoljene spremembe smeri in v_t trenutna hitrost. Večja kot je trenutna hitrost, manjše spremembe smeri dovolimo. Velikost spremembe smeri omejimo s ϕ_v . Tako ovci stabiliziramo smer gibanja po tem, ko je nova smer izračunana. Na ta način smo rešili problem med begom, saj gredo vse ovce v podobno smer z manj tresenja.

Ker pa so ovce ob tej fiksni hitrosti in majhnem dovoljenem kotu začele krožiti kot roj mušic, kadar grožnja ni bila prisotna, in je čreda zato obstala na mestu, smo uvedli spremenljivko v_t , ki predstavlja trenutno hitrost. Ovca se zdaj giblje s

hitrostjo enako $v_t = \frac{3}{10}v_z + \frac{7}{10}v_{t-1}$, kjer je v_{t-1} hitrost v prejšnjem koraku in v_z želena hitrost, ki je enaka v v času prisotne grožnje in v_p sicer med pašo. Tako se ovca v času odsotnosti grožnje premika počasneje in ne začne krožiti, gibanje pa postane bolj naravno. Prednost dodatnega modela je predvsem v tem, da je dovolj drugačen od ostalih dveh opisanih v poglavjih 2.1 in 2.3. Tako bomo lahko testirali robustnost pristopa vodenja s psi ovčarji za primer, ko bi želeli voditi drugačno vrsto agentov (živali, ljudi ali delcev).

Že Strömbomov model smo nekoliko spremenili, maš model mu ni več pretirano podoben, ampak ga vseeno imenujmo *Popravljen Strömbomov model*, saj smo ohranili osnovno idejo. Čeprav je tu gibanje bolj naravno kot v osnovnem modelu, pa ni dovolj podobno realnemu gibanju ovc.



Slika 6: Levo zgoraj vidimo značilno kristalno strukturo črede po Strömbomovem modelu, ki se ohranja celo med begom pred ovčarjem. Desno zgoraj opazimo nekoliko bolj naravno strukturo pri popravljenem Strömbomovem modelu. Ovce se po popravljenem modelu gibljejo bolj naravno, a ne dovolj podobno realnemu gibanju ovc v naravi. Na spodnji sliki je primer črede med begom v primeru Ginellijevega modela. Gre za naprednejši model, osnovan na dejanskih podatkih o gibanju ovc iz narave. Ta model je opisan v naslednjem podpoglavlju.

2.3 Ginellijev model

Osnovni Strömbomov model ima določene pomanjkljivosti, saj so ovce ves čas preveč oddaljene in se bolj ne približajo druga drugi niti med begom, kar pa se v naravi navadno ne zgodi. Strömbomov model ne temelji na posnemanju dejanskega gibanja ovc, ampak je le enostaven model za testiranje njihove ideje vodenja psa ovčarja. V članku avtorji namreč predstavijo model vodenja psa ovčarja, ki si ga bomo ogledali v naslednjem poglavju ter mu dodali nekaj izboljšav.

Ginelli je s sodelavci v svojem članku [6] opisal bolj kompleksen model gibanja ovc, ki so ga razvili in testirali z eksperimentalnimi posnetki črede ovc v kvadratni ogradi in so se s tem bolj približali dejanskemu obnašanju ovc. Opazili so nekakšno utripanje črede. Čreda se je ponavljajoče širila po pašniku in se ob naključnem času na hitro zbrala tudi brez prisotnosti grožnje. S tem so ovce hkrati popasle večje območje in ostale varno skupaj. Vsaka ovca je imela vedno več prostora, dokler ni kakšna ovca ob robu stekla proti notranjosti črede, kar je s črednim nagonom povzročilo tek večine ovc in so se tako hitro zbrale ter nekoliko premaknile center črede. Ta model pa za razliko od Strömbomovega modela upošteva poleg lokacije ovc \mathbf{A}_i^t tudi njihovo smer Θ^t in stanje obnašanja q_i^t . Stanja obnašanja so mirovanje, hoja in tek, ki jih označimo kot 0, 1 in 2. Ovca med mirovanjem ne spreminja lokacije in smeri, sicer pa ju prilagaja glede na bližnje ovce.

2.3.1 Izračun smeri

Agent se premakne po enačbi

$$\mathbf{A}_i^{t+\Delta t} = \mathbf{A}_i^t + \Delta t v(q_i^t) \mathbf{s}_i^{t+\Delta t}, \quad (2.6)$$

kjer je Δt sprememba časa med korakoma simulacije, hitrost je zdaj odvisna od stanja v prejšnjem koraku in smernega vektorja $\mathbf{s}_i^t = (\cos \Theta_i^t, \sin \Theta_i^t)$, odvisnega od trenutnega kota Θ_i^t . Ob tem upoštevamo, da je hitrost v stanju teka $v(2)$ veliko večja od hitrosti med hojo $v(1)$, ki pa je večja od 0. Torej velja $v(2) >> v(1) > v(0) = 0$.

Ovca želi svojo smer poravnati s smerjo drugih ovc, ki so od nje oddaljene največ r_o , po enačbi

$$\Theta_i^{t+\Delta t} = \operatorname{Arg} \left[\sum_{j \neq i, \|\mathbf{A}_i^t - \mathbf{A}_j^t\| < r_o} s_j^t \right] + \psi_i^t, \quad (2.7)$$

kjer je Arg argument vektorja oziroma kot od baznega vektorja in ψ_i^t naključen kot z intervala $[-\mu\pi, \mu\pi]$. S tem smo dobili povprečno smer okoliških ovc z nekaj šuma. V stanju teka pa je ovca pozorna le na ovce, ki so prav tako v stanju teka in so največ r_d oddaljene od nje. Pri tem upošteva njihovo smer gibanja, razdaljo do njih in v kateri smeri so. V stanju teka se smer gibanja izračunava na sledeč način:

$$\Theta_i^{t+\Delta t} = \operatorname{Arg} \left[\sum_{j \in \mathcal{V}_i} (\delta \mathbf{s}_j^t + \beta f(r_{i,j}^t) \mathbf{e}_{i,j}^t) \right], \quad (2.8)$$

kjer je $\mathcal{V}_i = \{j \neq i, \|\mathbf{A}_i^t - \mathbf{A}_j^t\| < r_d, q_j^t = 2\}$ množica vseh okoliških ovc v stanju teka, δ utež posnemanja smeri okoliških ovc, $\mathbf{e}_{i,j}^t$ enotski vektor od i -te do j -te ovce z utežjo β in $f(r_{i,j}^t)$ faktor privlačno-odbojne sile z ravnovesno razdaljo r_e , odvisen od razdalje med i -to in j -to ovco. Ta faktor je od razdalje r odvisen na sledeč način:

$$f(r) = \min(1, \frac{r - r_e}{r_e}). \quad (2.9)$$

To povzroči, da se ovca približa ostalim ovcam na udobno razdaljo r_e in teče v isti smeri kot one. Zaradi poravnave z bližnjo okolico se ob večji gneči ovce postavijo v vrste, kar so za podobne primere že dokazali avtorji članka [3], kjer matematično modelirajo obnašanje podolgovatih delcev, ki se zaradi tresenja in posledičnih trkov z bližnjimi delci poravnajo z njimi. Ovca smer še dodatno spremeni v bližini ograje na enak način kot v ostalih dveh modelih gibanja ovc, kakor smo opisali v poglavju 2.1.5 na strani 7.

Vrednosti parametrov, ki smo jih uporabili za ta model, si lahko ogledamo v tabeli 2. Večino vrednosti so predlagali avtorji člankov [6] in [4].

Oznaka	Opis parametra	Uporabljena vrednost
$v(1)$	Hitrost hoje	0,5 m/s
$v(2)$	Hitrost teka	5 m/s
$\tau_{0 \rightarrow 1}$	Stopnja prehoda iz mirovanja v hojo	70
$\tau_{1 \rightarrow 0}$	Stopnja prehoda iz hoje v mirovanje	16
$\tau_{0,1 \rightarrow 2}$	Stopnja prehoda v tek	N
$\tau_{2 \rightarrow 0}$	Stopnja prehoda iz teka v mirovanje	N
d_R	Utež prehoda v tek	31,6
d_S	Utež prehoda iz teka	2,1
r_e	Ravnovesna razdalja med ovcami	1 m
r_o	Razdalja za poravnavo z ovcami med hojo	1 m
r_d	Razdalja za poravnavo med tekom	15 m
r_s	Razdalja za zaznavo psa ovčarja	30 m
α	Utež intenzivnosti oponašanja	15
β	Moč privlačno-odbojne sile	0,8
δ	Moč poravnave med tekom	4
μ	Razpon šuma	0,13
γ	Moč strahu pred ovčarjem	0,1

Tabela 2: Parametri Ginellijevega modela, kjer je N število ovc.

2.3.2 Sprememba stanja obnašanja

Ovca spremeni svoje stanje obnašanja q_i^t glede na okoliške ovce in grožnje. Ovca z večjo verjetnostjo preklopi v neko stanje, če je v njeni okolici več ovc v tem izbranem stanju. Definirajmo verjetnost prehoda iz mirovanja v hojo $p_{0 \rightarrow 1}$ in verjetnost

prehoda iz hoje v mirovanje $p_{1 \rightarrow 0}$ za ovco i ob času t na podoben način

$$p_{a \rightarrow b}(i, t) = 1 - \exp\left(-\frac{1 + \alpha n_b^t(i)}{\tau_{a \rightarrow b}}\right), \quad (2.10)$$

kjer je α utež intenzivnosti obnašanja, $n_b^t(i)$ število ovc v stanju gibanja b ob času t v okolini ovce i , oddaljenih največ r_d , in $\tau_{a \rightarrow b}$ stopnja prehoda iz stanja a v stanje b .

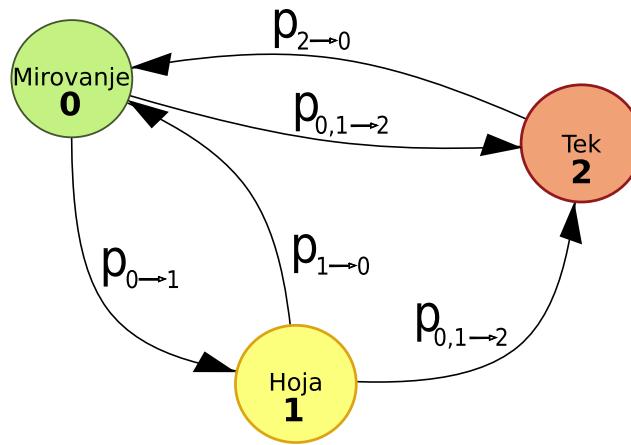
Avtorji so model poenostavili s tem, da je verjetnost prehoda iz mirovanja ali hoje v tek enaka in da ovca iz stanja teka vedno najprej preide v stanje mirovanja. Definirajmo še ti dve verjetnosti. Verjetnost prehoda iz mirovanja ali hoje v tek je enaka

$$p_{0,1 \rightarrow 2}(i, t) = 1 - \exp\left(-\frac{1}{\tau_{0,1 \rightarrow 2}} \left[\frac{l_i^t}{d_R} (1 + \alpha m_R^t(i)) \right]^\delta\right), \quad (2.11)$$

kjer je $\tau_{0,1 \rightarrow 2}$ stopnja prehoda v tek, l_i^t povprečna razdalja do metrično sosednjih ovc, d_R utež prehoda v tek, m_R^t število metrično sosednjih ovc v stanju teka in $\delta > 0$ moč poravnave med tekom. Ovca je metrično sosednja, če je na prvi Voronojevi lupini. S tem bolj razkropljena čreda prične teči z večjo verjetnostjo kot manj razpršena zaradi vpliva l_i^t , pozitiven δ pa poskrbi za konveksen vpliv razpršenosti na pričetek teka. Ovca se v stanju teka ustavi z verjetnostjo

$$p_{2 \rightarrow 0}(i, t) = 1 - \exp\left(-\frac{1}{\tau_{2 \rightarrow 0}} \left[\frac{d_S}{l_i^t} (1 + \alpha m_S^t(i)) \right]^\delta\right), \quad (2.12)$$

kjer je $\tau_{2 \rightarrow 0}$ stopnja prehoda iz teka v mirovanje, d_S utež prehoda iz teka in $m_S^t(i)$ število metrično sosednjih ustavlajočih se ovc. Opazimo, da ima tu povprečna razdalja do sosedov obraten vpliv kot v enačbi (2.11). Shemo sprememb stanj si lahko ogledamo na sliki 7.



Slika 7: Možni prehodi med stanji gibanja.

2.3.3 Beg pred ovčarjem

Ker se izvirni članek ne ukvarja z vodenjem črede, njihov model ne vsebuje odziva na prisotnost grožnje. Za naše potrebe smo modelu gibanja ovce dodali strah pred

ovčarjem. Že pri Strömbomovem modelu smo pri tem upoštevali možnost vodenja z več psi ovčarji.

Ovca naj gre v primeru prisotnosti grožnje na razdalji največ r_s vedno v stanje teka. Smer bega pred ovčarji v primeru njegove prisotnosti določimo kot

$$\Theta_i^t = \text{Arg}[\hat{\mathbf{s}}_i^t + \gamma \hat{\mathbf{R}}_s], \quad (2.13)$$

kjer je $\hat{\mathbf{s}}_i^t$ normirani smerni vektor s kotom, izračunanim po enačbi (2.8), γ moč strahu pred ovčarjem in $\hat{\mathbf{R}}_s$ normirana sila bega, izračunana enako kot za Strömbomov model v enačbi (2.4). Ovco je tako bolj strah bližje grožnje, a ji je smer bega celotne črede bolj pomembna kot smer ovčarja.

3 Ročno razviti model vodenja psa ovčarja

V tem poglavju si bomo ogledali model vodenja posameznega psa ovčarja in skupine psov ovčarjev. Model temelji na modelu, ki ga je predstavil Strömbom s sodelavci v članku [17]. Dodali smo nekaj izboljšav za bolj učinkovito vodenje. Model smo nadgradili s sodelovanjem skupine psov, da si med seboj delijo delo in ga tako opravijo hitreje in tudi bolj zanesljivo kot en sam ovčar.

Z vodenjem črede so se poleg avtorjev izbranega modela ukvarjali tudi drugi. Avtorja Bennett in Trafankowskista [2] sta med seboj primerjala svojo metodo in več različnih metod drugih avtorjev.

3.1 Stanja obnašanja ovčarja

Ovčar naj bi ovce pripeljal v stajo, a vodenje vsake ovce posebej bi bilo neučinkovito in nesmiselno, saj lahko izkoristimo dejstvo, da se ovce želijo združevati v čredo. Ker je ta čreda za vodenje lahko preveč razpršena, jo mora ovčar pred vodenjem najprej zbrati skupaj. Odločiti se mora za zbiranje ali vodenje črede. Kot predlagano v članku [17] ovčar zbira ovce, ko je vsaj ena ovca od globalnega težišča oziroma globalnega centra mase (GCM) celotne črede oddaljena več kot

$$f(N) := r_a \sqrt{N}, \quad (3.1)$$

kjer je N število ovc na pašniku in r_a faktor za dovoljeno velikost črede. Tako dovoljena velikost črede raste sorazmerno s površino, ki jo čreda potrebuje za vzdrževanje razdalje med ovčami. Dovolj velik faktor r_a čredi dovoljuje tudi nepravilne oblike črede. Ta ne sme biti prevelik, da strah pred ovčarjem doseže dovolj velik del črede, sicer ovčar ne more premakniti celotne črede, ampak le prestraši nekaj njemu najbližjih ovcev. Pri tem GCM izračunamo kot povprečno lokacijo vseh ovc, $\text{GCM} = \frac{1}{N} \sum_{i=1}^N \mathbf{A}_i$. Ker pa ima en ovčar lahko preveč težav z zbiranjem črede, saj potrebuje veliko časa za vodenje posameznih pobeglih ovc, smo ovčarju dovolili, da preklopi iz stanja zbiranja v stanje vodenja še preden so vse ovce zares zbrane, tako da ignorira nekaj pobeglih ovc.

Vodenju in zbiranju smo dodali tudi stanje naključnega premika in premika za čredo. Glede na stanje ovčar prilagaja smer in hitrost. Pri tem bomo upoštevali tudi morebitno prisotnost drugih ovčarjev.

3.1.1 Stanje zbiranja črede in stanje vodenja črede

Ko ovce niso zbrane v čredo, želi ovčar pobegle ovce pripeljati do črede, da je njena velikost znotraj želene. Ovčar gre v stanje zbiranja črede, kadar je vsaj ena ovca od GCM dlje kot $f(N)$. V primeru sodelovanja ovčarjev ovčar preklopi v stanje zbiranja le v primeru, če ima v svoji Voronoievi celici kakšno pobeglo ovco, sicer

gre v stanje vodenja črede. Idejo Voronoievih celic bomo opisali v poglavju 3.2, v primeru enega ovčarja je njegova Voronoieva celica kar enaka celotnemu pašniku.

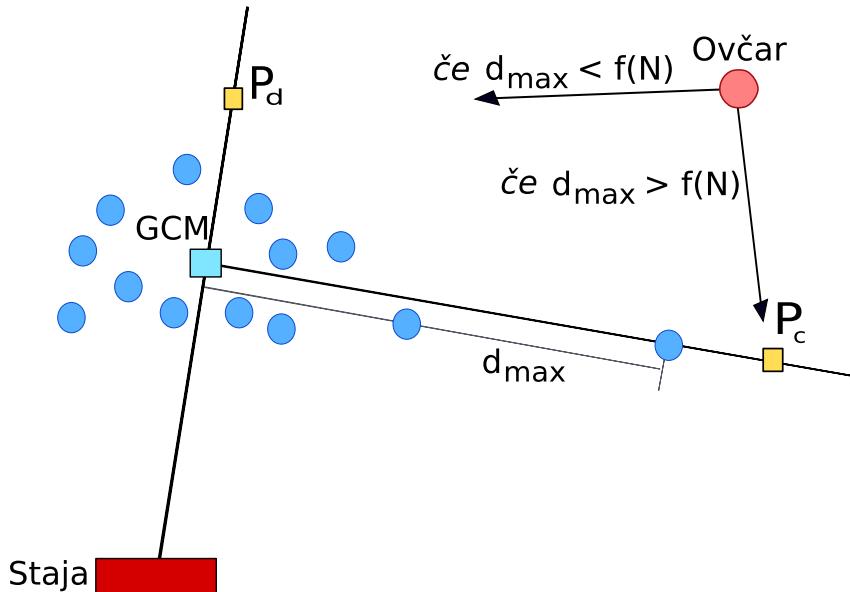
Ovčar si mora med pobeglimi ovčami v svoji Voronoievi celici izbrati eno, ki jo želi pripeljati bliže GCM. Ovčar si izbere ovco z najvišjo vrednostjo po formuli

$$r_i^{\sigma_r} / d_i^{\sigma_d}, \quad (3.2)$$

kjer je r_i razdalja i -te ovce do GCM, σ_r utež oddaljenosti ovce do črede, d_i razdalja izbranega ovčarja do i -te ovce in σ_d utež razdalje med ovco in ovčarjem. S tem bo ovčar stremel k vodenju njemu bližnjih ovc in tistih bolj oddaljenih od črede.

Pogosto se zgodi, da se ovčar postavi med čredo in stajo, kadar je pobegla ovca bliže staji kot GCM. Tedaj ovčar ignorira vse ovce, ki so stoji vsaj d_b bliže kot je stoji blizu GCM. S tem ovčar vodi tudi, kadar je kakšna ovca pobegla in le redko vodi čredo stran od staje.

Ko si ovčar v svoji Voronoievi celici izbere ovco za vodenje, ki je dovolj daleč od staje in črede ter ne predaleč od njega, se želi postaviti za njo na razdaljo d_c . S tem jo vodi proti čredi, saj je ovca med njim in GCM, kot je prikazano na sliki 8. Točko označimo kot \mathbf{P}_c .



Slika 8: Ovčar v stanju zbiranja ovc v čredo ali v stanju vodenja črede proti staji.

Ko ni nobena ovca več kot $f(N)$ oddaljena od GCM, se ovčar želi postaviti tako, da je čreda oziroma GCM med njim in stajo. Pri tem se postavi $f(N)$ stran od GCM, kot si lahko ogledamo na sliki 8. Temu stanju rečemo stanje vodenja črede. Točko označimo kot \mathbf{P}_d .

3.1.2 Ignoriranje dela črede

Včasih pa se ovčar znajde v brezizhodni situaciji, ko sam ne obvladuje celotne črede. Zato smo naredili mejo za preklop med zbiranjem in vodenjem bolj ohlapno. Če imamo eno samo pobeglo ovco, je včasih nima smisla pripeljati bliže čredi, ker bi hitreje ostalo čredo privedel do staje in naknadno vanjo pripeljali tudi to ovco. Pobegla ovca je v tem času ogrožena v primeru drugih nevarnosti, ampak v našem primeru ni volkov ali drugih groženj. Ovčar se zato odloči za vodenje črede že v primeru, ko je dovolj velik del črede največ $f(N)$ oddaljen od GCM. Dovoljeni delež pa se spreminja glede na število ovčarjev na sledeč način:

$$p = p_z / n_s^{\sigma_s}, \quad (3.3)$$

kjer je p_z utež doslednosti zbiranja, n_s število ovčarjev in σ_s moč vpliva velikosti skupine ovčarjev. Več kot je ovčarjev, manj ignoriranih ovc dopuščamo, saj je večja skupina tudi dosti bolj sposobna zbrati veliko čredo. S tem se želimo izogniti možnosti, da bi ovčar ves čas le zbiral čredo, ki bi se mu razpršila vsakič, ko bi šel po ovco, ki jo je izbral, da jo mora pripeljati v čredo.

Pri tem je težava predvsem, da se GCM ne prilagaja vodenemu delu ovc ampak vsem ovcam in se tako lahko prestavi celo tako daleč od črede, da je ovčar ne more več voditi, saj njegove grožnje ne čuti dovolj velik del črede. A to se ne zgodi tako pogosto, sploh pa ne v primeru več ovčarjev, ko je delež ignoriranih ovc tako majhen, da se GCM ne premakne dovolj na račun pobeglih ovc.

3.1.3 Stanje naključnega premika

Za primere, ko ovčar ovce pritisne v kot ograde ali pa se pojavi v kakšni drugi ravnovesni situaciji, ki ne dopušča uspeha, smo ovčarju dodali stanje naključnega premika.

V tem stanju si ovčar na vsakih $t_d + U(t_s)$ časovnih enot, kjer je t_d deterministični del in $U(t_s)$ naključni dodatek enakomerno porazdeljen na intervalu $[0, t_s]$, izbere naključno točko na pašniku, proti kateri se premika t_p časovnih enot.

3.1.4 Stanje premika za čredo

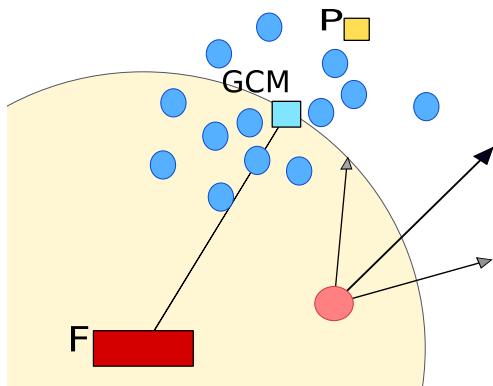
Ker pa se še vedno lahko zgodi, da se ovčar postavi bliže staji kot GCM, se ovčar v takem primeru premika proti svojemu cilju glede na stanje zbiranja ali vodenja na prilagojen način.

Smer premika izračunamo kot

$$\mathbf{s} = \sqrt{\|\mathbf{S}_i - \mathbf{F}\|}(\mathbf{S}_i - \mathbf{F}) + (\mathbf{P} - \mathbf{S}_i)\sigma_n, \quad (3.4)$$

kjer je $\mathbf{S}_i - \mathbf{F}$ vektor od staje proti ovčarju, $\mathbf{P} - \mathbf{S}_i$ vektor od ovčarja proti točki \mathbf{P}_d ali \mathbf{P}_c , odvisno od zadoščenega pogoja za vodenje ali zbiranje, in σ_n utež gibanja za

čredo. V primeru, da je ovčar od staje dlje kot GCM, pa se giblje v smeri vektorja od njega proti izbrani točki glede na stanje.



Slika 9: Ovčar v stanju premikanja za čredo.

3.1.5 Izbira hitrosti gibanja

Ovčar ima dve hitrosti med katerima gladko prehaja. Ovčar se premika s hitrostjo vodenja v_{min} , kadar je od točke \mathbf{P}_d ali \mathbf{P}_c oddaljen največ d_t in kadar je od njega najbližja ovca oddaljena največ d_0 . Sicer se giblje s hitrostjo v_{max} .

Tako ovčar upočasni v bližini ovc in črede, ker bi jih sicer razkropil, če bi se jim preveč približal.

3.2 Sodelovanje skupine ovčarjev

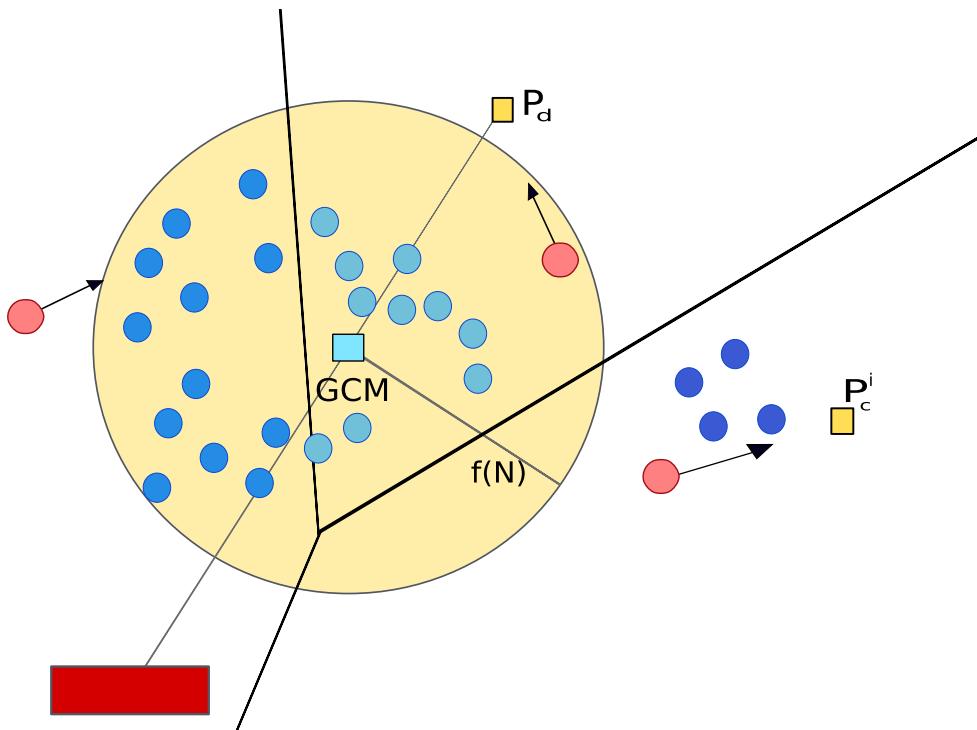
Ovčar sam pogosto ne more zbrati črede ali pa je pri tem delu zelo neučinkovit, na primer zaradi odsotnosti grožnje, ko gre po pobegle ovce. Tedaj bi bilo dobro, da bi bilo psov več, da si pomagajo in delijo delo. Če ovčarji ne sodelujejo, bodo vedno žeeli voditi isto ovco in učinka ne bo. V tem poglavju opišemo način, kako si v našem primeru ovčarji razdelijo delo.

Avtorici članka [1] sta primerjali štiri različne modele sodelovanja dveh psov ovčarjev. Pri tem se modeli razlikujejo v deljenju dela. V prvem modelu je en ovčar ves čas v stanju vodenja in eden v stanju zbiranja, v drugem modelu si vlogi izmenjujeta in nimata nikoli iste vloge, v tretjem največ en ovčar zbira čredo in v četrtem modelu preklaplja med stanjema neodvisno. Pri vseh modelih razen pri prvem si ovčarja čredo delita po premici skozi stajo in skozi GCM. Pri tem se je najbolje izkazal model, kjer sta ovčarja med stanjema neodvisno preklapljalna.

Druga možnost poleg deljenja črede bi bila tudi iskanje optimalnih točk, kamor se morajo postaviti ovčarji, in iskanje optimalne poti do teh točk. S temo problemoma se ukvarjajo avtorji člankov [8, 9, 11, 12]. Mi se bomo pri našem delu osredotočili

na delitev črede. Ker si želimo vodenja z večjo skupino psov, moramo najti tudi primerno razdelitev črede na več enot.

Čredo bomo delili na Voronojeve celice z ovčarji v središčih. To pomeni, da ima vsak ovčar svojo Voronojevo celico in v njej vse tiste ovce, ki jim je najbližje izmed vseh ovčarjev. S tem bo vsaka ovca odgovornost le enega ovčarja in ko bo šel ovčar od črede daleč proč po neko ovco, si bodo ostali ovčarji razdelili ostalo čredo. Na sliki 10 si lahko ogledamo razdelitev črede in dela, kot je opisano v poglavju 3.1.1.



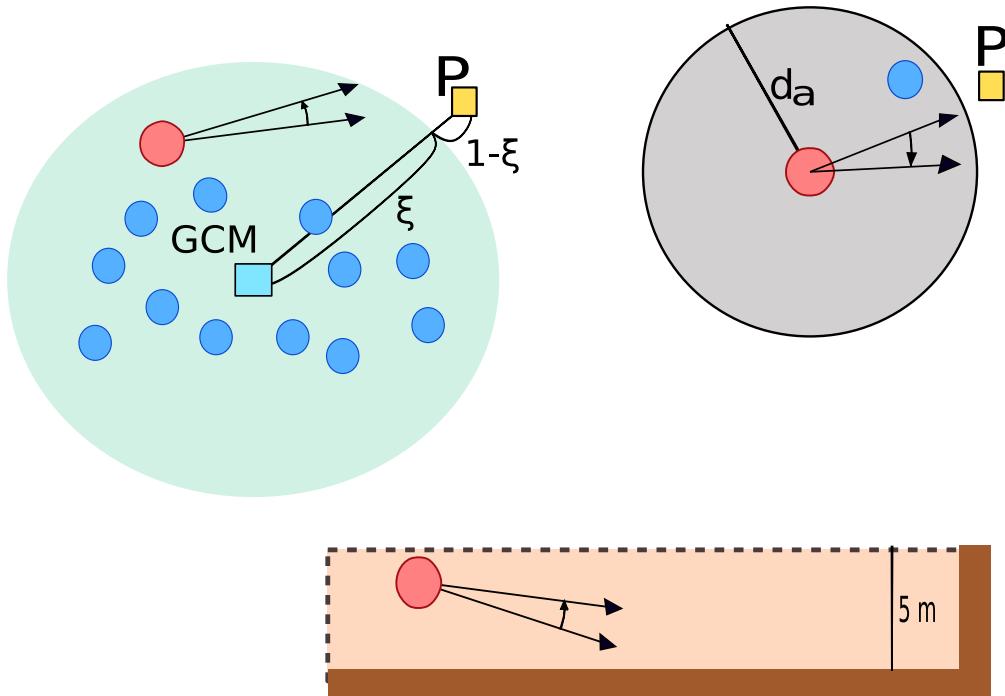
Slika 10: Razdelitev črede med tri ovčarje. Dva ovčarja sta v stanju vodenja in en je v stanju zbiranja.

Ovčarji želijo med seboj ohranjati neko minimalno udobno razdaljo. V ta namen smo med njimi dodali silo odboja z velikostjo $d_s/(r_{ij} + 1)$ v smeri od j -tega do i -tega ovčarja, ki ga močneje odbija od ovčarjev, ki so mu bliže, kjer je d_s udobna razdalja med ovčarjem in r_{ij} dejanska razdalja med njima povečana za ena v izogib deljenju z majhnim številom. Ker pa ne želimo, da bi se ovčarjem poti križale in želimo njihove poti postaviti bolj vzporedno kot zaporedno, izračunamo tudi kot med njegovo smerjo in lokacijo drugega psa. Njegovi smeri dodamo silo v smeri stran od drugega ovčarja veliko $\max(0, \sin(\omega)) \sigma_o / (r_{ij} + 1)$, kjer je ω kot, pod kakršnim gre ovčar proti drugemu ovčarju in je σ_o pomen smeri drugih ovčarjev.

3.3 Premik ovčarja

Ovčar se torej po zgoraj opisanem načinu giblje glede na stanje obnašanja in hitrost, pri čemer se giblje proti točki trenutnega cilja. Svojo smer pri tem prilagodi glede

na prisotnost drugih ovčarjev, na koncu pa njegovo smer še nekoliko popravimo. Dodamo ji šum tako, da jo zasukamo za naključen kot z intervala $[-e\pi/3, e\pi/3]$, kjer je e relativna moč šuma. Nato njegovo smer zasukamo za kot ν stran od ograje, če je tej bliže kot 5 m in se giblje proti njej. Poleg tega njegovo smer zasukamo tudi okrog ovce, če je ta bliže kot d_a , ter okrog črede v primeru, da ovčar pride središču črede bliže od ξ deleža razdalje med GCM in točko \mathbf{P} (\mathbf{P}_c ali \mathbf{P}_d). Vse tri primere kroženja si lahko ogledamo na sliki 11. Nato smer zgradimo glede na smer v prejšnjem časovnem koraku. S tem se ovčar giblje bolj naravno.



Slika 11: Ovčar zaokroži stran od ograje, ovce ali črede.

V tabeli 3 si lahko ogledamo uporabljene vrednosti parametrov in njihove meje za ostala dva modela.

Oznaka	Opis parametra	Vrednost	Interval
v_{max}	Najvišja hitrost	7,5 m/s	-
v_{min}	Hitrost v stanju vodenja	5 m/s	$[0, v_{max}]$
r_a	Faktor za dovoljeno velikost črede	2 m	$[0, 5]$
d_c	Razdalja za zbiranje	10 m	$[0, 30]$
d_a	Razdalja za zaznavo ovc na poti	20 m	$[0, 60]$
d_0	Razdalja za upočasnitve v bližini ovc	10 m	$[0, 30]$
d_f	Razdalja za upočasnitev v bližini cilja	5 m	$[0, 30]$
e	Relativna moč šuma	30 %	$[0, 50]$
t_p	Trajanje naključnega premika	3 s	$[0, 10]$
t_d	Fiksen čas do naključnega premika	60 s	$[0, 90]$
t_s	Razpon naključnega dodatnega časa	20 s	$[0, 30]$
σ_r	Pomen oddaljenosti pobegle ovce od črede	2	$[-1, 4]$
σ_d	Pomen oddaljenosti pobegle ovce od ovčarja	1	$[-1, 4]$
d_b	Ovčar bližje cilju kot čreda	2 m	$[-5, 50]$
p_z	Največji delež ignoriranih ovc	15 %	$[0, 40]$
σ_s	Vpliv števila ovčarjev na delež ignoriranih ovc	2	$[-1, 4]$
σ_n	Odpornost pred stanjem bližje cilju kot točka	2	$[0, 20]$
d_s	Udobna razdalja med ovčarji	20 m	$[0, 40]$
d_t	Razdalja za upočasnitev v bližini točke	12 m	$[0, 20]$
ξ	Odpornost pred stanjem pred točko	95 %	$[0, 1]$
ν	Zaokroževanje blizu ovc	$\pi/6$ rad/s	$[0, 2]$
σ_o	Pomen smeri drugih ovčarjev	0,1	$[-1, 1]$

Tabela 3: Parametri psa ovčarja, njihove uporabljene vrednosti za ročno razviti model in njihove meje za ostala dva modela gibanja psa ovčarja. Meje so v istih enotah kot izbrane vrednosti.

4 Model vodenja razvit z genetskim algoritmom

Če je ovčar čredi preblizu, se ovce razkropijo, če je predaleč, pa jih ne more voditi. Najboljša razdalja je odvisna od števila ovc. Prav tako verjetno obstajajo tudi boljše vrednosti preostalih parametrov modela (tabela 3) v odvisnosti od števila ovc, ovčarjev in modela gibanja ovc.

Ker je ročno razviti model pogosto neuspešen, si želimo najti optimalne vrednosti parametrov. Za vsak parameter lahko postavimo neke smiselne meje, kjer bi ta vrednost lahko bila. Ker je dobljeni prostor (produkt 21 intervalov, največja hitrost je konstantna) prevelik, moramo najti nek način iskanja najboljših vrednosti v tem prostoru. Četudi bi zvezne intervale razdelili na le 10 diskretnih vrednosti, bi morali preveriti 10^{21} kombinacij vrednosti parametrov. Če bi le vrnili ničelni izhod, bi en računalnik s 16 jedri in hitrostjo procesorja 3,5 GHz preverjal vse kombinacije več kot 580 let. Torej nam tu ne bi pomagala niti uporaba več računalnikov, saj nikakor ne bi zaključili vseh potrebnih simulacij v nekem doglednem času.

Odločili smo se za pristop z genetskim algoritmom, ki temelji na ideji naravne selekcije pri evoluciji v naravi. Genetski algoritmi so populacijski algoritmi za optimizacijske probleme. Ker je algoritem vseeno zamuden, smo simulacijo omejili na 3 minute (omejeno s številom korakov simulacije, ki ustreza 3 minutam in ne z dejanskim časom izvajanja) in jo nato prekinili, če se simulacija še ni sama zaključila z uspešnim izidom. Pri implementaciji smo se zgledovali po podrobni razlagi algoritma v [16].

Ideja genetskih algoritmov je ta, da imamo začetno populacijo naključnih genov, tej začetni generaciji pa sledijo generacije, ki naj bi bile sčasoma v povprečju vedno boljše. Pri tem moramo dobro definirati, kaj je gen in kaj pomeni, da je nek gen "boljši".

4.1 Gen in evalvacija uspešnosti gena

Vsek osebek v populaciji ima določene vrednosti vseh 21 parametrov in temu zaporedu števil pravimo gen. Tako je na primer genotip za hitrost ovčarja v stanju vodenja pri ročno razvitem modelu enak $2/3$ (le numerična vrednost), fenotip pa je njegova izražena hitrost v stanju vodenja 5 m/s . Opazimo, da se med evolucijo lahko pojavi tudi gen, ki ustreza fenotipu ročno razvitega ovčarja. Ob koncu evolucije torej lahko pričakujemo vsaj tako dober ali celo boljši gen in s tem večjo uspešnost vodenja, saj bi tudi ročno razviti model izumrl, če ne bi bil vsaj primerljiv z najboljšim genom.

Da evolucija deluje dobro, mora biti v začetnem genomu dovolj različnih genov. Če geni niso dovolj različni, bodo naslednje generacije preveč podobne prvi in odvisne od začetka simulacije. Če pa genov ni dovolj, se lahko hitro izgubi potencial, ki se skriva v posameznem genu.

Uspešnost posameznega gena moramo za potrebe simuliranja naravne selekcije nekako oceniti. Za ocenjevanje uspešnosti gena potrebujemo cenilko uspešnosti za evalvacijo gena. Pri tem upoštevamo čas potreben za simulacijo (zadnja ovca v staji), število ovc na pašniku ob koncu simulacije, oddaljenost preostanka črede od staje in čase prihodov v stajo. Funkcija naj bi bila konveksna v primeru, ko je pomembna dejanska ocenjena vrednost in ne le razvrstitev med ostalimi geni.

Naša izbrana funkcija za oceno uspešnosti posamezne simulacije je

$$\begin{aligned}\phi(N, (t_j)_{j_1}^k, \mathbf{GCM}) = & 210 \left(\frac{t_{MAX} - t_N}{t_{MAX}} \right)^2 \frac{1}{N} \sum_{j=1}^k \frac{t_{MAX} - t_j}{t_{MAX}} \\ & + \frac{1}{1 + N - k + (N - k)\|\mathbf{GCM} - \mathbf{F}\|},\end{aligned}\quad (4.1)$$

kjer je t_{MAX} časovna omejitev simulacije v sekundah, N začetno število ovc, k število ovc pripeljanih v stajo in \mathbf{F} lokacija staje. Po tej formuli dobimo večjo uspešnost za simulacije, ki prej v stajo pripeljejo večji delež črede. Tako je bolj uspešen gen, kjer je zadnja ovca prej v staji. Nekoliko manj pomembno je, da so bile ovce v povprečju prej pripeljane v stajo oziroma, da nam je ostal večji delež časa. Faktor 210 služi le normalizaciji, da je želena uspešnost med 0 in 100. Ker pa je v primeru $t_N = t_{MAX}$, ko je na pašniku še vsaj ena ovca, prvi člen enak 0, mu prištejemo še drugi člen, kjer je pomembno število preostalih ovc na travniku in je še nekoliko manj pomembna razdalja med GCM in F. S tem lahko po uspešnosti razvrstimo tudi simulacije, kjer je na pašniku ostala še vsaj ena ovca.

4.2 Nova generacija

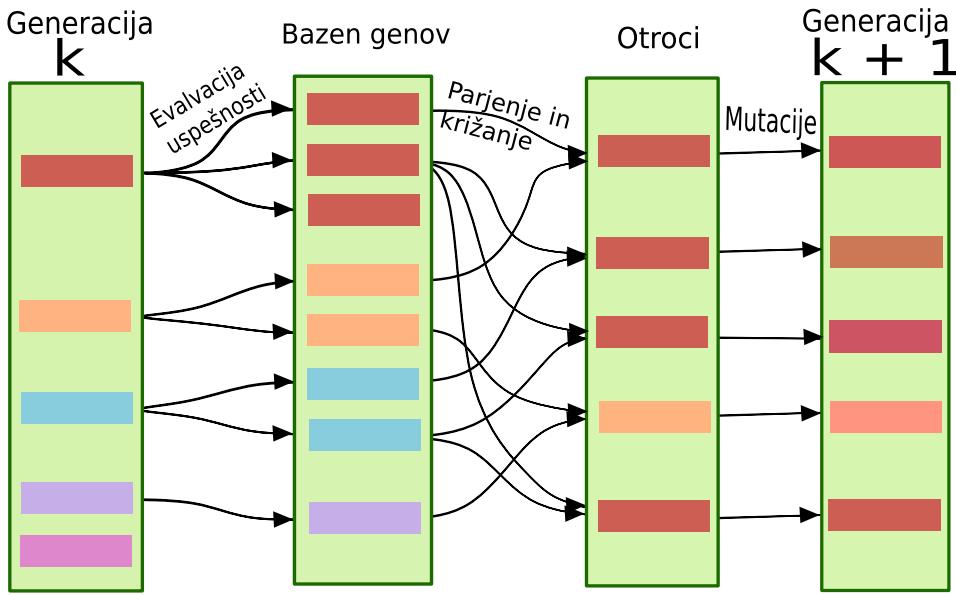
Ko ocenimo uspešnost vseh genov v generaciji, moramo narediti novo generacijo, kjer damo večjo verjetnost za ponovno pojavitev genom, ki jim je šlo bolje. Pri tem pa moramo ohranljati dovoljšnjo variabilnost genoma.

Genetski algoritem sestoji iz več generacij. Novo generacijo naredimo iz prejšnje z uporabo selekcije, kjer imajo bolj uspešni geni večjo verjetnost parjenja, križanja obeh genov iz para in mutacij na dobljenem genu. Oglejmo si podrobnosti teh korakov. Na sliki 12 lahko vidimo grafični prikaz razmnoževanja.

4.2.1 Parjenje

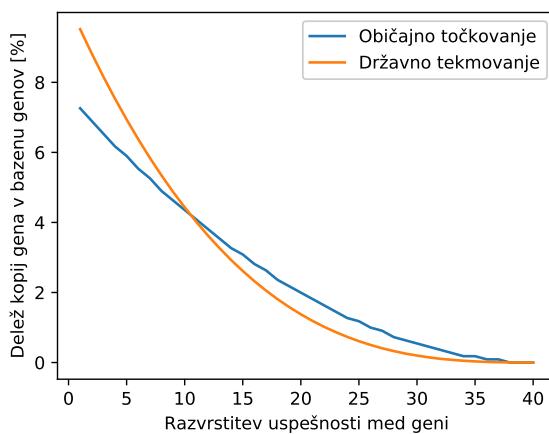
Obstaja mnogo različnih načinov parjenja. Lahko določimo, koliko najboljših genov se bo z enako verjetnostjo razmnoževalo, čemur rečemo elitna metoda, a tu se del genov ne pojavlja več v naslednjih generacijah razen v primeru mutacij. Mi smo si izbrali verjetnostni pristop, kjer ima vsak gen verjetnost parjenja odvisno od uspešnosti.

Za ta namen pripravimo bazen genov, iz katerega kasneje dobimo naključne pare. Pri tem v bazen postavimo $2i^2/n$ kopij gena (zaokroženo na najbližje celo število)



Slika 12: Najprej geni z izmerjeno uspešnostjo (vhod), več ponovitev v bazenu bolj uspešnim, parjenje, križanje in mutacije (izhod).

z i -to najnižjo uspešnostjo, če imamo do konca evolucije še vsaj tri generacije, kjer je n velikost populacije v eni generaciji. V zadnjih treh generacijah smo v bazen postavili $2i^3/n$ kopij i -tega gena, s čimer smo naredili še večji pritisk na naravno selekcijo, saj imajo slabši posamezniki še manjše možnosti za parjenje, s čimer je pristop na nek način bolj podoben elitni metodi. Na sliki 13 si lahko ogledamo verjetnosti za parjenje v seznamu genov razvrščenem po uspešnosti.



Slika 13: Graf z obema krivuljama.

Iz dobljenega bazena vzamemo n parov genov.

4.2.2 Križanje

Vsek par genov naključno križamo in s tem dobimo večjo variabilnost genov v populaciji. Uveljavljenih je več pristopov križanja. Ker želimo ohranjati dolžino gena, se osredotočimo le na rezanje obeh genov v istih točkah in na uniformno izbiro. Križanje z rezanjem pomeni, da v eni ali dveh točkah prerežemo oba gena in nato vrnemo gen, kjer se izmenjujejo kosi, ki prihajajo od obeh staršev. Ker pa sosednost znotraj gena nič ne pomeni, smo se odločili za uniformno križanje, kjer naključno vsako vrednost iz zaporedja v nov gen kopiramo od enega ali drugega izmed staršev. Tako se gena enakomerno premešata.

4.2.3 Mutacija

Da ohranjamo raznolikost genoma in se s tem izogibamo lokalnim optimumom, dobljenemu genu na vsakem mestu z verjetnostjo p izberemo novo naključno vrednost. Pri tem na vsakem koraku naredimo ožji interval, s katerega vrnemo naključno vrednost. S tem sčasoma dovoljujemo le še manjše spremembe, saj preveč mutacij lahko onemogoča evolucijski napredok skozi generacije. Vrnemo naključno vrednost z intervala $[x - \frac{1}{generacija}(M - m), x + \frac{1}{generacija}(M - m)] \cap [m, M]$, kjer je x vrednost po križanju, M zgornja meja izbranega parametra, m spodnja meja izbranega parametra in $generacija$ zaporedna številka generacije. Meje parametrov lahko najdemo v tabeli 3.

4.2.4 Državno tekmovanje

Pri istem genu so ovčarji lahko različno uspešni. Želimo se izogniti izbiru gena, ki je imel le *srečo* in v povprečju ni tako dober, prav tako želimo zmanjšati verjetnost neuspeha. V ta namen smo uvedli drugačno točkovanie v zadnjih treh generacijah. Do te točke so slabi geni večinoma že izumrli, saj je verjetnost, da ima slab gen čez veliko generacij še potomce, ki so mu podobni, vedno manjša. Celo v povprečju mu mora iti dobro, sicer najverjetneje do konca evolucije niti ne pride. Na koncu pa imamo že zelo uspešne gene, med katerimi si želimo vzeti najzanesljivejšega. Med zadnjimi tremi poostrimo vrednotenje, da onemogočimo uspeh na podlagi *sreče*. Temu pristopu bomo rekli *državno tekmovanje*. Prav tako bomo poleg drugačne funkcije uspešnosti uporabljali tudi bolj elitno parjenje, kot smo za zadnje tri generacije že omenili v poglavju 4.2.1.

Novo točkovanje temelji na treh evalvacijah istega gena z enako funkcijo uspešnosti kot prej po formuli (4.1). Za vsak gen dobljene tri uspešnosti f_1, f_2, f_3 združimo v

$$\Phi(f_1, f_2, f_3) = \frac{\sum_{i \in [3]} f_i}{3} \min_{i \in [3]} f_i, \quad (4.2)$$

kjer z $[3]$ označujemo množico naravnih števil do vključno 3. Gen je torej boljši, če je v povprečju uspešnejši in če je njegova najnižja uspešnost dovolj visoka.

4.3 Naša implementacija genetskega algoritma

Z genskim algoritmom smo iskali najboljši gen za določeno število psov ovčarjev, določeno začetno število ovc in model gibanja ovc. Velikosti črede, za katere smo iskali najboljši gen, so 5, 10, 25, 50, 75 in 100. Ovčarjev pa je bilo 1, 2, 3 ali 4. Za ostale velikosti črede ali število psov lahko model pospolimo tako, da vzamemo najbolj podobno število ovc in psov. Tako smo našli dobre gene za 72 začetnih kombinacij.

Vrednosti parametrov genetskega algoritma si lahko ogledamo v tabeli 4. Skico naše implementacije z državnim tekmovanjem lahko vidimo v algoritmu 1.

Oznaka	Opis parametra	Uporabljena vrednost
n	Velikost populacije	40
G	Število generacij	27
p	Verjetnost mutacije	1 %
t_{MAX}	Največji čas simulacije	180 s

Tabela 4: Parametri genetskega algoritma.

Algoritem 1 Genetski algoritem

```

1: Inicializacija
2: Evalvacija
3: for generacija = 1, 2, ...,  $G$  do
4:   if generacija  $\leq G - 3$  then
5:     Selekcija
6:   else
7:     Elitna selekcija
8:   end if
9:   Križanje
10:  Mutacija
11:  if generacija  $\leq G - 3$  then
12:    Evalvacija
13:  else
14:    3x Evalvacija
15:  end if
16: end for

```

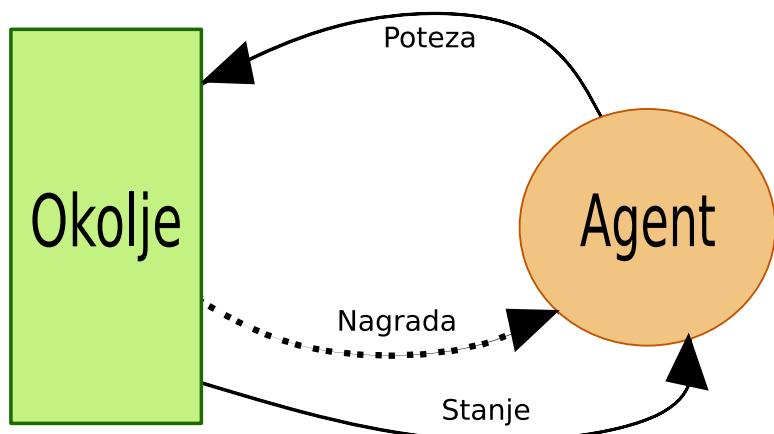
5 Model vodenja z adaptivnim genom

Problem se pri genetskem algoritmu lahko pojavi, ko se število ovc na pašniku močno zmanjša, saj so parametri ves čas enaki, odvisni le od začetnega števila ovc. Parametre želimo prilagajati trenutnemu stanju na pašniku, številu ovc, psov in še čemu, kar bi lahko koristilo uspešnosti vodenja črede. Celo bolje bi bilo, če bi se ovčarji lahko naučili česa novega, izvirnega, česar naš model ne predvideva niti ob sprememjanju parametrov. Želimo mu dati čim bolj proste roke pri izbiri smeri in hitrosti.

Zato bomo uporabili metode umetne inteligeunce, ki nam omogočajo učenje čim boljših potez v določenem trenutku. Z vodenjem črede s pomočjo spodbujevanega učenja so se ukvarjali tudi avtorji člankov [18] in [7]. Ker je naš program pripravljen v programu Unity, smo se odločili za uporabo paketa ML Agents, ki temelji na spodbujevanem učenju, konkretno na PPO algoritmu. V tem poglavju bomo predstavili ta izbrani algoritem.

5.1 Spodbujevano učenje

Spodbujevano učenje je osredotočeno na interakcijo med agentom in okoljem, pri čemer agent maksimizira nagrade, ki jih dobi iz okolja, ko v danem stanju naredi neko potezo [5] (slika 14). Agent je hkrati učenec in odločevalc. Tekom učenja skrbi za ravnotežje med raziskovanjem nepoznanega in izkoriščanjem pridobljenega znanja.



Slika 14: Interakcije med agentom in okoljem.

Model sicer pogosto temelji na markovskem procesu odločanja, za katerega bi lahko narisali graf stanj in povezav med stanji z verjetnostmi prehodov, ampak za algoritme spodbujevanega učenja ne potrebujemo poznavanja tega grafa, saj ga

algoritmom sam raziskuje in se uči najboljših odločitev v raziskanem prostoru brez računanja in pomnjenja samega grafa.

Okolje modela je določeno s stanji $s \in \mathcal{S}$, potezami $a \in \mathcal{A}(s)$, ki so agentu na voljo v stanju s , pripisanimi numeričnimi nagradami $r(s, s', a) \in \mathcal{R}$ ob izbrani potezi a in spremembji stanja iz s v stanje s' ter verjetnostmi stanj $\mathbb{P}(s'|s, a)$ doseženih iz stanja s' z uporabo poteze a . Prav ta verjetnost in vrednosti nagrad so nam običajno neznane.

Agent se med možnimi potezami odloča na podlagi svoje strategije izbire akcij. Med učenjem posodablja strategijo $\pi(a, s) = \mathbb{P}(a|s)$, ki predstavlja verjetnost izbire poteze a v stanju s tako, da je pričakovana vsota nagrad v prihodnosti največja, torej maksimiziramo vrednostno funkcijo izbranega stanja s ob uporabi izbrane strategije

$$V^\pi(s) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s_t = s, \pi], \quad (5.1)$$

kjer sledimo trenutni strategiji π in je $\gamma \in [0, 1]$ diskontni faktor, ki uravnoteži pomen trenutne in prihodnjih nagrad. Funkcija $V(s)$ oceni dolgoročni obet uporabe strategije π iz trenutnega stanja, pri čemer da manjšo težo oddaljenim nagradam. Tako se agent načeloma izogiba potezam, ki mu bodo dolgoročno škodovale in se raje odloča za tiste, ki mu obljudljajo večjo nagrado v krajšem času. Pri učenju pa je pomembno, da agent tudi raziskuje po prostoru stanj. K temu ga spodbudimo z naključnimi potezami z verjetnostjo ϵ . Ta parameter je običajno najprej nastavljen na neko začetno vrednost med 0 in 1, potem pa pada skozi čas proti 0. S tem agent najprej predvsem raziskuje, potem pa vedno bolj izkorišča svoje znanje, ki si ga je pridobil. Vseeno pa se lahko ujame v lokalnem optimumu.

Agent mora oceniti lastnosti okolja, predvsem verjetnost prehodov med stanji $\mathbb{P}(s'|s, a)$, da lahko oceni vrednosti stanj. Na principu spodbujevanega učenja z nagradami in kaznimi se v veliki meri učijo tudi človeški možgani. Razvitih je že mnogo algoritmov za učenje optimalne strategije, kot so TD-Learning, Q-Learning, SARSA, globoko Q-učenje, PPO in drugi. Paket ML Agents uporablja algoritom PPO, zato si tega nekoliko podrobnejše oglejmo.

5.2 Optimizacija z bližnjo strategijo

Motivacija za optimizacijo z bližnjo strategijo (Proximal Policy Optimization - PPO) je vprašanje, kako kar najbolj izboljšati strategijo na podlagi trenutne, ne da bi zradi slučajne velike nagrade pomotoma stopili predaleč in povzročili nenaden padec uspešnosti strategije. PPO za vrednosti parametrov nima omejitev, ampak ima omejitev le na velikost koraka, s katerim spremenimo parametre. Omejitev je postavljena z namenom, da je nova strategija dovolj podobna prejšnji, saj bolj zaupamo vrednostim parametrov blizu prejšnje strategije, ker nočemo uničiti naučenega na podlagi posamezne ocene.

PPO je algoritmom, ki se uči napovedovanja pričakovanih nagrad, da se tako odloča za najbolj obetavne poteze v danem stanju. S pomočjo gradientnega spusta

spreminjamo vektor parametrov θ_k in s tem strategijo π^{θ_k} , ki je odvisna od tega vektorja. Vektor parametrov si lahko predstavljamo kot vektor uteži v nevronski mreži. Pri tem moramo stanje in potezo nekako numerično predstaviti modelu. Poleg nevronskih mrež, ki izračuna verjetnost neke poteze na podlagi stanja, definiramo tudi vrednostno funkcijo $\hat{V}^{\phi_k}(s)$, ki oceni vrednost prihodnjih nagrad z uporabo nevronskih mrež, ki jo naučimo ocenjevati pričakovane nagrade na podlagi stanja. Tudi ta funkcija mora biti odvedljiva in odvisna od vektorja parametrov ϕ_k .

PPO je za razliko od Q-Learning in nekaterih drugih algoritmov spodbujevanega učenja primeren tudi za zvezna prostore možnih potez in stanj. Razvil ga je Schulman iz OpenAI s sodelavci [14]. Algoritem je po rezultatih primerljiv z drugimi trenutno najboljšimi algoritmi spodbujevanega učenja, njegova prednost pa je predvsem v relativno enostavni implementaciji in nastavljanju parametrov algoritma.

Pri tem algoritmu definiramo tudi primerjalno funkcijo

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_t^k(s, a), g(\epsilon, A_t^k(s, a)) \right), \quad (5.2)$$

kjer je θ_k trenutni vektor parametrov strategije (v našem primeru uteži nevronskih mrež), θ izbrani vektor parametrov strategije, ki ga primerjamo s trenutno strategijo, A_t^k je ocenjena donosnost poteze ob času t , ϵ je parameter, ki omejuje spremembo strategije, običajno med 0.1 in 0.2 ter

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0. \end{cases} \quad (5.3)$$

Donosnost poteze je enaka $A_t^k = V(s_t) - \hat{V}^{\phi_k}(s_t)$. Torej predstavlja razliko med točno vrednostjo vrednostne funkcije izračunane na podlagi dejanskih nagrad, ki ni samo ocena, in njeni oceno izračunano po formuli pridobljeni na podlagi preteklega znanja z vektorjem parametrov ϕ_k , ki se jih prav tako sproti učimo za boljše ocenjevanje pričakovanih nagrad.

PPO posodablja strategijo na sledeč način:

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s, a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)], \quad (5.4)$$

običajno z več koraki stohastičnega gradientnega spusta za maksimizacijo primerjalne funkcije. Za lažjo optimizacijo je priporočljivo, da sta vrednostna funkcija in strategija odvedljivi. Če je donosnost poteze pozitivna oziroma je višina nagrad nad pričakovanji, verjetnost izbire poteze a pa je po strategiji π_θ večja kot prej, raje izberemo to strategijo, saj želimo verjetnost dobre poteze povečati. Na ta način tudi zmanjšamo verjetnost slabe poteze. Funkciji \min in g pa v formuli (5.2) služita temu, da damo strategijam z relativno razliko med verjetnostjo izbire poteze a v primerjavi s staro strategijo večjo od ϵ enako primerjalno vrednost. S tem se izognemo navduševanju nad velikimi vrednostmi primerjalne funkcije, ki bi nas potegnile daleč od trenutne strategije. Stari strategiji dolgoročno namreč bolj zaupamo, saj ni odvisna le od posamezne uspešnosti simulacije. Pri višini nagrad lahko pride do večjega

šuma, ki ga povzroča velika variabilnost vrednosti nagrad. S tem se strategija ne more dramatično spremeniti, lahko pa se poveča vrednost vrednostne funkcije. V algoritmu 2 si lahko postopek podrobneje ogledamo.

Algoritem 2 PPO

- 1: Vhod: začetni parametri strategije θ_0 , začetni parametri vrednostne funkcije ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Zberi množico izvedenih simulacij \mathcal{D}_k z uporabo strategije π_{θ_k} .
- 4: Izračunaj dobljene nagrade $(r_t)_{t=0}^T$ za vsako simulacijo.
- 5: Izračunaj ocene donosnosti $(A_t^k)_{t=0}^T$ temelječ na vrednostni funkciji \hat{V}_{ϕ_k} .
- 6: Posodobi strategijo s pomočjo primerjalne funkcije:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T L(s_t, a_t, \theta_k, \theta),$$

običajno preko stohastičnega gradientnega spusta.

- 7: Popravi vrednostno funkcijo z regresijo, da bo čim manjša napaka vsote kvadratov:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (V(s_t) - \hat{V}^{\phi}(s_t))^2,$$

običajno preko algoritma gradientnega spusta.

- 8: **end for**
-

5.3 Adaptivni gen

Pri učenju modela smo najprej poizkusili na način, da bi ovčarju dali proste roke, da se odloča o smeri in hitrosti na podlagi opazovanj. Poizkušali smo z različnimi funkcijami za nagrajevanje, veliko različnimi opazovanimi vrednostmi in celo z demonstracijami, kar pomeni, da smo algoritom najprej učili ne da bi ta poskušal z naključnimi potezami, ampak s potezami, kakršne je izbiral ovčar z optimalnim genom iz genetskega modela. Če smo dali ovčarju popolnoma proste roke, je bilo učenje neuspešno celo v primeru z eno samo ovco. Ta prostor stanj je bil prevelik, da bi ga agent uspešno spoznal in se iz njega česa naučil. Običajno se v različnih virih ML Agents uporablja za bolj enostavne probleme. Poizkusili smo še z manj svobodnim modelom. Ta model smo poimenovali *adaptivni gen*.

Ideja je, da damo ovčarju le nekaj opazovanih vrednosti, on pa se odloči za najboljši gen znotraj dovoljenih meja za vsako vrednost v genu. Gen je definiran enako kot v poglavju 4. Adaptivni gen bi moral imeti celo boljše rezultate kot optimalni gen iz genetskega algoritma, saj ima ta model še več svobode in vključuje tudi možnost, da se nauči enakih genov kot druga dva modela, če jih vmes ne spreminja. Vemo, da model naučen z genetskim algoritmom deluje, dovolj bi ga bilo le izboljšati. Ovčar naj se nauči izbere gena iz le nekaj osnovnih opazovanih lastnosti. Ker pa je učenje precej zamudno, smo se odločili le za Ginellijev model ovc in le

za enega ovčarja na naključno veliki čredi med 1 in 10 ovcami, saj je ta model bolj podoben naravnemu obnašanju ovc, ovčar je imel probleme le sam, s sodelovanjem pa je model precej uspešen že z genom dobljenim z genetskim algoritmom. Večje črede se mu žal ni uspelo naučiti voditi niti v več urah.

5.4 Naša implementacija modela

Model se odloča vsakih 20 korakov, kar pomeni na 0,4 s. Med posameznimi odločtvami uporablja gen, ki ga je nazadnje izbral. Trajanje simulacije T smo navzgor omejili na 750 izračunov gena (300 sekund oziroma 5 minut), kar je več kot tri minute z namenom, da se lahko uči vodenja tudi, če je za zbiranje potreboval več časa. Za primerni vrednosti parametrov modela sta se izkazali vrednosti $\epsilon = 0,2$ in $\gamma = 0,99$. Pred učenjem smo naredili 128 simulacij za pospešitev učenja na podlagi demonstracij. Za izbiro poteze se je model naučil uteži za nevronsko mrežo s 5 plastmi po 512 nevroni.

5.4.1 Nagrajevanje

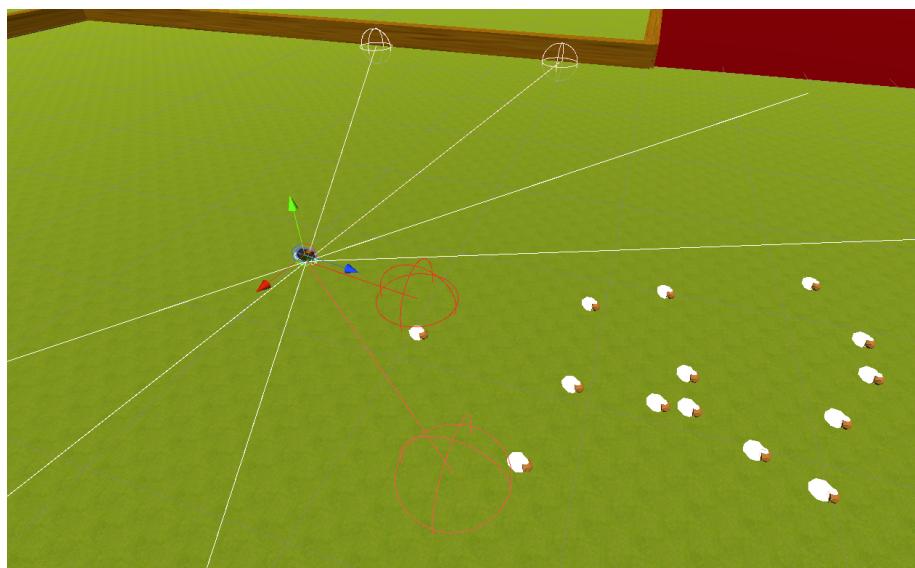
Ovčar dobi nagrado veliko $\frac{1}{20T}$ vsakič, ko se center črede (GCM) premakne bliže staji, kot je bil center blizu staji kadar koli pred tem znotraj simulacije. V vsakem časovnem koraku prav tako dobi kazen, ki ga spodbuja k hitrejšemu reševanju problema. Ovčar dobi kazen v primeru, da črede ne pripelje bliže staji, sicer ne dobi ničesar.

Ko spravi zadnjo ovco v stajo dobi še $4(\frac{T-t}{T})^2$ nagrade za uspešno opravljeno nalogu, kjer je t čas ob zaključku simulacije. Z višino nagrade ga spodbujamo k hitrejšemu vodenju.

5.4.2 Opazovanje vrednosti

Stanje, v katerem je ovčar, definiramo s številom ovc na pašniku, deležem pobeglih ovc glede na trenutni delež dovoljenih pobeglih ovc, razliko med GCM in stajo ter ovčarjem in stajo v smeri x in z . Uporabimo izračunane vrednosti v tem in prejšnjem koraku posodabljanja gena.

Poleg tega ovčar opazuje tudi oddaljenost objektov v smereh devetih žarkov razvrščenih okrog njega s skupnim kotom 120° . Pri tem lahko loči med ovco in ograjo oddaljeno največ 50 m. Na sliki 15 si lahko ogledamo razvrstitev žarkov.

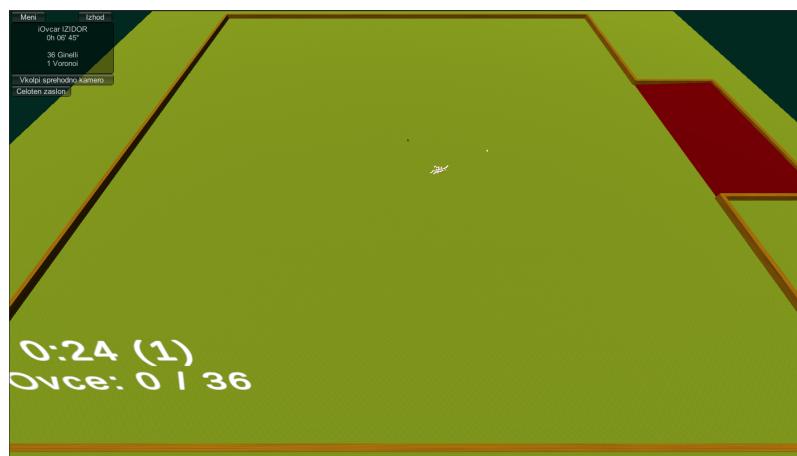


Slika 15: Predstavitev prostora z žarki. Ovčar lahko prepozna objekt in določi razdaljo do njega.

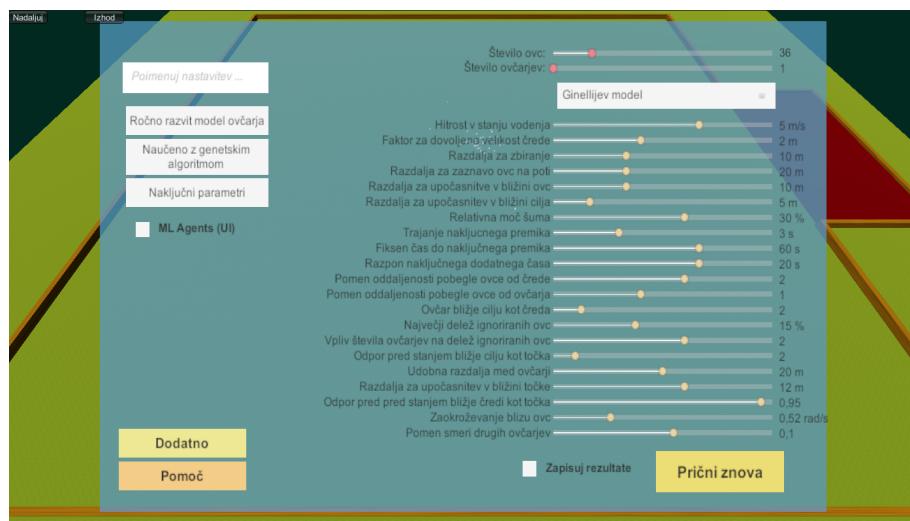
6 Program iOvčar IZIDOR

Za učenje in testiranje modelov smo pripravili program, ki smo ga poimenovali *iOvčar IZIDOR*. Program je napisan v programskem jeziku *C#*, za vizualizacijo smo uporabljali program Unity in za spodbujevano učenje paket ML Agents⁹. Program nam je omogočal grafični prikaz simulacije in beleženje rezultatov. Na slikah 16 in 17 si lahko ogledamo izgled simulacijskega okolja.

V programu so modeli vodenja psa ovčarja običajno poimenovani na drugačen način. Ročno razviti model je imenovan "Voronoi", model razvit z genetskim algoritmom je "AI1" in model z adaptivnim genom je "AI2".



Slika 16: Običajen pogled med simulacijo. Program nam omogoča tudi pogled od zgoraj in izza izbranega ovčarja.

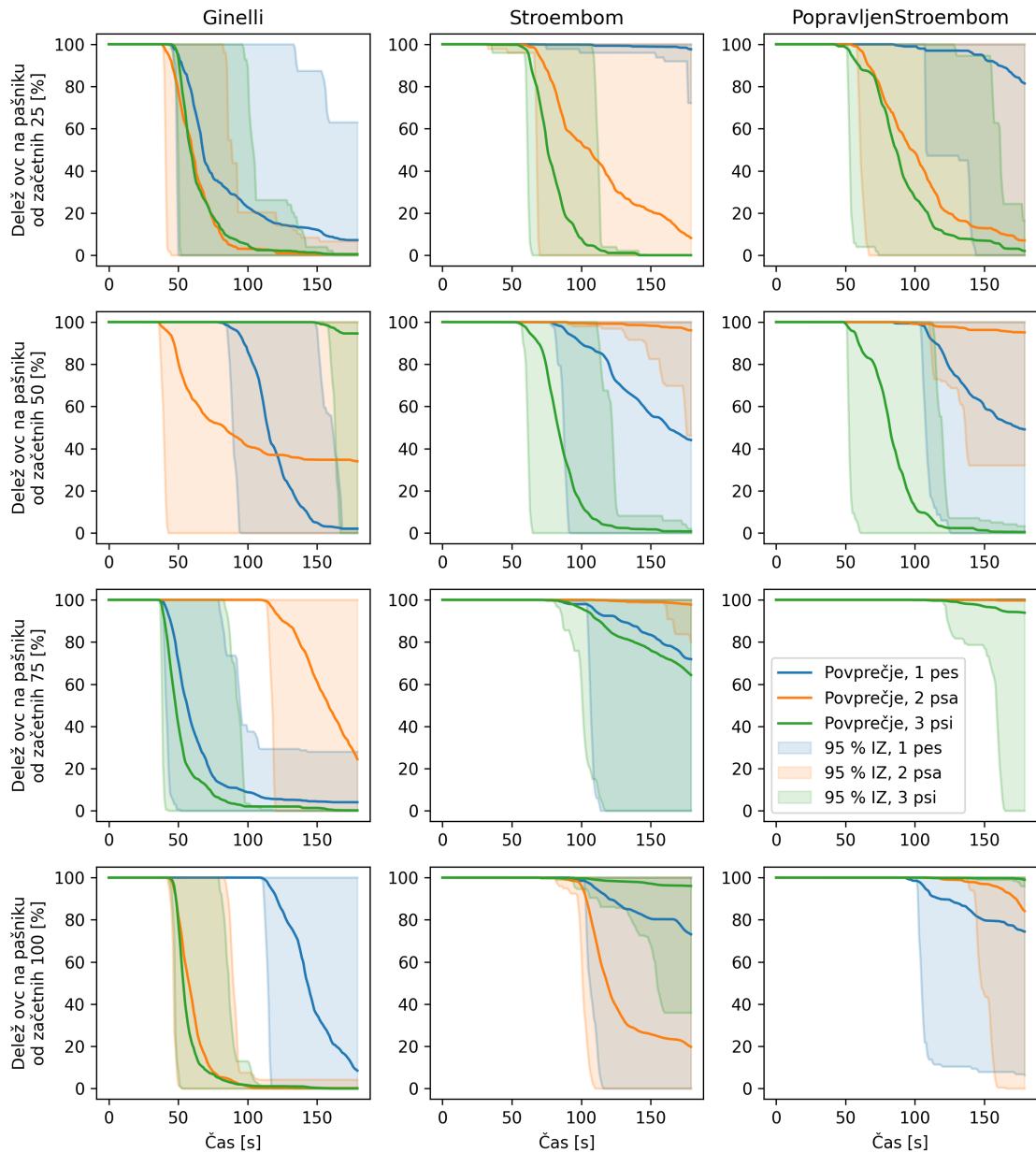


Slika 17: Meni za izbiro nastavitev v programu.

⁹Pri učenju nam je bila v veliko pomoč spletna stran <https://www.immersivelimit.com> [ogled 9. 3. 2021].

7 Rezultati

V tem poglavju si bomo ogledali nekaj rezultatov uspešnosti različnih modelov. Za vsako kombinacijo števila ovčarjev, ovc, modela gibanja ovc in modela vodenja ovčarja smo naredili 100 simulacij.



Slika 18: Delež ovc na pašniku skozi čas za ročno razvit model vodenja ovc. Legenda je za vse grafe enaka in se nahaja le na desnem grafu v tretji vrstici. Vsak stolpec grafov prikazuje izbran model gibanja ovc, vrstica pa velikost črede (25, 50, 75 ali 100). Poleg krivulje za povprečen delež ovc na pašniku ob danem času je na grafu viden tudi interval zaupanja s stopnjo zaupanja 95 %.

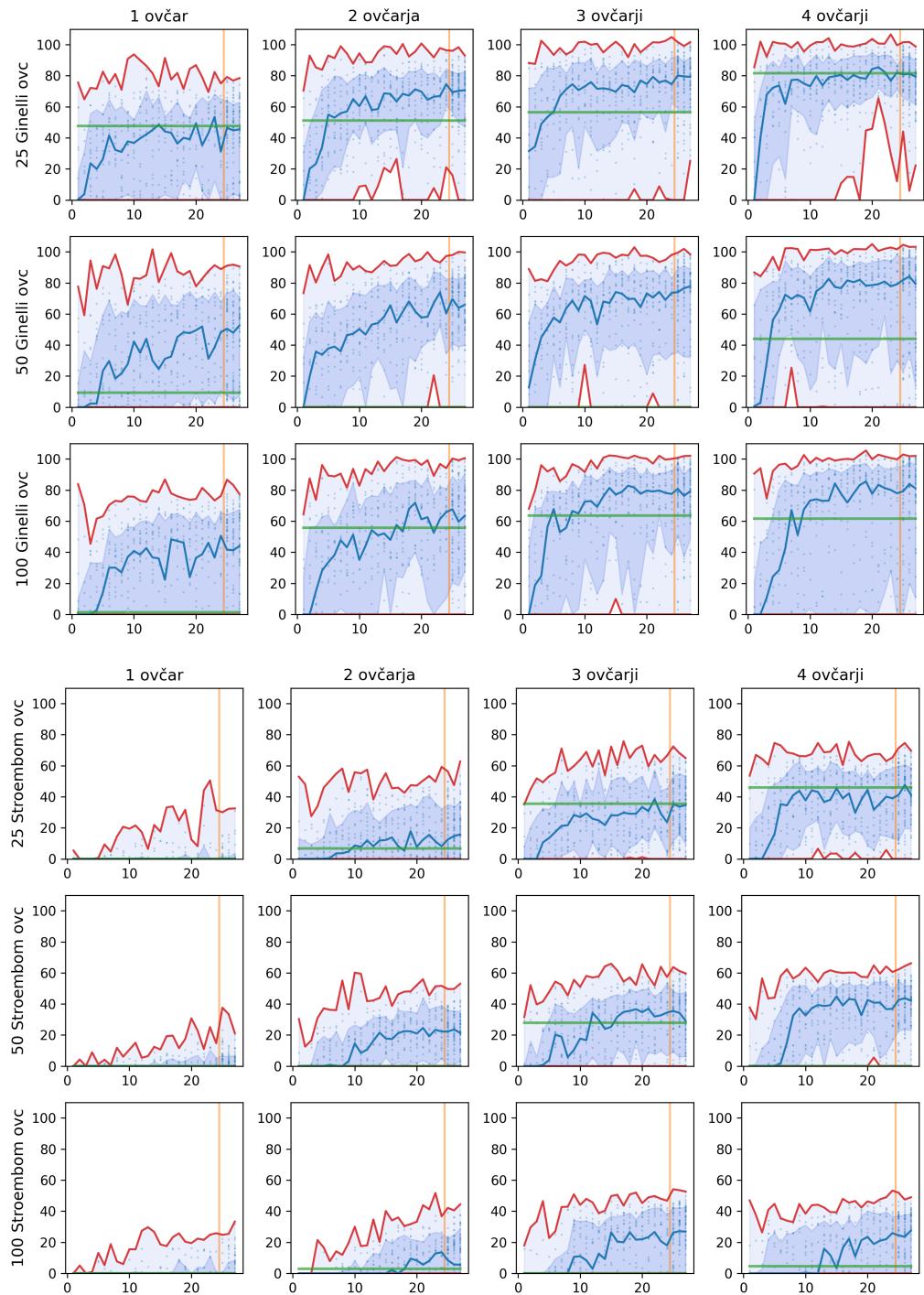
Na sliki 18 si lahko ogledamo povprečen delež ovc na pašniku ob določenem času do konca simulacije pri 180 sekundah pri vodenju psa z ročno razvitim modelom. Hitro opazimo, da so si rezultati pri Strömbomovem in popravljenem Strömbomovem modelu gibanja ovc zelo podobni. Ovčar je najuspešnejši pri Ginellijevem modelu. Več ovčarjev običajno pomeni večjo uspešnost. Pri tem je vpliv začetne razdalje najbližjega ovčarja do črede predvsem pri večjih čredah praktično zanemarljiv.

7.1 Iskanje optimalnega gena

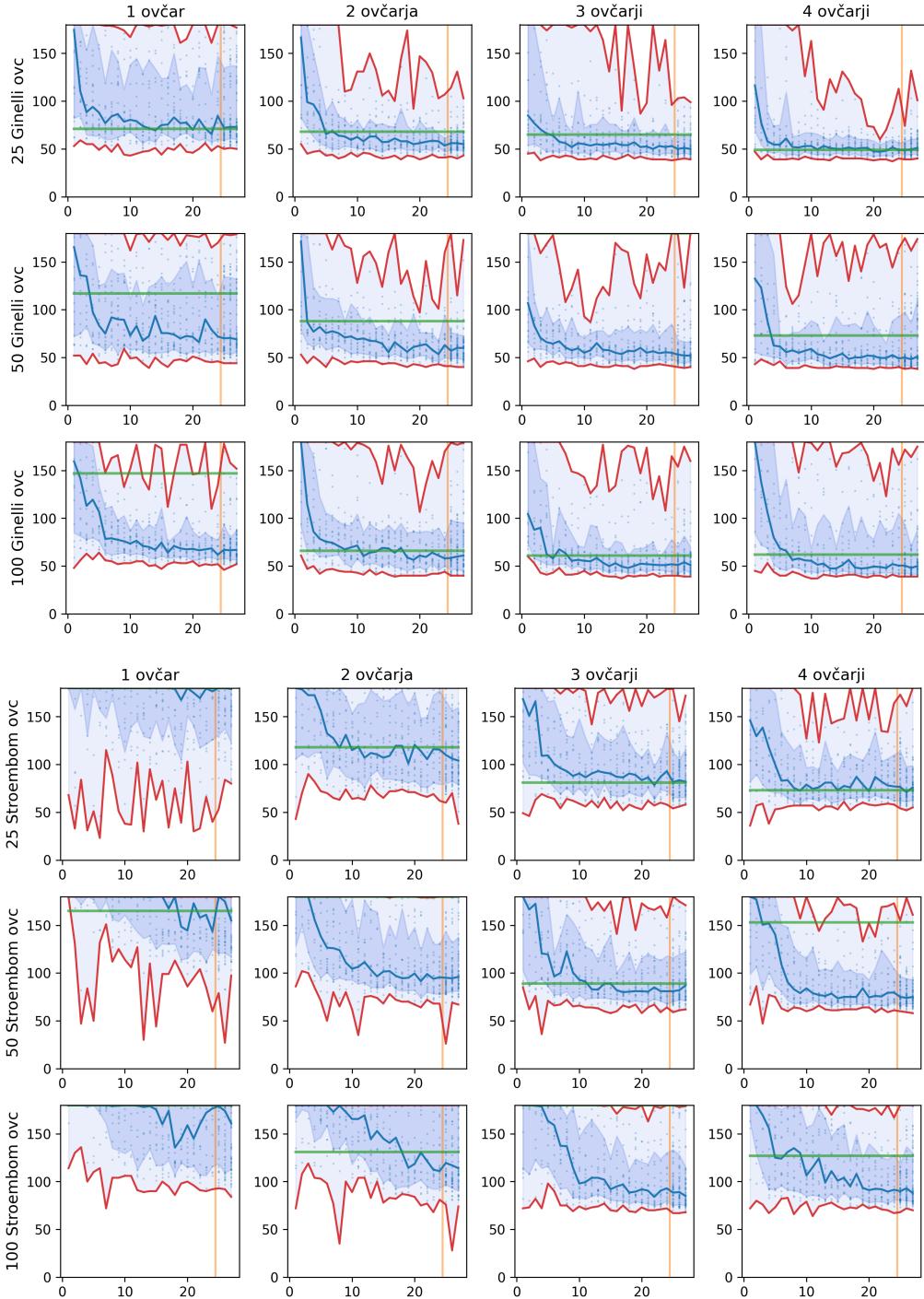
Na slikah 19–22 si bomo ogledali, kako so se nekatere opazovane lastnosti spremnjale skozi generacije. Na vseh grafih temno modra krivulja predstavlja mediano vrednosti izbrane lastnosti, rdeči črti predstavlja ekstremne vrednosti, vmes pa je poleg mediane tudi temnejši moder pas, ki se razprostira od 15. do 85. percentila. Zelena črta predstavlja mediano vrednosti za ročno razviti model, oranžna črta pa predstavlja začetek državnega tekmovanja. V tem delu se bomo pri izrisovanju osredotočili le na Ginellijev in Strömbomov model ovc.

Na sliki 19 vidimo gibanje ocenjene uspešnosti gena izračunane po formuli (4.1), na sliki 20 trajanje posamezne simulacije skozi generacije, na sliki 21 delež ovc v staji ob koncu simulacije, na sliki 22 pa si lahko ogledamo gibanje genotipa za parameter r_a , ki predstavlja faktor za dovoljeno velikost črede.

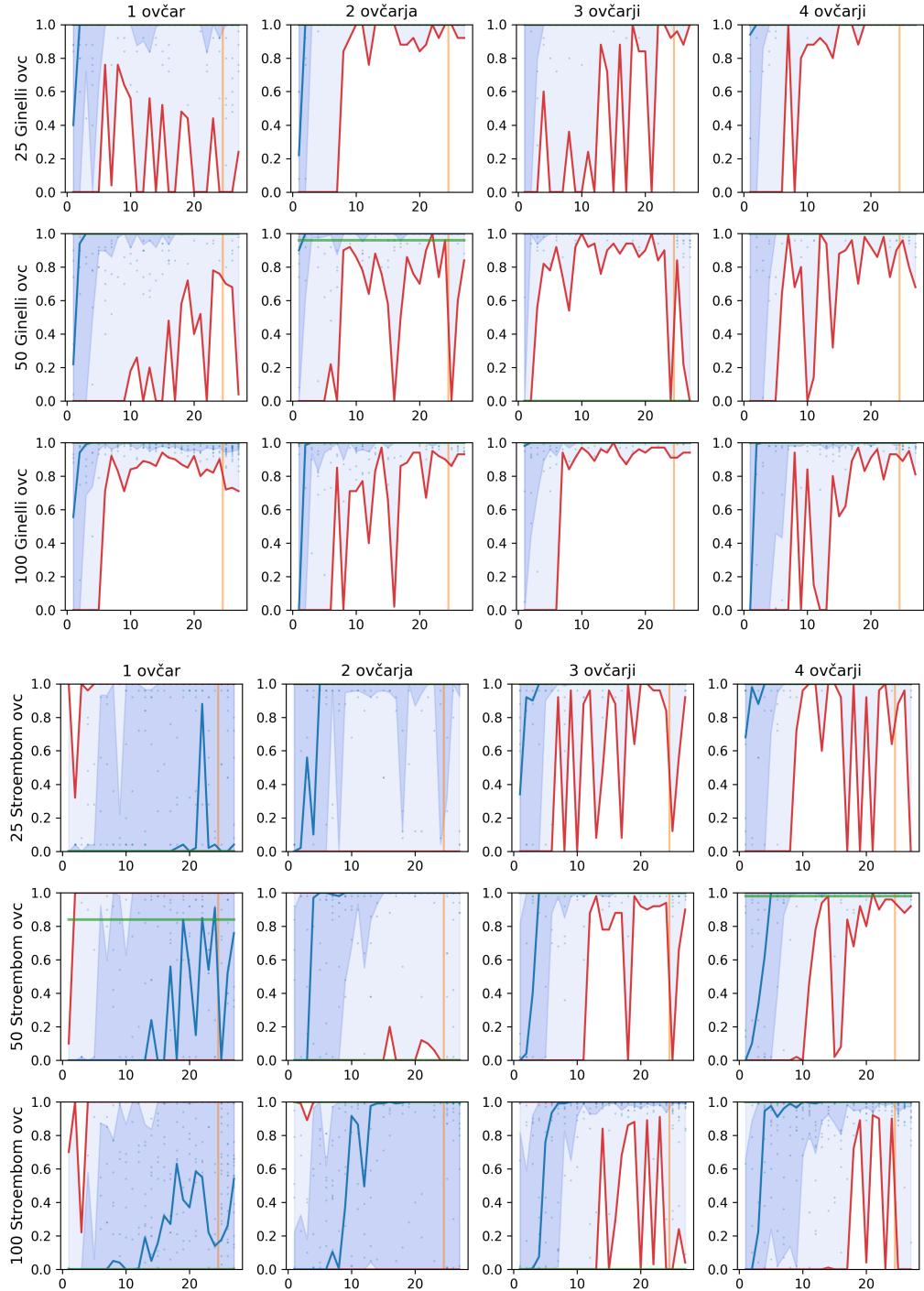
Na nobeni izmed slik ni opaziti pozitivnega pomena državnega tekmovanja, saj so na državnem tekmovanju večinoma le še geni, ki imajo dovolj visoko povprečno in minimalno uspešnost. Poleg tega je v tej točki variabilnost genoma že zelo majhna.



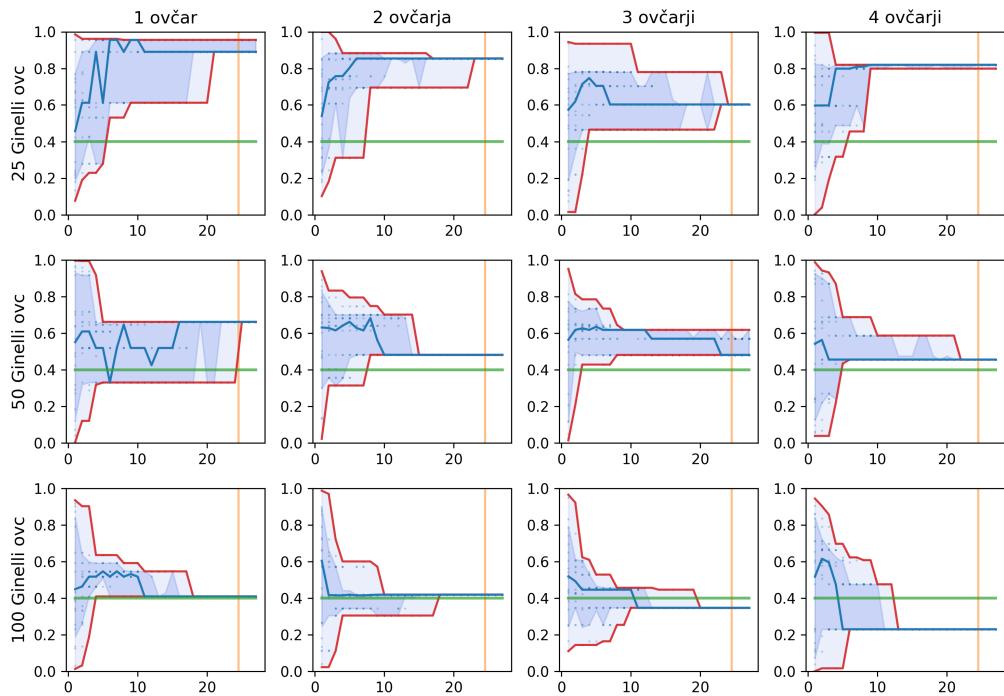
Slika 19: Uspešnost skozi generacije. Opazimo, da se praviloma hitro poveča. Pri tem je na koncu uspešnost običajno vsaj primerljiva z ročno razvitim modelom. Zelene črte pogosto ne vidimo, ker je več kot polovica simulacij v nekaterih primerih neuspešnih. Vodenje ovc po Ginellijevem modelu se zdi veliko lažje za naš model vodenja, saj se model hitro nauči dobrega gena za poljubno velikost črede ali število ovčarjev.



Slika 20: Trajanje simulacije skozi generacije. Rezultati so pričakovani glede na ocenjeno uspešnost na sliki 19. Na večini grafov opazimo, da je mediana na koncu blizu minimalnega časa. Trajanje simulacije je navzdol omejeno s hitrostjo črede. Hitrost teka je pri ovcah po obeh modelih enaka, ampak je čreda ovc po Strömbomovem modelu počasnejša zaradi tresenja, ki smo se ga žeeli znebiti z uvedbo popravljenega modela.



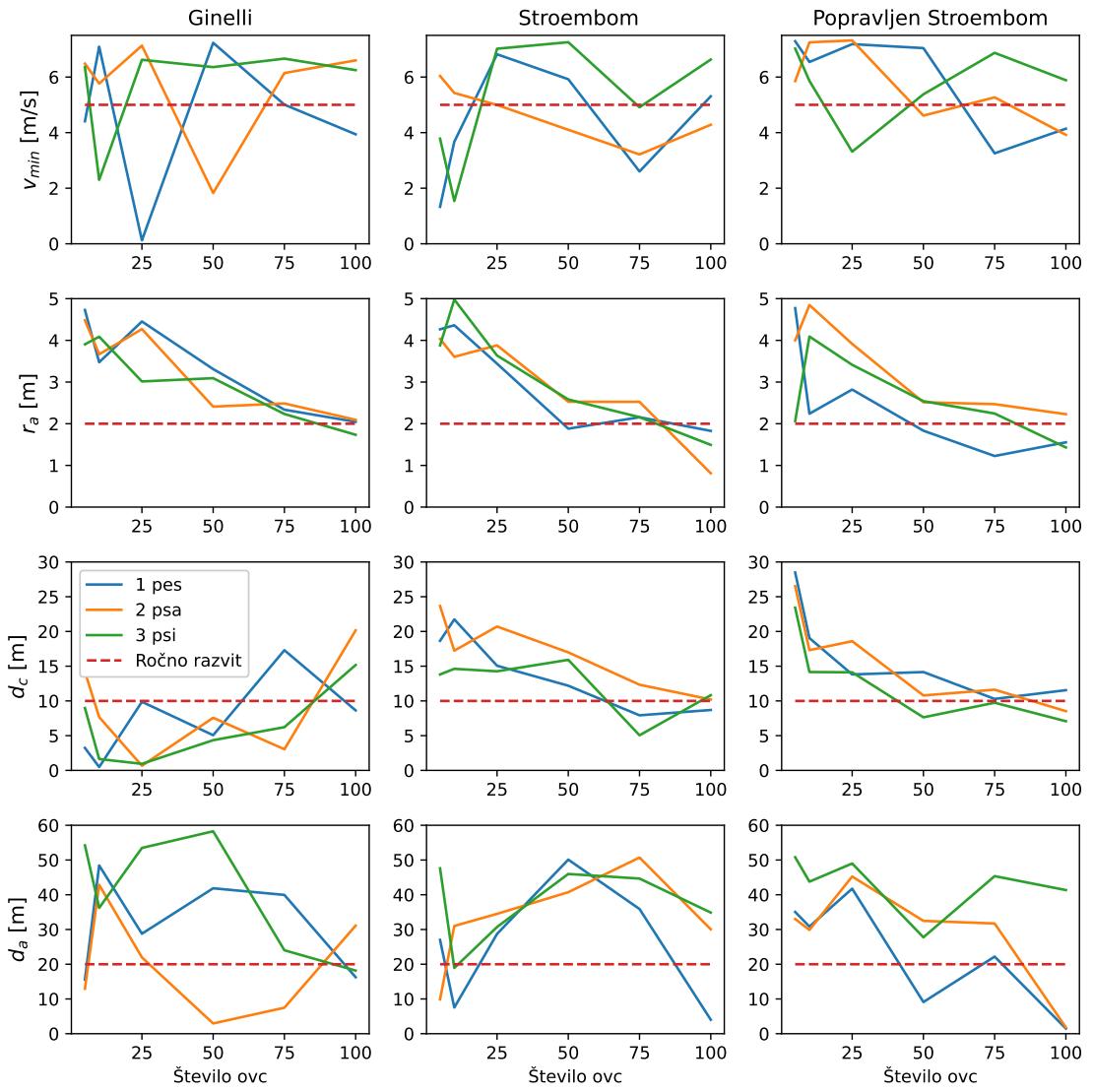
Slika 21: Delež ovc v staji ob koncu skozi generacije. Opazimo, da v večini primerov mediana pride vsaj blizu 100 %, za čredo ovc po Ginellijevem modelu pa se tej vrednosti približa tudi minimum, kar pomeni, da ovčarjem le redko ne uspe privesti vseh ovc v stajo, ampak tudi takrat jih na pašniku ostane le še nekaj. Višje število ovčarjev pozitivno vpliva na delež ovc v staji ob koncu simulacije.



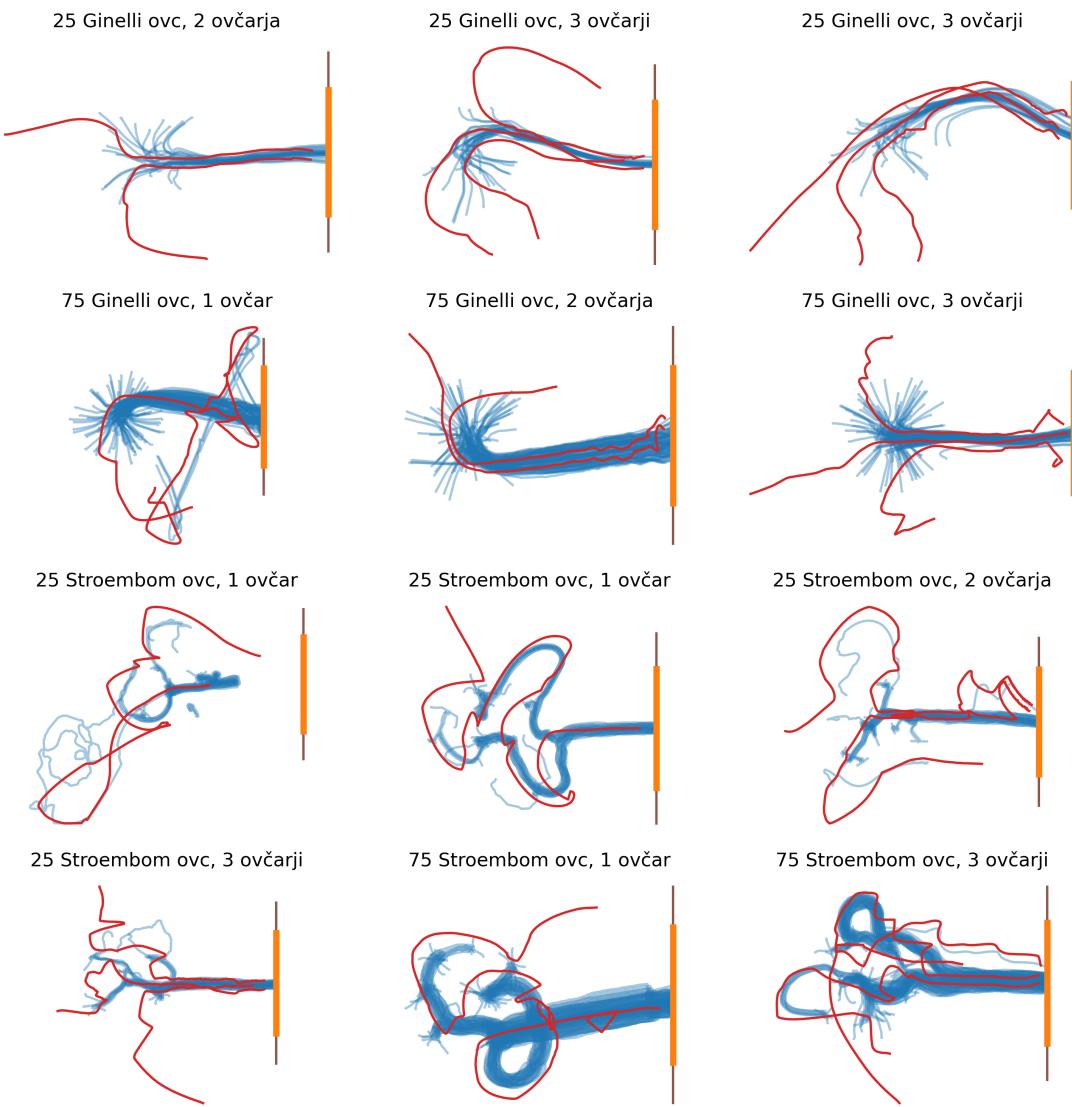
Slika 22: Vrednost genotipa za parameter r_a skozi generacije. Sčasoma večina genotipov izumre in ostane le še majhna variabilnost, ki bi se postopoma še zmanjšala. Neželenim izumrtjem bi se do neke mere lahko izognili z mutacijami na celotnem intervalu, kot je to običajno. Naša ideja ožjega intervala je morda naredila nekaj škode pri uspešnosti modela.

7.2 Model razvit z genetskim algoritmom

Na sliki 23 si lahko ogledamo vrednosti prvih štirih parametrov modela. Vrednosti so naučene z genetskim algoritmom v odvisnosti od števila ovc in ovčarjev. Pri ostalih parametrih ni videti tako lepe povezave med velikostjo črede in vrednostjo parametra. Na sliki 24 si oglejmo poti agentov med simulacijo.



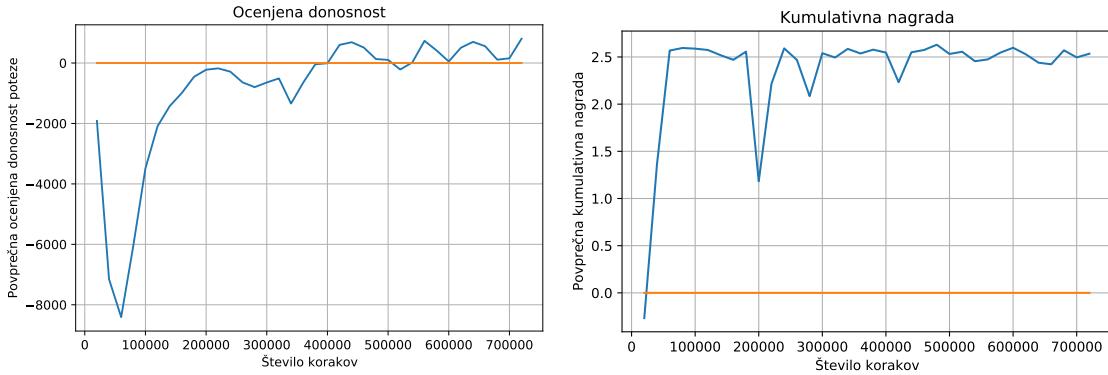
Slika 23: Naučene vrednosti prvih parametrov. Izbrani parametri predstavljajo hitrost v stanju vodenja, faktor za dovoljeno velikost črede, razdaljo za zbiranje in razdaljo za zaznavo ovc na poti. Vrednosti so si ne glede na število ovčarjev precej podobne. Za večjo čredo lahko opazimo, da je vrednost parametra podobna ročno razvitemu modelu. Pri tem modelu smo vzeli enake vrednosti kot avtorji modela. To velja za vse tri modele gibanja ovc. Legenda se nahaja le pri enem grafu, a velja za vse.



Slika 24: Poti agentov med simulacijo. Rdeča krivulja predstavlja pot ovčarja, modra je pot ovce, oranžen pa je vhod v stajo. Opazimo, da se ovce same zberejo že ob približanju ovčarjev, ti pa jih morajo pogosto le še privesti v stajo. Poleg tega se pri Ginellijevem modelu le redko zgodi, da bi katera od ovc pobegnila stran od črede.

7.3 Model z adaptivnim genom

Nato smo se lotili učenja modela z adaptivnim genom. Zaradi težavnosti učenja se nam je uspelo učiti le na čredi veliki do deset ovc po Ginellijevem modelu. Poleg tega smo se v tem primeru osredotočili na enega samega ovčarja, saj ima ta v preostalih dveh modelih največ težav. Na sliki 25 si lahko ogledamo potek učenja obeh nevronskih mrež.



Slika 25: Ocenjevanje dobljene nagrade in dejanska dobljena nagrada. Levi graf nam prikazuje povprečno ocenjeno donosnost poteze, pri čemer model najprej nagrade močno podcenjuje, kasneje pa se jih nauči dobro predvideti. Na desnem grafu lahko vidimo hiter vzpon povprečne kumulativne nagrade skozi simulacije in nato le še nekaj padcev uspešnosti zaradi raziskovanja. Ker postaja s časom prostor stanj vedno bolje raziskan, so ti padci kasneje redkejši in manjši.

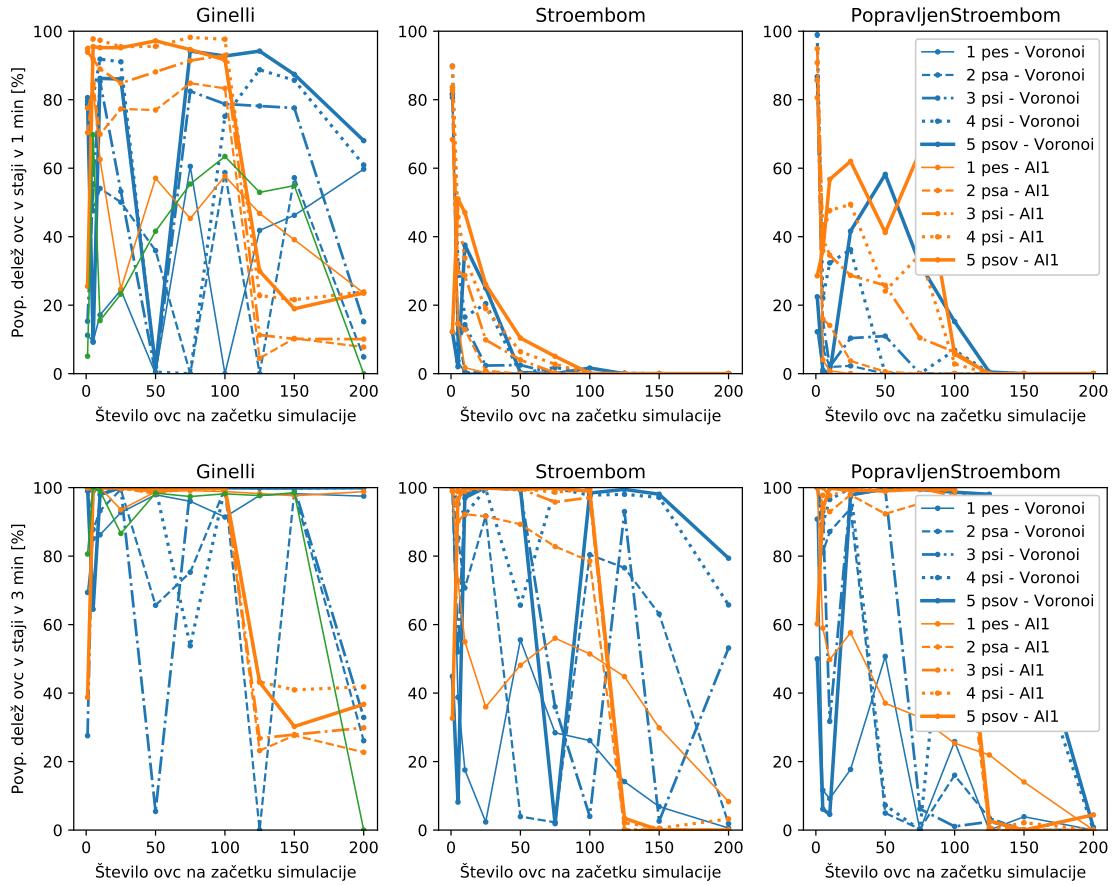
7.4 Primerjava modelov vodenja

Na zadnjih slikah si oglejmo še primerjavo vseh modelov ovc in ovčarjev. Na slikah 26 in 27 hitro opazimo višjo povprečno uspešnost modela razvitega z genetskim algoritmom v vseh izbranih lastnostih, ampak to le za območje, na katerem smo iskali najboljši gen. Za eno ovco ali več kot 100 ovc pa so rezultati veliko slabši kot pri ročno razvitem modelu, kjer smo vedno vzeli gen za najbolj podobno kombinacijo števila ovc in ovčarjev. To kaže na veliko preprileganje, predvsem pri čredah ovc po navadnem in popravljenem Strömbomovem modelu. To preprileganje je opazno predvsem pri večjem številu ovčarjev.

Na sliki 28 se, za lažjo primerjavo modelov, osredotočimo le na enega ovčarja in čredo ovc po Ginellijevem modelu. Prva dva modela vodenja ovčarja sta namreč za več ovčarjev že dovolj dobra, zato se zdi najbolj smiselno najti model, ki bo uspešen tudi v primeru enega samega ovčarja. Vidimo, da je tudi model z adaptivnim genom popolnoma primerljiv najboljšemu izmed ostalih dveh, razen pri čredi z 200 ovcami. To kaže na velik potencial tega modela, saj je dobro posplošil znanje pridobljeno na zelo majhni čredi.

7.5 Razprava

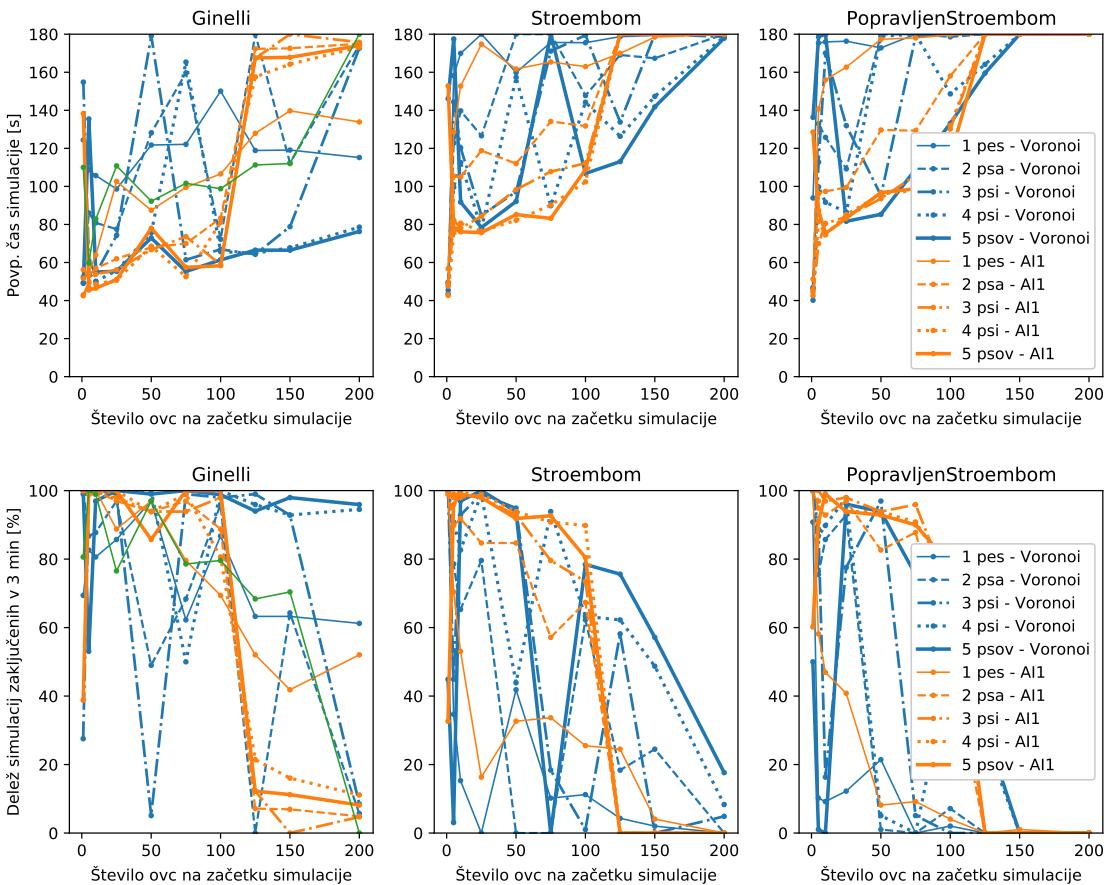
Kot pričakovano je vsak naslednji model v nečem boljši, je pa problem modela razvitega z genetskim algoritmom, da se ne prilagaja spremištanju velikosti črede ter da ne opazuje oddaljenosti od ovc. Zelo je pomembno, kako daleč je ovčar od črede, da jo še lahko vodi. V tem smislu se je za odličnega izkazal model z adaptivnim genom, ki je vse to upošteval.



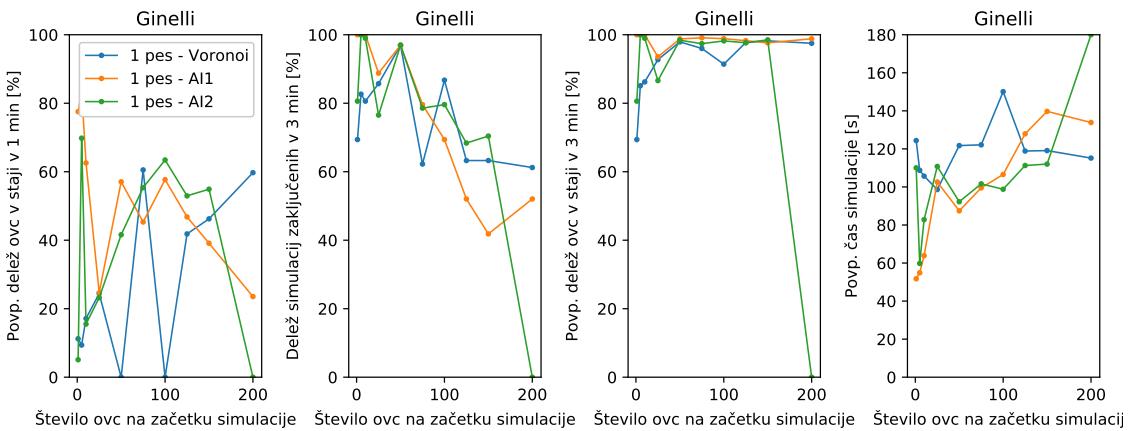
Slika 26: Delež ovc v staji za različne črede in število ovčarjev. V zgornji vrsti grafov vidimo povprečen delež ovc v staji po eni minuti simulacije, v spodnji vrsti pa po treh minutah. Ker je čreda počasnejša pri ovcah po Strömbomovem modelu, je po eni minuti delež opazno nižji, še posebno za velike črede. Popravljeni model je očitno nekoliko hitrejši, a še vedno zelo podoben.

Težava zadnjega modela je zamudno in nestabilno učenje, zaradi katerega je težko pridobiti dober model naučen na večjih čredah. Za to bi potrebovali veliko časa in pametno nastavljene parametre PPO algoritma.

Ideja sodelovanja ovčarjev se je izkazala za dobro vsaj pri ovcah po Ginellijevem modelu. Za ostala dva modela pa bi bilo bolje, če bi si ovčar izbral del črede, ki se drži skupaj in ga pripeljal bliže drugemu delu, dokler ne bi zbral vseh ovc v eno čredo. Zdaj se namreč pogosto zgodi, da pride v ravnovesno stanje in stoji med čredama ali se brez koristi sprehaja od ene do druge. To bi lahko rešili z računanjem gruč in določanjem, katero gručo vodi kateri ovčar, dokler ni celotna čreda v eni sami gruči.



Slika 27: Povprečen čas in delež zaključenih simulacij.



Slika 28: Primerjava modelov vodenja enega ovčarja za čredo po Ginellijevem modelu. Opazimo, da je model z adaptivnim genom primerljiv ali celo boljši kot najboljši izmed ostalih dveh modelov. Njegova uspešnost presega ostala dva še posebej pri čredi 100–150 ovc. Za manjše črede je običajno najuspešnejši model razvit z genetskim algoritmom.

8 Zaključek

Idejo vodenja in sodelovanja bi lahko še nekoliko razvili, a se zdi primerna za preizkušanje v resničnem svetu. V primeru, da strošek robotov ne bi bil prevelik, se zdi uporaba treh ovčarjev najboljša, saj je dovolj zanesljiva, četrti in peti ovčar pa pri naših nastavitevah simulacij nista posebno izboljšala rezultata.

Naš model sodelovanja dovoljuje uporabo enega vsevednega robota, ki ostalim robotom pošilja informacije o postavitvi ovc. S tem lahko imamo namesto skupine dragih robotov le enega dragega in nekaj cenejših, pri čemer dražji robot cenejše upravlja. Ta koncept, ki je v ozadju interakcij med psom in ovcami, lahko močno zniža cene pri več različnih problemih v robotiki.

Možnosti nadaljnega dela so široke. Poleg izboljšave modela z adaptivnim genom bi število ovc poslali proti neskončnosti in čredo obravnavali kot sistem delcev. Tako bi se ukvarjali s porazdelitvami namesto z dejanskimi lokacijami agentov, kar lahko razumemo kot razlito tekočino na vodi. Poleg tega bi na pašnik lahko dodali ovire in vodo, kar bi v simulaciji vneslo dodatne omejitve in dinamike. Poleg tega bi lahko naredili naključne oblike pašnikov, dodali volkove (grožnjo ovcam), pred katerimi bi morali ovčarji varovati čredo. S posplošitvijo v 3D prostor pa bi lahko simulirali tudi vodenje jate ptic in ideje preizkusili še v drugih domenah.

Literatura

- [1] C. Aiba in K. Fujioka, *A suggestion for effective shepherding models with two sheepdogs*, v: IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2020, str. 77–81.
- [2] B. Bennett in M. Trafankowski, *A comparative investigation of herding algorithms*, v: Proc. Symp. on Understanding and Modelling Collective Phenomena (UMoCoP), 2012, str. 33–38.
- [3] P. Degond, A. Manhart in H. Yu, *A continuum model for nematic alignment of self-propelled particles*, arXiv preprint arXiv:1509.03124 (2015).
- [4] J. Demšar, W. Blewitt in I. Lebar Bajec, *A hybrid model for simulating grazing herds in real time*, Computer Animation and Virtual Worlds **31**(1) (2020) e1914.
- [5] V. François-Lavet in dr., *An introduction to deep reinforcement learning*, arXiv preprint arXiv:1811.12560 (2018).
- [6] F. Ginelli in dr., *Intermittent collective dynamics emerge from conflicting imperatives in sheep herds*, Proceedings of the National Academy of Sciences **112**(41) (2015) 12729–12734.
- [7] C. K. Go in dr., *A reinforcement learning approach to the shepherding task using sarsa*, v: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, str. 3833–3836.
- [8] W. Lee in D. Kim, *Autonomous shepherding behaviors of multiple target steering robots*, Sensors **17**(12) (2017) 2729.
- [9] J.-M. Lien in dr., *Shepherding behaviors with multiple shepherds*, v: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, IEEE, 2005, str. 3402–3407.
- [10] A. A. Paranjape in dr., *Robotic herding of a flock of birds using an unmanned aerial vehicle*, IEEE Transactions on Robotics **34**(4) (2018) 901–915, doi: 10.1109/TRO.2018.2853610.
- [11] A. Pierson in M. Schwager, *Bio-inspired non-cooperative multi-robot herding.*, v: ICRA, Citeseer, 2015, str. 1843–1849.
- [12] A. Pierson in M. Schwager, *Controlling noncooperative herds with robotic herders*, IEEE Transactions on Robotics **34**(2) (2017) 517–525.
- [13] C. W. Reynolds, *Flocks, herds and schools: A distributed behavioral model*, v: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 1987, str. 25–34.
- [14] J. Schulman in dr., *Proximal policy optimization algorithms*, arXiv preprint arXiv:1707.06347 (2017).

- [15] R. Šeme, *Računalniška simulacija črede ovc in psa ovčarja*, 2016, diplomsko delo, Univerza v Ljubljani.
- [16] D. Shiffman, S. Fry in Z. Marsh, *The nature of code*, 2012.
- [17] D. Strömbom in dr., *Solving the shepherding problem: heuristics for herding autonomous, interacting agents*, Journal of the royal society interface **11**(100) (2014) 20140719.
- [18] J. Zhi in J.-M. Lien, *Learning to herd agents amongst obstacles: Training robust shepherding behaviors using deep reinforcement learning*, arXiv preprint arXiv:2005.09476 (2020).