RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts
Sistēmu teorijas un projektēšanas katedra

Sergejs Luhmirins
Akadēmiskās bakalaura studiju programmas „Intelektuālās robotizētās sistēmas" students
(stud. apl. nr. 111RDB275)

EVOLUCIONĀRĀS SKAITĻOŠANAS PIELIETOŠANAS ANALĪZE MOBILA ROBOTA VADĪBAI

Bakalaura darbs

Zinātniskais vadītājs
Mg.sc.ing., pētnieks
ALEKSIS LIEKNA

Rīga 2014

DARBA IZPILDES UN NOVĒRTĒJUMA LAPA

Bakalaura darbs izstrādāts ***Sistēmu teorijas un projektēšanas katedrā.***

Ar parakstu apliecinu, ka visi izmantotie materiāli ir norādīti literatūras sarakstā un iesniegtais darbs ir oriģināls.

Darba autors:
stud. **S. Luhmirins** …….....…….....
(paraksts, datums)

Bakalaura darbs ieteikts aizstāvēšanai:

Zinātniskais vadītājs:
Mg.sc.ing., pētnieks **A. Liekna** …….....……...
(paraksts, datums)

Bakalaura darbs pielaists aizstāvēšanai:

Sistēmu teorijas un projektēšanas katedras vadītājs:
Dr.habil. sc.ing., prof. **J.Grundspeņķis**.....……...
(paraksts, datums)

Bakalaura darbs aizstāvēts Sistēmu teorijas un projektēšanas katedras Gala pārbaudījumu

komisijas ……...gada…….....…sēdē un novērtēts ar atzīmi ( )….....….……..
(gads) (datums, mēnesis)

Sistēmu teorijas un projektēšanas katedras Gala pārbaudījumu komisijas

sekretāre…….....……
(uzvārds, paraksts)

ANOTĀCIJA

EVOLUCIONĀRA SKAITĻOŠANA, MOBILU ROBOTU VADĪBA, NEIRONU TĪKLI

Bakalaura darba "Evolucionārās skaitļošanas pielietošanas analīze mobila robota vadībai" ietvaros ir darbībā pārbaudīta ģenētiskā algoritma izmantošana neironu tīkla svaru noskaņošanai ar mērķi automatizēti iegūt robota vadības mehānismu.

Darba teorētiskajā daļā apkopota teorētiskā bāze par evolucionāro skaitļošanu un tās pielietojumiem robotikā.

Praktiskajā daļā izveidots mobila robota imitācijas modelis un izstrādāta programmatūra neironu tīklu veidošanai ar ģenētisko algoritmu. Ar izveidotajiem rīkiem praktiski analizēts dažādu neironu tīklu topoloģiju pielietojums šķēršļu apiešanas uzdevuma risināšanai.

Bakalaura darba nobeigumā ir secinājumi par darbā iegūtajiem rezultātiem, kā arī identificēti turpmāko pētījumu virzieni.

Darbā iekļauts viens pielikums ar izstrādātās programmatūras lietošanas instrukciju.

Darba pamattekstā ir 61 lappuses, 30 attēli, 1 tabula, 39 nosaukumu informācijas avoti un 1 pielikums.

ABSTRACT

EVOLUTIONARY COMPUTATION, MOBILE ROBOT CONTROL, NEURAL NETWORKS

Bachelor thesis “Analysis of evolutionary computation applications for mobile robot control” provides a practical overview of genetic algorithm use for neural network training with purpose to gain automatized robot control mechanism.

Theoretical part includes summary of theoretical basis for evolutionary computation and its applications in robotics.

Practical part includes created mobile robot imitation model and software for neural network creation by means of genetic algorithm. Created tools were used to analyse different neural network topology applications for obstacle avoidance task.

Conclusion contains discussion of attained results and identified topics for future research in this field.

Appendix contains user’s guide for developed software.

The thesis contains 61 pages, 30 figures, 1 table, 39 information sources and 1 appendix.

АННОТАЦИЯ

ЭВОЛЮЦИОННЫЕ ВЫЧИСЛЕНИЯ, УПРАВЛЕНИЕ МОБИЛЬНЫМИ РОБОТАМИ, НЕЙРОННЫЕ СЕТИ

В бакалаврской работе «Анализ применений эволюционных вычислений в управлении мобильных роботов» практически проверено использование генетического алгоритма для настройки нейронных сетей, используемых в механизмах управления мобильных роботов.

В теоретической части работы обобщены теоретические основы эволюционных вычислений и их применений в робототехнике.

В практической части разработана имитационная модель робота и программа для создания нейронных сетей используя генетический алгоритм. С помощью разработанных инструментов проведен практический анализ применения различных топологий нейронных сетей, используемых в решении задач уклонения от препятствий.

В заключении работы предоставлены выводы о достигнутых результатах работы, а также определены темя для будущих исследований.

Приложения содержат руководство пользователя для разработанных инструментов.

Объём работы: 61 страница, 30 изображений, 1 таблица, 39 литературных источника и 1 приложение.

SATURS

IEVADS

Pēdējos gados pieaug mobilu robotu pielietošanas sfēru skaits, tas nozīmē, ka robotiem jāspēj darboties arvien sarežģītākā vidē. Šādu robotu vadības mehānismu izveide prasa daudz laika un ieguldītā darba, tāpēc zinātnieki meklē pieejas šī procesa automatizācijai. Viena no pieejām robotu vadības mehānismu veidošanas automatizācijai ir evolucionārā skaitļošana. Evolucionārās skaitļošanas pielietošanu robotikā dēvē par evolucionāro robotiku, kuras galvenais mērķis ir izveidot metodes robotu vadības sistēmu automatizētai iegūšanai. Viens no evolucionārās robotikas pētījumu virzieniem ir uz neironu tīkliem balstītu robotu vadības mehānismu automatizēta iegūšana, pielietojot ģenētiskos algoritmus neironu tīklu svaru noskaņošanai, kas arī ir pētīts šajā bakalaura darbā.

Darba mērķis ir analizēt evolucionārās skaitļošanas pielietošanu mobila robota vadības mehānismu automatizētai iegūšanai. Definētā mērķa sasniegšanai izvirzīti šādi **darba uzdevumi:**

1. Apkopot evolucionārās skaitļošanas teorētisko bāzi, analizēt ģenētiskā algoritma pielietošanu neironu tīklu apmācībā.
2. Identificēt un analizēt risinājumus mobila robota vadības mehānisma iegūšanai ar evolucionāro skaitļošanu.
3. Izstrādāt un darbībā pārbaudīt mobila robota imitācijas modeli, kura vadības mehānisms tiek iegūts automātiski, pielietojot evolucionāro skaitļošanu.
4. Veikt eksperimentus ar izstrādāto modeli, novērtēt eksperimentāli iegūtos rezultātus un izdarīt atbilstošus secinājumus

Darba *pirmajā nodaļā* aprakstīti robotikas pamatjēdzieni, apkopoti autonomu robotu vadības mehānismu veidi, identificēti neironu tīklos sakņotie robota vadības modeļi.

Otrajā nodaļā apkopoti neironu tīklu pamatjēdzieni, kā arī veikts esošo neironu tīklu topoloģiju apkopojums un analīze. Identificēti neironu tīklu apmācības veidi, kuros ietilpst arī evolucionārā skaitļošana.

Trešajā nodaļā ir apkopota teorētiskā bāze par evolucionārās skaitļošanas metodēm un to īpašībām. Pamatojoties uz izvirzītajiem darba uzdevumiem, detalizēti analizēts tieši ģenētiskais algoritms – tā sastāvdaļas un pielietojumi neironu tīklu veidošanai un noskaņošanai. Analizēti arī evolucionārās skaitļošanas pielietojumi robotikā, identificētas un analizētas risināmās problēmas. Analīzes rezultātā izvirzīts praktiskās daļas pētījumu objekts.

Ceturtajā nodaļā praktiski analizēta robota vadības mehānisma automatizēta iegūšana, pielietojot dažādu topoloģiju neironu tīklus, kuru svari tiek noskaņoti ar ģenētisko algoritmu. Nodaļā aprakstīts vadības mehānisma iegūšanas mērķis un plāns, dots izveidotā imitācijas modeļa un izstrādātās programmatūras apraksts, kā arī pielietotā ģenētiskā algoritma parametru izvēles pamatojumus. Izmantojot izstrādāto imitācijas modeli un programmatūru veikti praktiski eksperimenti automatizēti iegūstot robota vadības mehānismu šķēršļu apiešanas uzdevuma risināšanai. Nodaļā aprakstīti un analizēti veikto eksperimentu rezultāti.

Darba nobeigumā ir *secinājumi* par paveikto darbu, sasniegtajiem rezultātiem, norādīta darba mērķa un izvirzīto darba uzdevumu izpildes pakāpe, kā arī identificēti turpmāko pētījumu virzieni.

Darbam ir viens pielikums ar praktiskajā daļā izstrādātās un izmantotās programmatūras lietošanas instrukciju.

1. IEVADS ROBOTIKĀ

1.1. Robota jēdziens

Vārdu "robots" popularizēja čehu rakstnieks Karels Čapeks (Karel Čapek) ar 1920. gadā uzrakstīto lugu "Rossuma Universālie Roboti". Termins izveidojies apvienojot čehu vārdus *rabota* (no čehu val. darbs) un *robotnik* (no čehu val. darbinieks) (Angelo, 2007).

Robots ir autonoma, pārprogrammējama, daudzfunkcionāla mehāniska sistēma ar vairākām brīvības pakāpēm (Khalil & Dombre, 2004), kas eksistē fizikālajā pasaulē, spēj uztvert ārējo vidi un veikt darbības kāda mērķa sasniegšanai (Matarić, 2007), veidota vienveidīgu, atkārtojošu, bīstamu uzdevumu vai citu operāciju veikšanai, tiešā cilvēku kontrolē vai patstāvīgi, izmantojot iebūvētās instrukcijas vai mākslīgo intelektu. Robotu veido šādas komponentes (Matarić, 2007; Khalil & Dombre, 2004):

1. Ķermenis – mehāniska konstrukcija, kas satur visus robota elementus kopā.
2. Sensoru kopa – uztver robota stāvokli un apkārtējo vidi.
3. Izpildmehānismi – nodrošina kustību un iedarbību uz apkārtējo vidi.
4. Vadības mehānisms – pārveido sensoru sniegto informāciju izpildmehānismu vadības signālā saskaņā ar iekšējo programmu.
5. Komunikācijas interfeiss – nodrošina komandu saņemšanu un informācijas nodošanu lietotājam.

Robots var būt statisks (fiksēts vienā vietā) vai mobils (Angelo, 2007). Šis darbs ir veltīts mobiliem robotiem. Par mobiliem tiek dēvēti roboti, kuri ir spējīgi brīvi pārvietoties noteiktā vidē – uz zemes, ūdenī, gaisā vai kosmosā. Atšķirībā no fiksētiem robotiem, kuri pilda vienveidīgas strukturētas operācijas, mobili roboti spēj pildīt daudzveidīgāku uzdevumu klāstu (Cook, 2011). Statiski roboti pārsvarā tiek pielietoti ražošanas līniju vajadzībām, tādām kā materiālu pārvietošana un metināšana. Turpretim mobiliem robotiem atrasti daudzi pielietojumi pakalpojumu sfēra (dažādu veidu tīrītāji), militārajā sfērā (izlūklidmašīnas), medicīnā (roboti aukles), zinātnē (pētnieciskie kosmosa aparāti) un izklaidē (piemēram, platforma Lego Mindstorm) (Bekey, 2008).

Eksistē divas pieejas robotu vadībā – manuālā un autonomā vadība. Autonomā vadība paredz, ka robots ir spējīgs patstāvīgi pieņemt lēmumus un darboties bez jebkādas lietotāja iejaukšanās, sekmīgi izpildot uzstādītos uzdevumus. Daļēja autonomija nozīmē, ka lietotājs

sniedz tikai augstākas pakāpes komandas un to izpildes detaļas tiek atstātas paša robota ziņā. Manuālās vadības gadījumā robots kalpo par starpnieku starp lietotāju un vidi kura atrodas robots (Cook, 2011). Sakarā ar izvirzītajiem darba mērķiem un uzdevumiem, šajā darbā tiek apskatīta tikai mobilo robotu autonomā vadība.

Uzdotā uzdevuma izpildes laikā mobils robots var sastapties ar dažādā veida šķēršļiem, kurus ir svarīgi atklāt pirms notiek sadursme (Cook, 2011). Par šķērsli tiek uzskatīts jebkurš fizikāls objekts, kas traucē robota kustībai un nav saistīts ar uzdotā uzdevuma izpildi, piemēram, sienas, mēbeles, akmeņi un pat cilvēki (Cook, 2011). Kaut arī eksistē daudzi pētījumi šajā jomā, robotu vadības problēma joprojām ir nopietns izaicinājums, lielā nezināmo skaita dēļ. (Matveev, Wang u.c., 2012). Mobilu roboti vadību statisku vai kustīgu šķēršļu klātbūtnē sauc par kustības plānošanu. Izveidot ātru un efektīvu kustības plānošanu ir viens no svarīgākajiem uzdevumiem mūsdienu robotikā (Abiyev, Ibrahim u.c., 2010). Mobila robota kustības plānošana sastāv no četriem posmiem (Tzafestas, 2014):

1. Vides uztveršana – ar sensoriem robotam jāspēj apkopot un interpretēt informāciju par apkārtējo vidi un robota stāvokli.
2. Lokalizācija – robotam ir jānosaka sava pozīcija apkārtējā vidē.
3. Lēmumu pieņemšana – balstoties uz iepriekšējiem soļiem robots pieņem lēmumu par mērķa sasniegšanai nepieciešamo soļu izpildi
4. Kustības vadība - tiek veidots izpildmehānismu vadības signāls saskaņā ar pieņemto lēmumu.

Visas autonoma robota mijiedarbības ar ārējo vidi notiek ar sensoru un izpildmehānismu palīdzību, tāpēc to izvēle ir svarīgs posms robota izveidē un tiem ir veltīta nākošā apakšnodaļa.

1.2. Sensori un izpildmehānismi

Informāciju par apkārtējo vidi roboti saņem ar sensoriem, bet iedarbībai uz ārējo vidi tiek izmantoti dažāda veida izpildmehānismi (Tzafestas, 2014).

Robots var būt aprīkots ar dažādu veidu sensoriem, kas uztver informāciju gan par robota iekšējo sistēmu stāvokli, gan par ārējo vidi un robota novietojumu tajā. Sensoru klasifikācijai tiek izmantoti divi raksturojumi: ārējs/iekšējs un pasīvs/aktīvs (Cook, 2011):

1. *Iekšējs* sensors mēra iekšējo sistēmu parametru vērtības, piemēram, motoru ātrumu, riteņu noslodzi, baterijas lādiņu u.tml.

2. *Ārējs* sensors sniedz informāciju par ārējo vidi, piemēram, attālumu līdz tuvākajiem objektiem, gaismas intensitāti u.tml.
3. *Pasīvs* sensors tikai uztver enerģiju no ārējās vides. Pasīvo sensoru piemēri: temperatūras sensors, kompass, kamera.
4. *Aktīvs* sensors izdala enerģiju un mēra ārējās vides reakciju. Aktīvo sensoru piemēri: ultraskaņas sensors, radars.

Tabulā 1.1. apkopoti robotikā biežāk izmantoto sensoru veidi.

1.1. tabula. Sensoru klasifikācija (aizgūts no (Tzafestas, 2014))

Kopēja klasifikācija (pielietojums)	**Sensors, sensoru sistēma**	**Iekšējs vai ārējs**	**Aktīvs vai pasīvs**
Pieskāriena sensori	Saskares slēdzis	Ārējs	Pasīvs
(nosaka fizisku kontaktu)	Optiskā barjera	Ārējs	Aktīvs
	Tuvuma slēdzis	Ārējs	Aktīvs
Riteņu/motoru sensori	Potenciometrs	Iekšējs	Pasīvs
(nosaka riteņu ātrumu un pozīciju)	Optisks enkoderis	Iekšējs	Aktīvs
	Magnētisks enkoderis	Iekšējs	Aktīvs
	Induktīvs enkoderis	Iekšējs	Aktīvs
	Kapacitatīvs enkoderis	Iekšējs	Aktīvs
Virziena sensors	Kompass	Iekšējs	Pasīvs
	Žiroskops	Iekšējs	Pasīvs
Pozicionēšanas sistēma	GPS	Ārējs	Aktīvs
(nosaka atrašanas vietu)	Ultraskaņas bāka	Ārējs	Aktīvs
Aktīva attāluma mērīšana	Atstarošanas sensori	Ārējs	Aktīvs
	Ultraskaņas sensori	Ārējs	Aktīvs
	Lāzera tālmērs	Ārējs	Aktīvs
Redzes sensori	CCD/CMOS Kamera(s)	Ārējs	Pasīvs

Bieži uzstādītā uzdevuma izpildei nepietiek tikai ar apkārtējās vides uztveršanu, robotam jāspēj veikt kāds darbs. Šim nolūkam tiek izmantoti izpildmehānismi.

Izpildmehānismi ir ierīces, kas ļauj robotam iedarboties uz ārējo vidi. Pie tiem pieskaita dažādu veidu motorus, hidrauliskos un pneimatiskos cilindrus, siltumjūtīgus elementus un citas tehnoloģijas. Izpildmehānismi darba veikšanai patērē enerģiju. Robotos visbiežāk sastopami ar elektrību darbināmi izpildmehānismi. Tomēr eksistē arī pasīvie izpildmehānismi, kuri izmanto tikai ķermeņu potenciālo enerģiju.

Izpildmehānismi kustina riteņus, sliedes, robotu rokas un citus efektorus. Efektors ir izpildmehānisma daļa, ar kuru robots tiešā veidā iedarbojas uz apkārtējo vidi (Matarić, 2007).

Visbiežāk izmantoto izpildmehānismu veidi darbības principi (Matarić, 2007):

1. Elektriskais motors – ar magnētisko lauku palīdzību pārvērš elektrisko enerģiju rotācijas kustībā (visbiežāk pielietoti, pateicoties vienkāršai ekspluatācijai un pieejamībai).
2. Hidrauliskie un pneimatiskie cilindri – balstīti uz attiecīgi šķidruma vai gaisa spiediena izmaiņām.
3. Foto-reaktīvie materiāli – darbojas gaismas iedarbībā (parasti tiek pielietoti mikro-izmēru robotos).
4. Ķīmiski-reaktīvie materiāli – saraujas un izplēšas dažādu ķīmisko vielu iedarbībā.
5. Siltumjūtīgi materiāli – reaģē uz temperatūras izmaiņām.

Sasaiste starp sensoriem un izpildmehānismiem notiek ar vadības sistēmas palīdzību.

1.3. Robota vadības sistēma

Atkarībā no pieejamajiem resursiem un zināšanām par vidi, kurā darbosies robots, var izdalīt divas kategorijas – kartē balstīta un uzvedībā balstīta vadības sistēma. Katra kategorija piedāvā atšķirīgu skatījumu uz robotu pozicionēšanu un izmanto dažādus algoritmus kustības plānošanai (Cook, 2011).

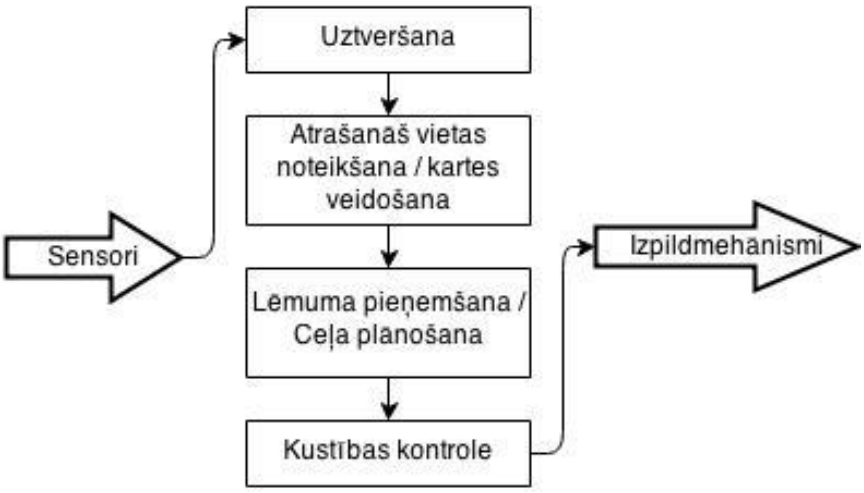
Kartē balstītā vadības sistēma veido vai izmanto jau gatavu ārējās vides modeli, kurā arī nosaka savu atrašanas vietu, mērķa stāvokli un nepieciešamos manevrus (Stenning & Barfoot, 2012). Šīs pieejas arhitektūra ir parādīta 1.1. att. attēlā. Šādā gadījumā vidi uzskata par pilnībā vai daļēji zināmu un izmanto globālā ceļa plānošanas algoritmus (Abiyev, Ibrahim u.c., 2010):

1. Mākslīgais potenciālu lauks.
2. Attāluma funkcijas pieeja.
3. Vektoru laika histogramma.
4. Dinamiska loga pieeja.

Ja ir zināms, ka ārējā vide ir nemainīga, tad kustības trajektoriju ir nepieciešams aprēķināt tikai pirms kustības uzsākšanas. Dinamiskā vidē trajektorija tiek pārrēķinātā vai

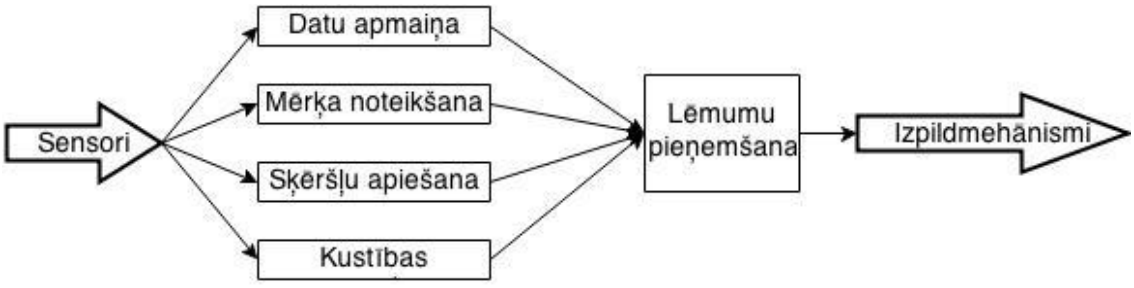
vismaz pārbaudīta katrā solī. Abos gadījumos, tas prasa ievērojamus skaitļošanas un atmiņas resursus, kuri autonoma robota gadījumā ne vienmēr var būt pieejami (Cook, 2011)..

Galvenais šādas pieejas trūkums ir reālās pasaules un robota izveidotā modeļa atšķirības, kas var traucēt robotam izpildīt uzstādītos uzdevumus, un pat novest pie bojājumiem. Šīs problēmas risināšanai izmanto ticamības attēlojumu, laikā mainīgas kartes un varbūtību kartes (Tzafestas, 2014).



1.1. att. Kartē balstītas vadības sistēmas shēma (aizgūts no (Tzafestas, 2014))

Uzvedībā balstīta vadības sistēmai nepieciešami mazāki skaitļošanas resursi, jo apkārtējās vides modelis netiek veidots vispār, tā vietā tiek pielietotas dažādas iepriekš izveidotas uzvedības stratēģijas. Izdevīgākās stratēģijas noteikšanai izmanto sensoru rādījumus. Katra no stratēģijām ir speciāli piemērota kāda mērķa īstenošanai, piemēram, iziet labirintu vai sekot kreisajai sienai (Tzafestas, 2014). Pieejas shēma parādīta 1.2. att. attēlā.



1.2. att. Uzvedībā balstīta vadības sistēmas shēma (aizgūts no (Tzafestas, 2014))

Tā kā robotam nav pieejama informācija par apkārtējo vidi – vide ir pilnīgi nezināma, šķēršļu apiešanai tiek izmantota lokālās kustības plānošana (Matveev, Wang u.c., 2012). Lokālās kustības plānošanā notiek dinamiska ārējās vides analīze uz kuras pamata tiek veiktas izmaiņas kustībā. Tas nozīmē, ka lokālās kustības plānošanai nav nepieciešamas glabāt

informāciju par vidi. Šādas pieejas galvenā priekšrocība ir viegla implementācija kādai noteiktai videi, tomēr šāds risinājums nevar būt universāls jebkurai situācijai, tātad ir uzmanīgi jāizvēlas un jāpielāgo uzvedību konkrētajai videi. Citas problēmas var rasties liela uzvedības stratēģiju daudzuma dēļ – lēmumu pieņemšanas process kļūst apgrūtināts, jo vadības sistēmai jāizvērtē visu stratēģiju piemērotību katrai situācijai (Tzafestas, 2014).

Izvēlēto stratēģiju klāsts ir atkarīgs no robota uzdevuma un vides, kurā robots darbosies. Tomēr neatkarīgi no abiem minētajiem faktoriem, mobilam robotam ir nepieciešama kustības stratēģija šķēršļu apiešanai (Petrič & Žlajpah, 2013). Šķēršļu apiešanas stratēģijas pielieto kādu no šādiem principiem (Abiyev, Ibrahim u.c., 2010; Wang, Tan u.c., 2006):

1. Likumu kopa – produkcijas likumi, kas pielāgoti robota darbības videi. Nezināmas vides gadījumā šīs pieejas pielietošana ir apgrūtināta, jo paredzēt visus iespējamos scenārijus nav iespējams. Tādos gadījumos izmanto nestriktās loģikas paņēmienus (Abiyev, Ibrahim u.c., 2010).
2. Nestrikta loģika – vadības sistēma sastāv no nestrikto likumu bāzes un piederības funkcijām (ieejas un izejas vērtības), kas balstās uz ekspertu zināšanām (Samsudin, Ahmad u.c., 2011).
3. Neironu tīkli – vadības sistēma sensoru datu apstrādei un vadības signāla veidošanai, izmanto mākslīgo neironu tīklu. Pateicoties mākslīgo neironu tīklu īpašībām, šādas vadības sistēmas ir efektīgas trokšņainu un nepilnu datu gadījumā (Yoo, 2013).

Eksistē arī citas pieejas, kā arī minēto pieeju apvienojumi, kuri noteiktos apstākļos dod labāku rezultātu. (Lin & Lian, 2009).

Sakarā ar to, ka viens no darba uzdevumiem ir saistīts ar neironu tīklu pielietojumu analīzi, šajā darbā tiek apskatītas tikai neironu tīklos balstītas vadības sistēmas. Nākošajā nodaļā tiek aprakstīti ar neironu tīkliem saistītie pamatjēdzieni.

2. MĀKSLĪGIE NEIRONU TĪKLI

2.1. Mākslīgo neironu tīklu jēdziens

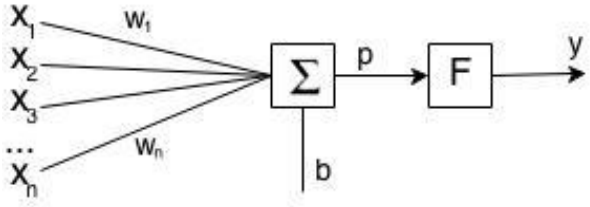
Mākslīgs neironu tīkls ir dzīvo organismu nervu šūnu mijiedarbības matemātisks modelis, kā arī šī modeļa atveidojums elektroniskos elementos vai datorprogrammā. Mākslīgajiem neironu tīkliem piemīt spējas risināt problēmas un glabāt zināšanas. Matemātiski mākslīgo neironu tīklu atveido kā svērtu orientētu grafu, kura virsotnes sauc par neironiem un šķautnes par sinapsēm (Graupe, 2013). Neironi ir izvietoti slāņos tādā veidā, ka starp viena slāņa neironiem nepastāv saites. Slāņu skaits un neironu skaits slānī, kā arī saites starp neironiem ir atkarīgi no risināmā uzdevuma un izvēlētā neironu tīkla veida (Zhang, 2010). Neironu tīklu darbību nosaka tā uzbūve un savienojumu svari. Zināšanas ir sadalītas visā tīklā. Tīkla uzvedību nosaka neironu aktivizācijas modelis pie noteiktām ieeju vērtībām. Neironu tīklu noskaņošana notiek mainot sinapšu svarus. Neironu tīklus pielieto, jo tiem piemīt šādas īpašības (Floreano & Mattiussi, 2008):

1. Stabilitāte – neironu tīkli ir noturīgi pret ieejas signāla degradāciju un trokšņiem, kā arī pret paša tīkla fiziskās realizācijas nepilnībām un bojājumiem. Kļūdas var rasties pie noteiktām ieejas datu kopām, kamēr pārējos gadījumos atbildes būs pareizas.
2. Elastība – neironu tīkli nav piesaistīti noteiktai nozarei vai problēmu kopai. Tos var pielietot daudzās jomās visdažādāko uzdevumu risināšanai, tomēr nevar teikt, ka jebkurš tīkls ir piemērots visu uzdevumu izpildīšanai.
3. Vispārinājums – neironu tīkli, apmācīti ar galīgu skaitu piemēru, ir spējīgi veiksmīgi klasificēt vel neredzētus piemērus, kas ir noteiktā mērā līdzīgi jau redzētajiem. Šī īpašība ir īpaši noderīga gadījumos, kad nav iespējams iegūt pilnīgu sarakstu ar visām ieejas vērtību kombinācijām.
4. Uz satura balstīta atgriešana (angļu val. *content-based retrieval*) – neironu tīkli atgriež datus balstoties uz to satura, pat pie trokšņainiem un bojātiem ieejas datiem. Kas stipri atšķiras no klasiskās pieejas, kurā datu atgriešanai izmanto adresi atmiņā. Ja šāda adrese ir bojāta, datus atgūt vairs nebūs iespējams.

Neironu tīklu galvenā sastāvdaļa ir neirons, tāpēc tas tiek sīkāk aprakstīts nākošajā apakšnodaļā.

2.2. Neirons

Katram neironam ir viena vai vairākas ieejošās sinapses un viena izejošā vērtība, kas var tikt nodota kā ieeja vairākiem citiem neironiem. Neirons sevī ietver divas funkcijas– summēšanu un aktivizācijas funkciju. Attēlā 2.1. att.shematiski attēlota neirona ar n ieejām uzbūve, kur x_1-x_n ir ieejas vērtības, w_1-w_n - ieejas vērtību svari, b – neirona papildus svars, p – summēšanas funkcijas rezultāts, F – aktivizācijas funkcija un y ir izejas vērtība (Zhang, 2010).



2.1. att. Neirona ar n ieejas sinapsēm uzbūve (aizgūts no (Zhang, 2010))

Matemātiski to pašu var izteikt ar funkciju (2.1):

$$y = F(\sum_{i=1}^{n} w_i x_i + b) \,, \tag{2.1}$$

kur y – neirona izejas vērtība;

F() – aktivizācijas funkcija;

n – neirona ieeju skaits;

w_i – kārtējās ieejas svars;

x_i – kārtējās ieejas vērtība;

b – neirona papildus svars.

Summēšanas funkcijas darbības princips ir apvienot visu ieejas vērtību un to svaru reizinājumus. Neirona papildus vērtību sauc arī par nobīdi, to pievieno, lai pārvietotu aktivizācijas funkciju pa horizontālo asi, tādā veidā regulējot neirona jūtīgumu. Neskatoties uz nosaukumu, summēšanas funkcija var ietvert sevī citas matemātiskās operācijas, piemēram maksimuma vai minimuma noteikšana (Graupe, 2013).

Aktivizācijas funkcija nosaka neirona aktivizēšanos un izejas vērtības atkarību no ieejas vērtību apvienojuma. Par aktivizācijas funkciju var kalpot jebkāda nepārtraukta funkcija, vairākumā gadījumu šī funkcija ir monotoni augoša vai pakāpeniska. Balstoties uz (Sibi, Jones u.c., 2013) pētījumu, populārākie aktivizācijas funkciju veidi ir šādi. Matemātiskajās izteiksmēs lietotie apzīmējumi: x – funkcijas arguments, y – funkcijas vērtība, s – stāvuma koeficients, g – sliekšņa vērtība.

1. Lineāra funkcija – vērtības var būt ierobežotas noteiktā intervālā [A,B]. Šo funkciju apraksta ar formulu (2.2).

$$y = x \cdot s \tag{2.2}$$

2. Sliekšņveida funkcija – atgriež vērtību 1 tikai ja arguments ir lielāks par noteiktu sliekšņa vērtību, citos gadījumos tiek atgriezta 0. Šo funkciju apraksta nevienādību sistēma (2.3).

$$y = \begin{Bmatrix} 0; x \le g \\ 1; x > g \end{Bmatrix} \tag{2.3}$$

3. Sigmoīda – apraksta ar izteiksmi (2.4), atgriež vērtības apgabalā [0,1]. Pēc pētījuma (Sibi, Jones u.c., 2013) rezultātiem, tā ir visbiežāk izmantotā aktivizācijas funkcija.

$$y = \frac{1}{(1 + e^{-xs})} \tag{2.4}$$

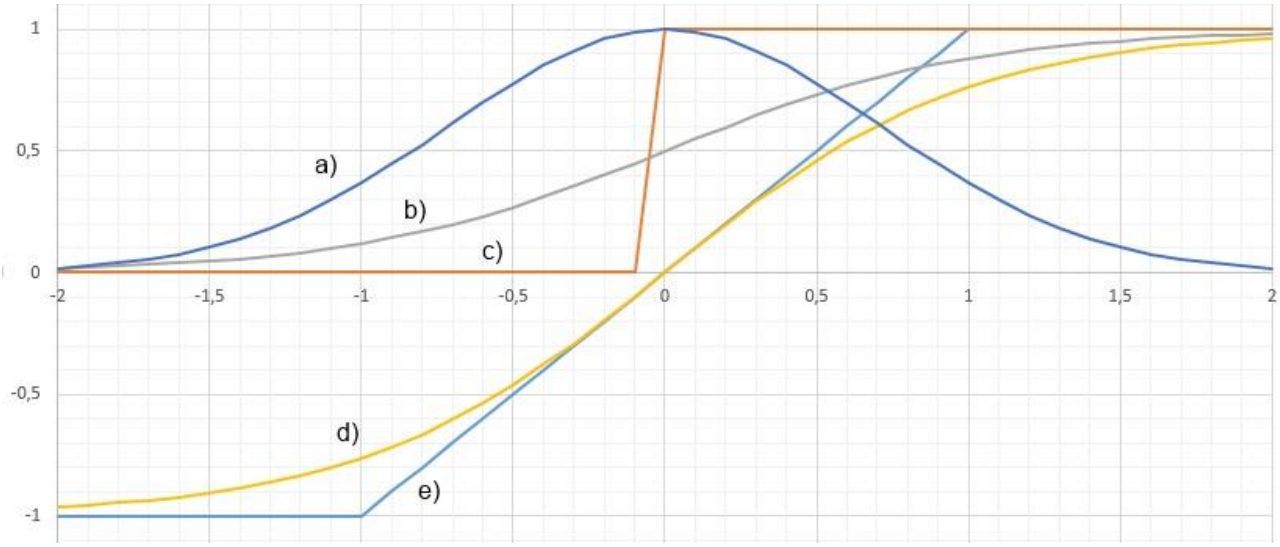
4. Hiperboliskā tangensa funkcija – dažreiz sauc arī par sigmoidāli simetrisku funkciju, apraksta izteiksme (2.5), atgrieztās vērtības ir intervālā [-1,1]. Var tikt aizvietota ar pakāpienveida tuvinājumu.

$$y = \tanh(xs) = \frac{2}{(1 + e^{-2xs})} - 1 \tag{2.5}$$

5. Gausa funkcija – apraksta ar izteiksmi (2.6). Gausa funkciju pielieto, kad ir nepieciešams kontrolēt intervālu, kurā neirons tiek aktivizēts.

$$y = e^{-x^2 s^2} \tag{2.6}$$

Attēlā 2.2 parādīti minēto aktivizācijas funkciju grafiki intervālā [-2,2] pie s=1, lineāra funkcija ir ierobežota intervālā [-1,1] un sliekšņveida funkcijas gadījumā g=0.



2.2. att Aktivizācijas funkciju grafiki: a) Gausa funkcija; b) Sigmoidāla funkcija; c) siekšņveida funkcija; d) hiperboliskā tangensa funkcija; e) lineārā funkcija

Atkarībā no neirona izvietojuma tīklā, tos iedala trijās grupās (Graupe, 2013):

1. Ieejas neironi – saņem ieejas vērtības no ārpuses.
2. Iekšējo slāņu neironi – saņem un nodod vērtības citiem neironiem.
3. Izejas neironi – saņem vērtības no citiem neironiem, atgriež savas vērtības kā tīkla aprēķina rezultātu.

Pēc ieejas datu tipa, neironi var būt analogie – ieeju vērtības ir reāli skaitļi (parasti intervālā [0,1] vai [-1,1], bet var būt arī bez ierobežojumiem), un binārie – ieeju vērtības var būt tikai 0 vai 1 (Graupe, 2013).

Neironu savstarpējie savienojumi tīklā veido tīkla topoloģiju.

2.3. Neironu tīklu topoloģijas

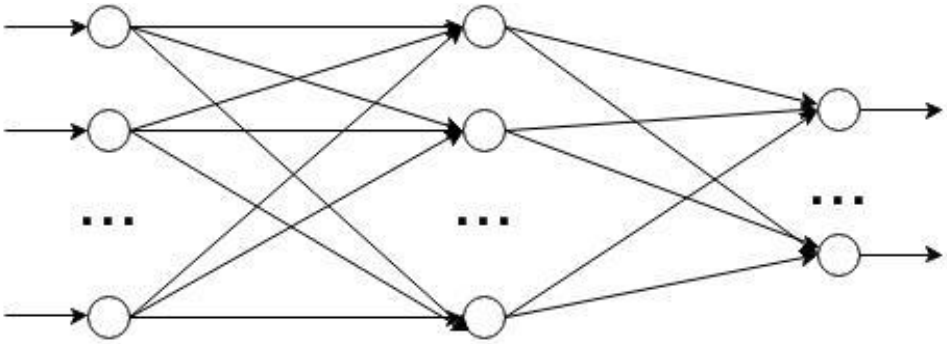
Neironu tīkla topoloģija ir shematisks neironu un to savienojumu attēlojums vai apraksts. Veiksmīgai neironu tīklu pielietošanai nepieciešams izvēlēties piemērotu tīkla topoloģiju, atkarībā no uzstādītā uzdevuma un pieejamo apmācības datu daudzuma. Eksistē četri galvenie tīklu topoloģijas veidi (Zhang, 2010):

1. Vienvirziena (angļu val. *feedforward*) tīkli,
2. Rekurentie, jeb tīkli ar atgriezenisko saiti,
3. Radiālās bāzes funkcijas (angļu val. *radial basis function*),
4. Pašorganizējoši tīkli.

Šie tīklu tipi ir sīkāk aprakstīti turpmākajās apakšnodaļās.

2.3.1. Vienvirziena tīkli

Vienvirziena tīklos neironi saņem ieejas vērtības tikai no neironiem iepriekšējā slānī un atgriež rezultātus tikai neironiem nākošajā slānī. Nekāda veida atgriezeniskās saites šādos tīklos nepastāv. Vienvirziena neironu tīkls var tikt attēlots kā orientēts grafs bez cikliem (2.3. attēls). Pēc savas būtības vienvirziena neironu tīklus var uzskatīt par saliktu funkciju, kas ir veidota apvienojot vairākas nelineāras funkcijas (Zhang, 2010). Pie vienvirziena tīkliem pieskaita Rozenblata perceptronu, daudzslāņu perceptronus un Vorda tīklus (Graupe, 2013)



2.3. att. Vienvirziena neironu tīkls ar 3 slāņiem

Vienkāršākais vienvirziena tīkla piemērs ir perceptrons. Tas ir neironu ar vismaz diviem slāņiem: sensoru slānis (ieejas neironi) un reaģējošais slānis (izejas neironi). Starp šiem slāņiem var atrasties viens vai vairāk slēptie slāni, jeb asociatīvie slāņi (iekšējie neironi). Neskatoties uz vienkāršo uzbūvi perceptrons ir spējīgs risināt sarežģītus klasifikācijas uzdevumus (Graupe, 2013). Palielinot asociatīvo slāņu skaitu, tīkls var modelēt sarežģītākas nelineāras funkcijas (Zhang, 2010).

2.3.2. Radiālās bāzes funkcijas tīkls

Radiālās bāzes funkcijas neironu tīkls ir vienvirzienu tīklu paveids. Tas ir veidots no trīs neironu slāņiem. Pirmajā slānī ir parasti ieejas neironi. Otrā slāņa neironi izmanto radiālo bāzes funkciju kā aktivizācijas funkciju. Izejas neirons apvieno slēpta slāņa funkciju rezultātus ar lineāru funkciju (Qian, 2002).

Radiālās bāzes funkcijas (2.7) vērtība ir atkarīga tikai no attāluma starp funkcijas argumentu un nulles vērtību, kas var būt koordināšu sākumpunkts vai kāds noteikts punkts, ko sauc par centru (Buhmann, 2003). Attāluma noteikšanai parasti izmanto Eiklīda attālumu (2.8), bet var izmantot arī jebkādu citu piemērotu attāluma noteikšanas metodi (Buhmann, 2003).

$$\phi(x,c) = \phi(\|x - c\|), \tag{2.7}$$

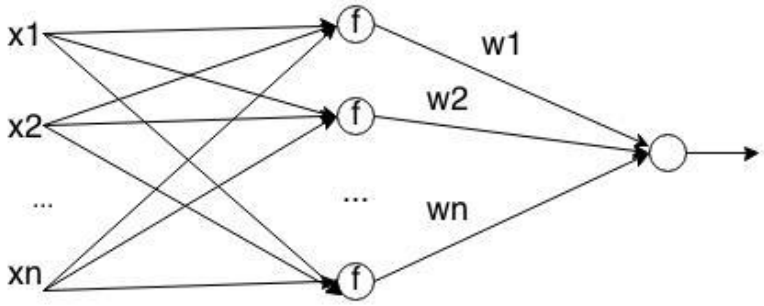
kur x – funkcijas arguments,

c – funkcijas centrs.

$$\|a - b\| = \sqrt{(a-b)\cdot(a-b)}, \tag{2.8}$$

kur a, b – punkti koordinašu plaknē, starp kuriem tiek rēķināts attālums.

Attēlā 2.4. shematiski parādīta radiālās bāzes funkcijas neironu tīkla uzbūve, ar x1-xn apzīmētas ieejas vērtības, f – kāda radiālās bāzes funkcija, w1-wn – funkcijas rezultāta svars.



2.4. att. Radiālās bāzes funkcijas neironu tīkla shēma

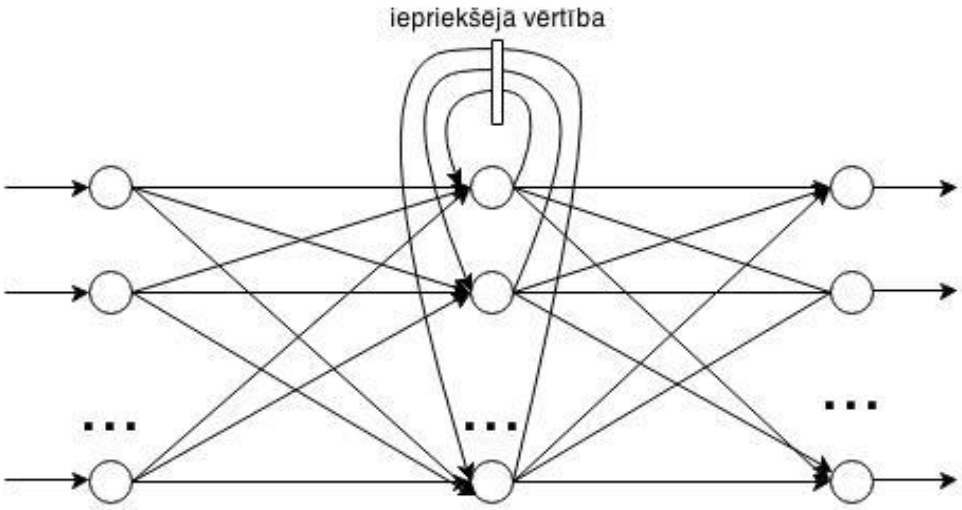
Radiālās bāzes funkcijas tīklu galvenais pielietojums ir nezināmu funkciju tuvināts aprēķins, dinamikas rindu prognozēšana, klasifikācija un vadības sistēmas (Zhang, 2010).

2.3.3. Rekurentie neironu tīkli

Atšķirībā no vienvirziena tīkliem, rekurentajos neironu tīklos eksistē saites uz iepriekšējo līmeņu neironiem. Tāda tīklu uzbūve ļauj izveidoties vienkāršai asociatīvai atmiņai. Atgriezeniskās saites padara neironu tīklu par nelineāru sistēmu ar iespēju veidot daudzveidīgu un sarežģītu uzvedību (Zhang, 2010).

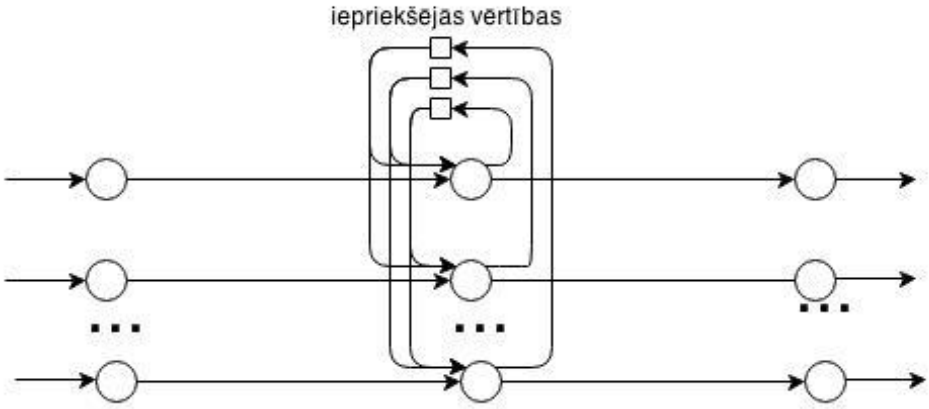
Rekurentajos tīklos visi neironi ir vienādi un var būt savienoti ar jebkuru citu neironu. Ja tīkla ieeju un izeju skaiti ir vienādi un ja katra izeja ir tieši savienota ar attiecīgo ieeju, veidojas divu-līmeņu (angļu val. *two-layer*) vienvirziena tīkls (Zhang, 2010). Visbiežāk sastopamie rekurento tīklu veidi ir Elmana neironu tīkls un Hopfilda neironu tīkls (Zhang, 2010).

Elmana neironu tīkls sastāv no vairākiem slāņiem. Pirmais slānis saņem tikai tīkla ieejas vērtības, katrs nākošais slānis iegūst vērtības no iepriekšējā slāņa un atgriezenisko saiti. Pēdējais slānis ir tīkla izeja (Graupe, 2013). Elmana neironu tīkla shēma parādīta attēlā 2.5.



2.5. att. Elmana neironu tīkla shēma

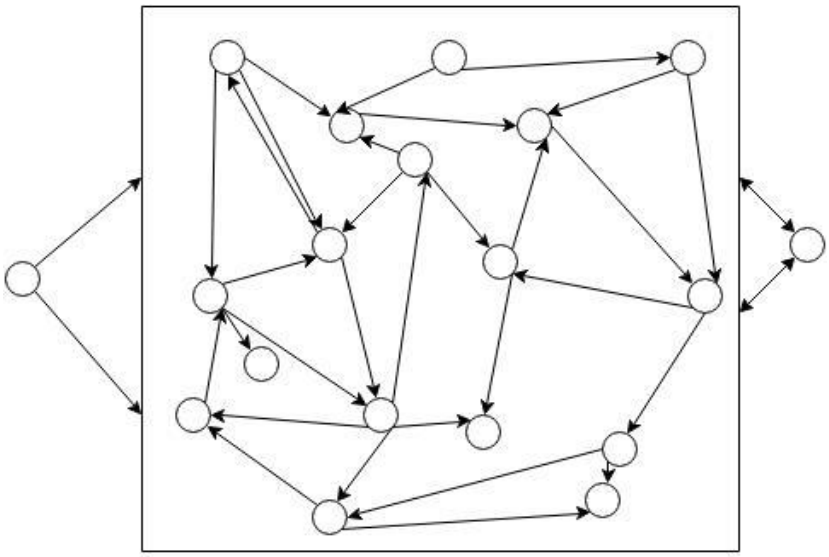
Hopfilda neironu tīklam piemīt visas rekurento tīklu īpašibas un divas papildus īpašības: savienojumu svari ir simetriski ($w_{ij}=w_{ji}$) un neirons neatgriež vērtību sev ($w_{ii}=0$). Hopfilda tīklus izmanto viena vai vairāku mērķa stāvokļu glabāšanai, kas ļauj tos pielietot tēlu atpazīšanai, optimizācijas uzdevumam, un analogi—digitālajai pārveidošanai (Zhang, 2010). Hopfilda tīkla shematiskā uzbūve ir parādīta 2.6. att. attēlā.



2.6. att. Hopfilda neironu tīkls

Cits rekurento neironu tīklu veids ir atbalss stāvokļu tīkls (angļu val. *echo state network*). Šādā tīklā starp ieejas un izejas slāņiem pastāv slēptais slānis ar lielu skaitu (50-1000) nejaušā veidā savienotu neironu, tādejādi veidojot vairākus brīvi savienotus apakštīklus. Mainīti tiek tikai svari savienojumiem starp slēptā un izejas slāņu neironiem, visas pārējās

saites ir noteiktas iepriekš un netiek mainītas. Izejas slāņa neironi satur atgriezenisko saiti uz slēpto slāni, ja tāda saite nepastāv, veidojas atbalss stāvokļu tīkla vienkāršota versija – šķidruma stāvokļu mašīna (Floreano & Mattiussi, 2008). Atbalss stāvokļu tīkla shematiska uzbūve parādīta 2.7. att.



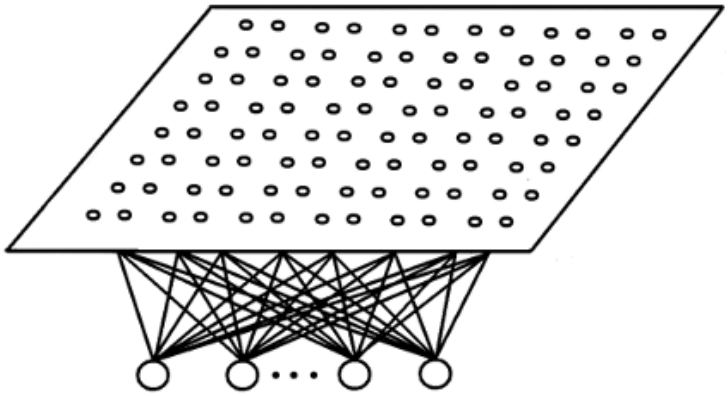
2.7. att. Atbalss stāvokļu tīkls (aizgūts no (Floreano & Mattiussi, 2008))

2.3.4. Pašorganizējoši neironu tīkli

Pašorganizējošu neironu tīklu apmācībai, atšķirībā no citiem tīklu veidiem, izmanto apmācību bez skolotāja. Apmācība notiek balstoties uz ieejas datos slēptiem likumiem un sakarībām. Šāda veida tīkli ir piemēroti datu primārajai analīzei un grupēšanai (Zhang, 2010). Eksistē dažādas pašorganizējošo tīklu realizācijas, piemēram, Kohonena karte, augoša pašorganizējoša karte un laikā mainīga pašorganizējoša karte. Vairums pašorganizējošo tīklu ir Kohonena kartes paplašinājums.

Kohonena karte sastāv no mezgliem (neironiem), kuru daudzumu nosaka kartes veidotājs. Katru mezglu apraksta divi vektori: svara vektors *w*, kuru veido no tik daudz komponentēm, cik ir ieejas vērtību, un vektors *r* ar mezgla atrašanās vietu kartē. Mezgli atrodas četrstūru vai sešstūru režģa virsotnēs. Režģa šķautnes nosaka saites starp virsotnēm. Apmācības procesā visu mezglu vektoru tuvina ieejas datu vērtībām. Katram datu piemēram tiek izvēlēts mezgls ar vislīdzīgāko svara vektoru. Paša mezgla svarus vēl vairāk pietuvina datu vērtībai, to pašu dara ar noteiktu skaitu mezgla kaimiņu. Rezultātā līdzīgi piemēri ieejas datos būs attēloti ar kartē blakusesošiem mezgliem. Process tiek cikliski atkārts līdz netiek

sasniegta noteikta precizitāte vai zināms skaits iterāciju (Kohonen, 2001). Kohonena kartes arhitektūras shēma ir parādīta 2.8. att.



2.8. att. Kohonena kartes arhitektūra (aizgūts no (Zhang, 2010))

2.4. Neironu tīklu apmācība

Neatkarīgi no neironu tīkla topoloģijas, pirms to varēs pielietot reālam uzdevumam, tam ir jāiziet apmācības process. Neironu tīkla apmācības mērķis ir iegūt spēju risināt noteiktus uzdevumus. Mērķi sasniedz mainot atsevišķu neironu un to saišu īpašības (galvenokārt svarus). Retos gadījumos apmācības procesā var tikt pievienoti vai dzēsti atsevišķi neironi un sinapses (Graupe, 2013).

Apmācības efektivitāti nosaka ar kļūdas funkcijas palīdzību. Šī funkcijas nosaka kādā mērā tīkla atrastais atrisinājums atšķiras no optimālā. Apmācības algoritma uzdevums ir atrast tādu tīklu, kura risinājumiem it mazākā iespējamā kļūdas funkcijas vērtība. Katra uzdevuma atrisinājumam kļūdas funkcija atšķirsies, jo tā izriet no uzdevuma nosacījumiem (Graupe, 2013).

Pēc apmācības veida neironu tīklus iedala trīs kategorijās (Graupe, 2013):

1. Apmācība ar skolotāju. Piemērota, kad eksistē datu kopa ar zināmām pareizām atbildēm, kura tiek piedāvāta neironu tīklam. Rezultāts tiek salīdzināts ar pareizo atbildi, un tiek veiktas attiecīgās svaru izmaiņas proporcionāli izrēķinātajai kļūdai. Kā kļūdas funkciju bieži izmanto vidējo kvadrātisko kļūdu starp tīkla izeju vērtībām un pareizām atbildēm. Šādi apmācības algoritmi ir viegli pielietojami un efektīgi, tomēr ir nepieciešama pietiekami liels datu daudzums pareizai tīkla apmācībai. Apmācība ar skolotāju ir visbiežāk

pielietota tēlu atpazīšanas, klasifikācijas un funkciju aproksimācijas uzdevumiem (Graupe, 2013).

2. Apmācība bez skolotāja – piemērota gadījumos, kad nav pieejama datu kopa ar pareizām ieejas-izejas vērtībām. Rezultāti tiek veidoti balstoties tikai uz ieejas vērtībām un slēptām sakarībām datos. Kļūdas funkcija ir atkarīga no uzdevuma formulējuma un pieņēmumiem par vēlamo tīkla rezultātu. Visbiezāk šis apmācības veids tiek pielietots pašorganizējošiem tīkliem, lai risinātu datu grupēšanas un novērtēšanas uzdevumus (Graupe, 2013).
3. Stimulētā apmācība (no angļu val. *reinforcement learning*) – apmācības vide piedāvā sodu un atalgojumu sistēmu. Šis ir apmācības ar skolotāju veids, kad skolotāju lomu izpilda nevis tīkla veidotājs, bet kāda vide vai tās modelis. Kļūdas funkciju vērtības nosaka vide, bieži vien tās precīzs formulējums nav nosākams. Stimulētu apmācību izmanto vadības uzdevumu risināšanai, kad ir nepieciešams veidot secīgu lēmumu ķēdes, piemēram, spēlēs, robotu vadībā vai medicīnā (Graupe, 2013). Stimulētās apmācības kategorijā ietilpst dažādas evolucionārās skaitļošanas metodes, tai skaitā ģenētiskā programmēšana, daļiņu spieta optimizācija un ģenētiskais algoritms. Šīs metodes tiek sīkāk apskatītas nākošajā nodaļā.

3. EVOLUCIONĀRĀ SKAITĻOŠANA

3.1. Evolucionārās skaitļošanas jēdziens

Evolucionārā skaitļošana ir pētījumu lauks mākslīgajā intelektā, kurā tiek pētīti uz evolucionāriem procesiem, piemēram, ģenētiku un dabisko atlasi balstīti mašīnapmācības algoritmi. Šādi algoritmi tiek pielietoti stohastiska rakstura problēmu risinājumu meklēšanai un optimizācijai (Eberhart & Shi, 2007). Evolucionārās skaitļošanas galvenās īpatnības ir (Eberhart & Shi, 2007; Shi, 2011):

1. Meklēšanā tiek izmantotas iespējamo risinājumu kopas - populācijas.
2. Tiek izmantota tieša risinājumu novērtēšana ar piemērotības funkciju (angļu val. *fitness function*).
3. Parametri bieži tiek iekodēti binārajos vai cita veida simbolos.

Evolucionārās skaitļošanas algoritmus var pieskaitīt pie globālās optimizācijas metodēm. Parasti tiek meklēts kādas nelineāras un bieži daudzdimensiju funkcijas minimums vai maksimums. Grūtības sagādā fakts, ka funkcijai var eksistēt daudzi ekstrēmi, kas tiek saukti par lokāliem atrisinājumiem (minimumiem vai maksimumiem). Algoritmam jāspēj atrast vislabāk piemēroto ekstrēmu vērtību, kas bieži nav triviāls uzdevums (Yang, Xu u.c., 2006).

Klasiskie analītiskie algoritmi bieži nav spējīgi atrast vairāk par vienu lokālo atrisinājumu, jo tie sava darbībā apskata tikai blakusesošos atrisinājumus. Savukārt, evolucionāras skaitļošanas algoritmi vienlaikus apskata daudzu nesaistītu atrisinājumu populāciju, kas algoritma darbības sākumā ir izkliedēta visā atrisinājumu telpā un pakāpeniski konverģē globālā atrisinājumā virzienā (Wang & Zhang, 2007).

Visi evolucionārās skaitļošanas algoritmi pārsvarā seko vienādai procedūrai – izveido sākuma populāciju balstoties uz nejauši izvēlētām vērtībām, aprēķina katra populācijas locekļa derīgumu, izvēlās noteiktu skaitu labāko indivīdu un no tiem veido jaunu paaudzi, atkārto ciklu atgriežoties pie populācijas locekļu piemērotības aprēķina (Eberhart & Shi, 2007; Corne & Fogel, 2003).

Evolucionārās skaitļošanas algoritmus iedala divās kategorijās atkarībā no bioloģiskajiem procesiem, uz kuriem šie algoritmi balstās – evolucionārajos algoritmos

(balstīti uz evolūcijas un dabiskās atlases procesu) un spieta algoritmiem (balstīti uz dažādu sociālo dzīvnieku uzvedības modeļiem) (Eberhart & Shi, 2007; Simon, 2013).

Pie evolucionārajiem algoritmiem pieskaita (Brownlee, 2011; Eberhart & Shi, 2007; Simon, 2013):

1. Ģenētiskais algoritms – meklēšanas un optimizācijas algoritms, kas iekļauj sevī tādus bioloģiskas evolūcijas mehānismus, kā krustošanās, mutācija un dabiskā atlase. Pārsvarā tiek izmantoti optimizācijas uzdevumos.
2. Evolūcijas stratēģijas – līdzīgi ģenētiskajam algoritmam, bet ģenētiskās informācijas apmaiņai starp populācijas locekļiem krustošanās vietā tiek pielietota rekombinācija.
3. Ģenētiskā programmēšana – metode, kas balstās uz programmas koda evolūciju. Parasti tiek mainīta kāda hierarhiska koka struktūra, kura attēlo programmas kodu.
4. Evolucionārā programmēšana – līdzīgs ģenētiskajai programmēšanai, bet programmas kods ir fiksēts un mainīti tiek tikai atsevišķi parametri.
5. Gēnu izteiksmes programmēšana – līdzīgi ģenētiskajai programmēšanai, bet programmas kods tiek veidots noteikta garuma rindās – hromosomās.
6. Neiroevolūcija (angļu val. *neuroevolution*) – ģenētiskā algoritma vai ģenētiskās programmēšanas pielietošana neironu tīklu veidošanai. Algoritmā var tikt mainīta gan tīkla struktūra, gan svari.

Pie spieta algoritmiem pieskaita (Brownlee, 2011; Eberhart & Shi, 2007; Simon, 2013):

1. Daļiņu spieta optimizācija (angļu val. *particle swarm optimisation*) – iespējamie risinājumi tiek attēloti uz daudzdimensiju virsmas. Daļiņas kustās pa šo telpu, pakāpeniski novērtējot visus izietos risinājumus, virzienā uz risinājumu ar augstāko vērtējumu. Ar laiku visas daļiņas atradīsies viena vai vairāku labāko risinājumu apkārtnē.
2. Skudru kolonijas optimizācija – uz varbūtībām balstīta metode, kas ir noderīga optimālā ceļa meklēšanai grafā. Mākslīgas skudras nejaušā veidā kustās grafā. Atrodot mērķa virsotni skudra atgriežas sākuma virsotnē atzīmējot savu ceļu. Ja citas skudras sastop atzīmētu ceļu, tās izpētīs, kas atrodas ceļa galā. Ja šāds ceļš noved pie mērķa virsotnes, jaunas skudras virzīsies uz sākuma virsotni, vēl

vairāk pastiprinot atzīmēto ceļu, padarot to pievilcīgāku citām skudrām. Atzīmējums ar laiku pazūd, tāpēc jo mazāks laiks ir nepieciešams, lai šo ceļu izietu, jo vairāk skudras to izstaigās. Rezultātā īsākais ceļš būs visvairāk atzīmēts.

3. Bišu kolonijas algoritms – tiek modelēta medus bišu uzvedība. Tiek veidoti divi mākslīgo bišu veidi – izlūki un vācēji. Izlūki nejaušā veidā pārvietojas iespējamo risinājumu telpā meklējot potenciāli labas virsotnes. Atrastās virsotnes tiek nodotas vācējiem, kuri pēta šo virsotņu apkārtni meklējot lokālos maksimumus.

Viens no šī darbā izvirzītajiem uzdevumiem saistīts ar ģenētiskā algoritma un neiroevolūcijas analīzi, tāpēc nākošajās nodaļās šie algoritmi tiek apskatīts sīkāk.

3.2. Ģenētiskais algoritms

Ģenētiskā algoritma ideja aizgūta no bioloģiskā evolūcijas procesa. Mākslīgā evolūcija, līdzīgi tās bioloģiskajam prototipam, balstās uz četriem pamatprincipiem: populācija, dažādība, mantošana un selekcija. Evolucionārs process var pastāvēt tikai tad, ja eksistē kāda populācija, kurā ir divi vai vairāk indivīdi. Indivīdi savā starpā ir noteiktā mērā atšķirīgi – dažāds ģenētiskais materiāls, kas arī nosaka indivīda pazīmes un uzvedību. Mākslīgas evolūcijas gadījumā, genotips attēlo kādu no iespējamajiem atrisinājumiem, piemēram, neironu tīkla noskaņošanas gadījumā genotips sastāv no neironu tīkla sinapšu svariem. Šīs atšķirības rodas vairošanas procesā, jo jaunas paaudzes veidošana iekļauj genotipu krustošanu un nelielu mutāciju varbūtību. Šīs īpašības tiek mantotas nākošajās paaudzēs, garantējot labvēlīgo īpašību saglabāšanos. Selekcija nodrošina to, ka tikai vislabāk pielāgotie indivīdi veido nākošo paaudzi. (Floreano & Mattiussi, 2008)

Mākslīgas evolūcijas procesa izveidošana parasti ietver šādus posmus (Floreano & Mattiussi, 2008):

1. Ģenētiskās informācijas jeb genotipa attēlošanas veida izvēle.
2. Sākuma populācijas izveide.
3. Piemērotības, jeb novērtēšanas, funkcijas izvēle.
4. Selekcijas mehānisma izveide.
5. Krustošanas mehānisma izveide.
6. Mutācijas mehānisma izveide.

7. Datu analīzes procedūras izveide.

Turamākajās apakšnodaļās sīkāk aprakstīti svarīgākie šī saraksta elementi.

3.2.1. Genotipa attēlošanas veida izvēle un sākotnējās populācijas izveide

Genotipa attēlošana ir kāda atrisinājuma apraksts simbolu rindas, grafa vai kādā citā veidā. Attēlojumam ir jābūt izvēlētam ievērojot sekojošas prasības (Yoon & Kim, 2013):

1. Ģenētiskajam algoritmam jāspēj veikt rekombinācijas un mutācijas operācijas.
2. Ģenētiskā algoritma darbības rezultātā ar pēc iespējas lielāku varbūtību jāveidojas labākiem indivīdiem.
3. Visām iespējamajām genotipa kombinācijām ir pēc iespējas pilnīgāk jāpārklāj visa atrisinājumu telpa.

Ģenētiskās informācijas glabāšanai un attēlošanai praksē tiek izmantotas trīs pieejas (Floreano & Mattiussi, 2008):

1. Attēlošana ar diskrētām vērtībām – informācija tiek attēlota izmantojot kādu alfabētu ar noteiktu simbolu skaitu. Pie tam, katrs simbols var reprezentēt gan atsevišķu vērtību, gan daļu no simbolu rindas. Piemēram, binārā alfabēta simbolu rinda 100101 var tikt izmantota gan kā skaitļu 4 un 5 binārā interpretācija, gan kā loģisko ieeju signālu vērtības – uz pirmo, ceturto un sesto ieeju tiek padotas patiesas vērtības.
2. Attēlošana ar daļskaitli – genotips sastāv no noteikta skaita reālo skaitļu. Šāda pieeja ir piemērota gadījumos, kad ir nepieciešama precīzu parametru optimizēšana.
3. Attēlošana ar kokiem – tiek pielietota, kad genotipu var attēlot ar hierarhisku struktūru. Koks sastāv no galīga skaita iekšējo virsotņu ar funkcijām un galīga skaita ārējo virsotņu ar vērtībām. Izmantoto funkciju klāstam jābūt pietiekošam, lai programma varētu sasniegt iecerēto mērķi.

Sākuma populācija jābūt pietiekami lielai, lai nodrošinātu to, ka indivīdi uzrādīs dažādus rezultātus, jo pretējā gadījumā selekcijas mehānisms nevarēs efektīvi pildīt savu uzdevumu. Tomēr jāpatur prātā, ka palielinot populācijas izmēru, palielinās arī skaitļošanas ilgums (Banzhaf, 2013). Genotipu sākotnējās vērtības parasti tiek piešķirtas pēc nejaušības principa. Decimāldaļskaitļu vērtību gadījumā, lai paātrinātu evolūcijas procesu, sākuma vērtībām var tikt piemērotas noteiktas robežas. Koku struktūras tiek veidotas pielietojot

rekursīvas funkcijas, kas veido sākuma populācijas kokus, izmantojot nejaušu skaitļu ģeneratoru. Populāciju var izveidot arī no nelielai mutācijai pakļautiem daudzsološiem indivīdiem no iepriekšējiem eksperimentiem, tomēr tādā veidā pastāv risks iegūt nepietiekamu dažādību, kas var novest pie lokālā atrisinājuma atrašanas (Floreano & Mattiussi, 2008).

Piemērotības funkcija nepieciešama katra indivīda piemērotības novērtēšanai, tās rezultāts ir skaitliska vērtība. Funkcijas vērtība netieši ļauj novērtēt kopējo evolūcijas gaitas kvalitāti. Piemērotības funkcijas izveidei nav striktu noteikumu, tomēr par pamatu parasti tiek ņemta svērtā summa no dažādu mērķu izpildīšanās novērtējuma (Nelson, Barlow u.c., 2009). Piemēram, ja tiek veidots mobils robots šķēršļu apbraukšanai, piemērotības funkciju var sastādīt no robota vidējā kustības ātruma, laika kurā robots sasniedz norādīto mērķi un nepieciešamo sensoru skaita apgrieztās vērtības. Summas locekļus un to svarus nosaka heiristiski, pielietojot zināšanas par risināmo problēmu. Veidojot piemērotības funkciju, jāņem vērā, ka indivīda novērtēšana ir laikietilpīgs process un no novērtēšanas kvalitātes ir atkarīga evolūcijas efektivitāte (Floreano & Mattiussi, 2008).

Pētījumos pielietotās piemērotības funkcijas var iedalīt septiņās kategorijās (Nelson, Barlow u.c., 2009):

1. Apmācības datu piemērotības funkcijas (angļu val. *training data fitness function*) – šī funkciju klase tiek izmantota apmācības ar skolotāju gadījumā, kad ir pietiekami liels zināmo ieejas-izejas datu daudzums. Funkcijas vērtība ir apgriezti proporcionāla sistēmas izejas vērtības kļūdai (Nelson, Barlow u.c., 2009).
2. Uzvedības piemērotības funkcijas (angļu val. *behavioral fitness function*) – specifiskiem uzdevumiem veidotas funkcijas, kas novērtē ko un kā dara sistēma. Parasti tās iekļauj dažādas vairākas apakš funkcijas, kuras vērtē vienkāršāko notikums-reakcija sakarību atbilstību uzstādītajiem mērķiem. Piemēram, ja ir nepieciešams izveidot robotu, kas izvairās no šķēršļiem, tad derīguma funkcija var palielināties, kad robots, atbildot uz attāluma sensoru datiem, aktivizē attiecīgo motoru. Tomēr jāņem vērā, ka tādā gadījumā netiek vērtētas robotā spējas apiet šķēršļus, bet gan tā reakcija uz sensoru aktivizēšanos, tātad sekmīga uzstādīta uzdevuma izpilde nav garantēta. Šādas

derīguma funkcijas izveidei nav nepieciešams daudz iepriekš sagatavoto datu (Nelson, Barlow u.c., 2009).

3. Summas piemērotības funkcija (angļu val. *aggregate fitness function*) – novērtē cik lielā mērā ir izpildīts uzdevums, nepievēršot uzmanību kādā veidā uzdevums tika pildīts. Piemēram, ja robota uzdevums ir pārvietot objektus noteiktā vietā, tad derīguma mērs būtu objektu skaits paredzētajā vietā vērtēšanas beigās. Šī pieeja tiek pamatoti kritizēta, kā neefektīga, jo pirmās paaudzes faktiski nekad nav spējīgas pat daļēji izpildīt uzdevumu. Šādu situāciju sauc par palaišanas problēmu (angļu val. *bootstrap problem*). Tomēr summas piemērotības funkcijas ir svarīgas sarežģītas uzvedības veidošanā (Nelson, Barlow u.c., 2009).
4. Pielāgotās piemērotības funkcijas (angļu val. *tailored fitness function*) – apvieno uzvedības un summas piemērotības funkcijas, atrisinot abu pieeju problēmas. Atšķirībā no tīrām summas derīguma funkcijām, pielāgotā derīguma funkcija ir spējīga novērtēt daļēji izpildītu uzdevumu. Tomēr šādu funkciju izveidošanai ir nepieciešamas pietiekami dziļas zināšanas par sagaidāmo sistēmas uzvedību, sistēmas apkārtējo vidi un mērķa sasniegšanas ceļiem. Bieži šāda veida funkcijas tiek veidotas mēģinājumu un kļūdu procesā un balstās uz veidotāja pieredzi un zināšanām (Nelson, Barlow u.c., 2009).
5. Funkcionāli papildināmas piemērotības funkcijas (angļu val. *functionally incremental fitness function*) – evolūcijas sākumā tiek veidota vienkāršākā uzvedība. Kad šāda uzvedība ir sasniegta pieņemamā līmenī, piemērotības funkcija tiek uzlabota, lai sasniegtu kādu sarežģītāku sistēmas uzvedību. Šāds process tiek daudzreiz atkārtots līdz tiek sasniegta prasītā sarežģītā uzvedība. Šāds pakāpenisks evolucionārs process ir nepieciešams īpaši sarežģītu uzdevumu gadījumā, kad iespējamība, ka visai sākotnējā populācijai nav nosakāmu spēju izpildīt uzstādīto uzdevumu (Nelson, Barlow u.c., 2009).
6. Vides papildināšanas piemērotības funkcijas (angļu val. *environmental incremental fitness function*) – evolūcijas procesā pakāpeniski tiek palielināta vides sarežģītība. Ar šādu pieeju ir iespējams iegūt efektīgu sistēmu, tomēr pētījumu un eksperimentu daudzums ir salīdzinoši mazs (Nelson, Barlow u.c., 2009).

7. Konkurējoša piemērotības izvēle (angļu val. *competative fitness selection*) – sistēmas ir vērtētas tādos apstākļos, ka atsevišķu populācijas locekļu darbības var ietekmēt citu vērtējumu. Piemēram, ja robotu uzdevums ir atrast labāko ceļu starp diviem punktiem un viens indivīds iztraucē cita kustību, abiem robotiem sasniedzot mērķi, otrais indivīds saņems labāku novērtējumu, jo tas spēja kompensēt traucējumus (Nelson, Barlow u.c., 2009).

3.2.2. Selekcija

Selekcijas galvenā funkcija ir nodrošināt, ka no visas populācijas tiks izvēlēti indivīdi ar lielāko piemērotības funkcijas vērtību. Attiecību starp izvēlēto indivīdu skaitu un kopējo populācijas lielumu sauc par selekcijas spiedienu. Veidojot selekcijas mehānismu ar augstu selekcijas spiedienu var panākt ļoti ātru piemērotības pieaugumu, tajā pašā laikā pastāv risks pazaudēt dažādību populācijā, kas bieži noved pie lokālā atrisinājuma atrašanas. Tāpēc ir svarīgi uzturēt balansu, starp selekcijas spiedienu un citiem faktoriem, piemēram, mutāciju līmeni (Floreano & Mattiussi, 2008).

Eksistē vairākas stratēģijas selekcijas veikšanai (Floreano & Mattiussi, 2008):

1. Proporcionāla selekcija – pieņemot, ka novērtējums ir tikai pozitīvs, varbūtība, ka indivīds tiks izvēlēts nākošās paaudzes izveidei ir vienāda ar šī indivīda novērtējuma attiecibu pret visu populācijas indivīdu novērtējumu summu. Tomēr šī pieeja nedarbojas divos gadījumos: ja visiem indivīdiem ir aptuveni vienāds novērtējums un ja vienam indivīdam novērtējums ir ievērojami lielāks par citiem. Pirmajā gadījumā selekcijai tiek zaudēta jēga, jo varbūtība tikt izvēlētam visiem indivīdiem ir praktiski vienāda, kas nenes labumu evolūcijas gaitā, tā ka labi indivīdi vienalga var tikt zaudēti. Otrajā gadījumā viss populācijas genotips ātri sāk līdzināties labākā indivīda genotipam, kas negatīvi ietekmē evolūcijas procesu.
2. Selekcija pēc ranga – visi indivīdi tiek kārtoti pēc piemērotības funkcijas vērtības, un noteikts skaits labāko tiek izvēlēti nākošās paaudzes veidošanai Tādā veidā tiek nodrošināts tas, ka tiek izvēlēti gan labākie indivīdi, gan indivīdi ar salīdzinoši zemu, bet tomēr augstāku par vidējo piemērotību.
3. „Turnīra" selekcija – veidojot katru jaunu pēcteci, no visas populācijas tiek izvēlēts neliels skaits indivīdu, kuri tiek novērtēti un labākie tiek izvēlēti

pēcteča veidošanai. Obligāts nosacījums – katra jauna indivīda veidošanai tiek izvēlēta cita indivīdu kopa.

Izšķir trīs populācijas paaudzes nomaiņas veidus (Floreano & Mattiussi, 2008):

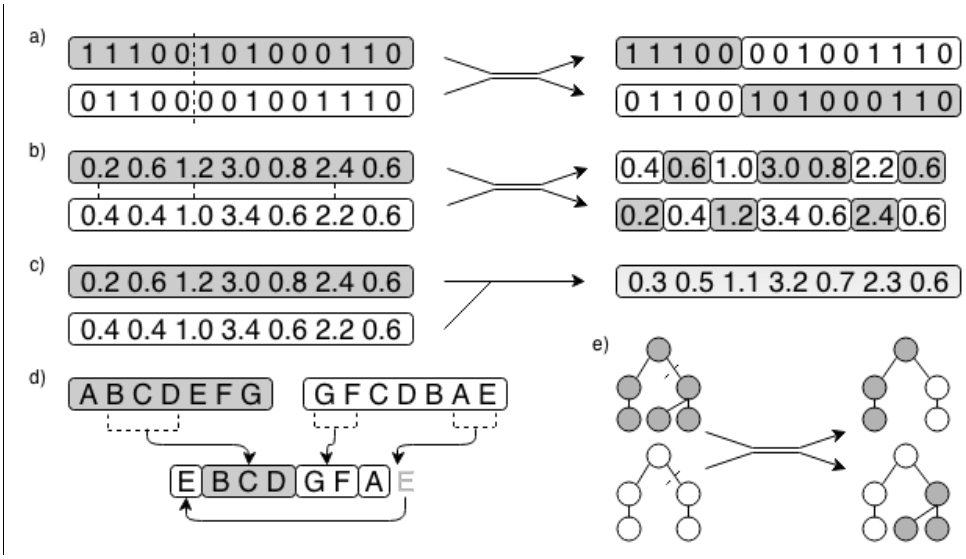
1. Pilnīga paaudzes nomaiņa – visi vecie indivīdi tiek izvākti no populācijas un aizvietoti ar jaunu paaudzi.
2. Elitisms– noteikts skaits labāko indivīdu no iepriekšējās paaudzes tiek pārcelts uz nākošās paaudzes populāciju bez izmaiņām.
3. Pakāpeniskā paaudzes nomaiņa – aizvietots tiek tikai noteikts skaits indivīdu ar sliktāko novērtējumu.

3.2.3. Jaunas populācijas veidošana

Evolucionārā procesa svarīgākie rīki ir ģenētiskie operatori, tādi kā krustošana un mutācija. Tie uztur populācijas dažādību un paver iespēju jaunu risinājumu atklāšanai, saglabājot iepriekš iegūtās derīgās pazīmes (Floreano & Mattiussi, 2008).

Krustošana ir svarīgākais process jaunas paaudzes veidošanā, tajā no divu vai vairāku esošās paaudzes indivīdu genotipiem tiek kombinēts jaunā indivīda ģenētiskais materiāls. Galvenais krustošanas iemesls ir doma, ka, daļēji apvienojot divu vecāku atrastos risinājums, pastāv iespēja iegūt vēl labāku risinājumu (Rivero, Dorado u.c., 2010). Tomēr jāņem vērā, ka krustošana ne vienmēr uzlabo jauno indivīdu rezultātus, jo procesa izpildes gaitā netiek ņemts vērā, kādi tieši genotipa apgabali ir atbildīgi par iegūto rezultātu, tādā veidā var tikt iegūts arī ievērojami sliktāks rezultāts (Floreano & Mattiussi, 2008).

Eksistē dažādas krustošanas procesa realizācijas dažādiem genotipa attēlošanas veidiem, bet kopējā ideja vienmēr ir ļoti līdzīga. Populārākās krustošanas pieejas ir ilustrētas 3.1. att.



3.1. att. Krustošanas pieeju piemēri: a) viena punkta, b) vienveidīga, c) aritmētiskā, d) simbolu virknēm, e) kokiem (aizgūts no (Floreano & Mattiussi, 2008))

a) Viena punkta krustošana – tiek pielietota diskrētu vai nepārtrauktu vērtību rindām. Tās veikšanai tiek nejauši izvēlēts viens vai vairāki punkti, kuros abi genomi tiek sadalīti un savstarpēji apmainīti.

b) Vienveidīga krustošana – pārsvarā pielietota nepārtrauktu skaitļu virknēm. Nejaušs skaits vērtību tiek apmainīts vietām.

c) Aritmētiskā krustošana – piemērota nepārtrauktu skaitļu virknēm. Jauna genotipa veidošanai izskaitļo vidējo aritmētisko, vai kādu citu vērtību, balstoties uz abu vecāku genotipiem.

d) Genotipiem, kas attēloti virknē, var tikt piemēroti daudzi citi nosacījumi, piemēram, virknei obligāti jāsatur visi simboli tieši vienu reizi. Tādos gadījumos var tikt izvēlēts nejaušs apgabals no viena genotipa, kuram klāt piekārto atlikušos simbolus secībā, kādā tie parādās otrajā genotipā.

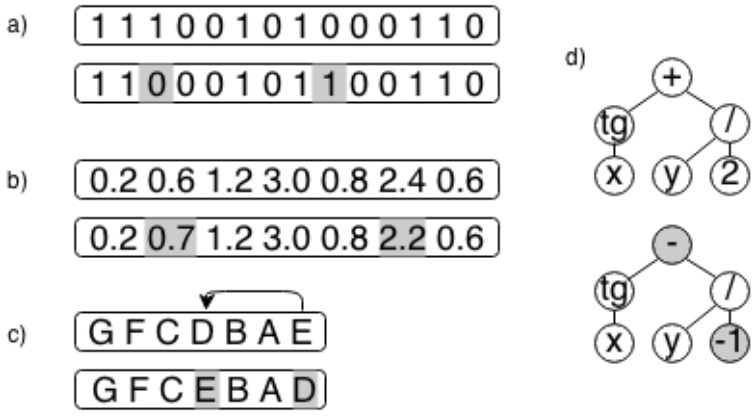
e) Genotipiem, kas tiek attēloti koku viedā, parasti tiek nejauši izvēlēti zari, kuri tiek savā starpā apmainīti.

Pēc krustošanas jaunie indivīdi tiek pakļauti mutācijai, kas ir nelielas nejaušas izmaiņas indivīdu genotipā ar nolūku ieviest lielāku dažādību un potenciāli atrast jaunus risinājumus. Mutācijas ir īpaši noderīgas, lai izvairītos no lokālā atrisinājuma atrašanas, kad populācija dominē ļoti līdzīgi genotipi. Tomēr mutāciju līmenim jāpaliek pietiekami zemam, lai netiktu zaudētas iepriekš atrastās derīgās īpašības (Floreano & Mattiussi, 2008).

Tipiski mutāciju līmeni definē kā katras genotipa pozīcijas izmaiņas varbūtību. Literatūrā piemin varbūtības ar pakāpi 10^{-2}, kas ir ievērojami augstāk par dabā sastopamo,

tomēr piemērotāko vērtību ir jāizvēlas atkarībā no mutācijas ietekmes uz kopējo populācijas piemērotības funkcijas vērtību (Floreano & Mattiussi, 2008).

Visbiežāk lietotie mutācijas veidi dažādiem genotipa attēlošanas variantiem ir ilustrēti 3.2. att.



3.2. att. Mutāciju piemēri dažādiem genotipa attēlojuma veidiem: a) binārais, b) decimāldaļskaitļi, c) simbolu virkne, d) koki (aizgūts no (Floreano & Mattiussi, 2008))

a) Attēlojums bināro skaitļu veidā – nejaušs skaits vērtību tiek apmainīts uz pretējo.

b) Decimāldaļskaitļu virknei - nejaušam skaitam vērtību tiek pieskaitīts skaitlis no Gausa sadalījuma N(0, σ) , kur 0 – vidējā vērtība un σ - variācija.

c) Simbolu virknei – nejaušs skaits simbolu pāru tiek apmainīts vietām.

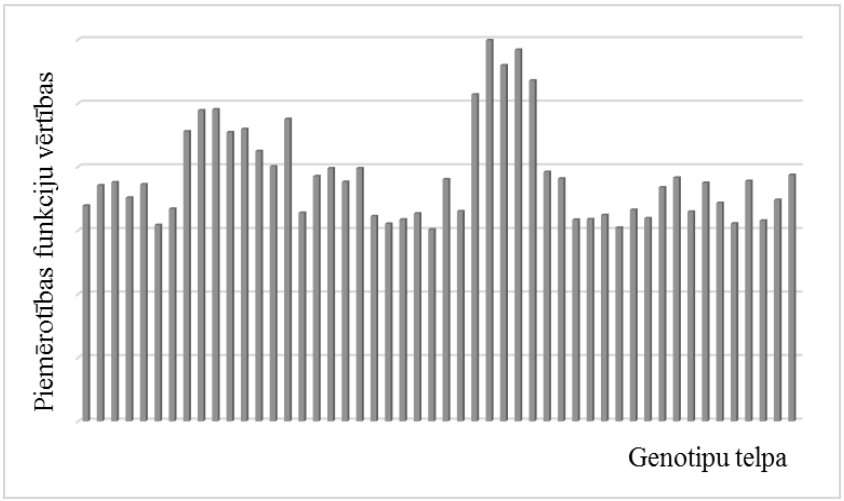
d) Attēlojumam koku veidā – nejauši izvēlētas virsotnes vērtība tiek aizvietota ar citu vērtību no tās pašas klases.

3.2.4. Evolūcijas rezultātu analīze

Pārskatu par pārmeklēšanas telpu var attēlot ar derīguma funkcijas ainavu, kur uz divu vai trīs dimensiju grafika horizontālajām asīm tiek atliktas visu iespējamo genotipu reprezentācijas, sakārtotas pēc noteikta kritērija, un uz vertikālās ass ir attiecīgo genotipu piemērotības funkciju vērtības (Karafotias, Haasdijk u.c., 2011). Piemērotības funkcijas ainavas piemērs ir parādīts 3.3. attēlā.

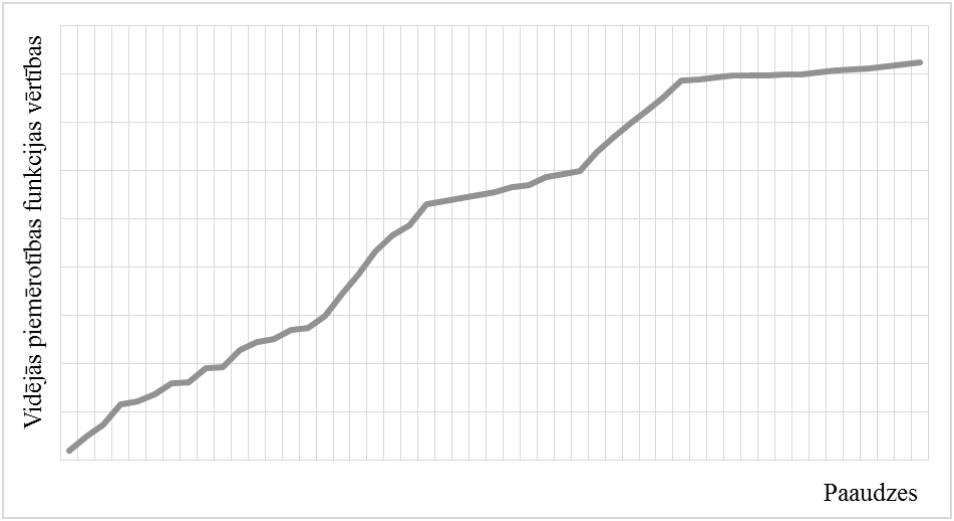
Ar līdzīgu grafiku var tikt novērtēta kāda konkrēta paaudze. Šajā gadījumā uz horizontālās ass tiek izvietoti visi šīs paaudzes indivīdi. Jāņem vērā, ka katras paaudzes piemērotības funkcijas ainava dod priekšstatu tikai par konkrētu paaudzi, tās nav tiešā veidā saistītas savā starpā (Floreano & Mattiussi, 2008). Krasas atšķirības starp piemērotības funkcijas vērtībām vienas paaudzes ietvaros norāda uz pietiekošu dažādību, kas ir īpaši svarīgi

evolūcijas procesa sākumā. Ar laiku šādas atšķirības samazinās, atrisinājumiem konverģējot uz kādu no maksimumiem. Vidējo starpību var izmantot evolūcijas procesa kontrolēšanai, apstādinot procesu, kad tiek sasniegts noteikta vidējās starpības vērtība (Floreano & Mattiussi, 2008)



3.3. att. Derīguma funkcijas ainavas piemērs

Evolūcijas rezultātus cauri visām paaudzēm var attēlot ar piemērotības grafiku. Parasti tajā secīgi attēlo katras paaudzes vidējo vai maksimālo piemērotību. Tāds attēlošanas veids ļauj novērtēt kopējo evolūcijas efektivitāti. (Floreano & Mattiussi, 2008). Piemērotības grafika piemērs ir paradīts 3.4. att.



3.4. att. Piemērotības grafika piemērs

Piemērotības grafika pieaugums liecina par sekmīgu evolūcijas gaitu, "līdzenumi" norāda uz nepietiekošu dažādību populācija, kam par iemeslu var būt kāda lokāla atrisinājuma sasniegšana, pazeminājumi norāda uz labvēlīgo iezīmju zaudēšanu. Ja grafikā nav redzamas

augšanas tendences, tad evolūcijas process neatšķiras no nejaušu atrisinājumu pārbaudes un ir vērts pārskatīt izvēlētos procesa parametrus (Floreano & Mattiussi, 2008).

3.3. Neiroevolūcija

Tā kā darba uzdevums ir izpētīt neironu tīklu apmācību ar evolucionārās skaitļošanas metodēm, ir vērts atsevišķi apskatīt neiroevolūciju, kas ir ģenētiskā algoritma vai ģenētiskās programmēšanas pielietojums neironu tīklu iegūšanai (Brownlee, 2011).

Neironu tīkla veidošanas process sastāv no diviem svarīgiem posmiem – neironu tīkla uzbūves izvēles un neironu tīkla noskaņošanas. Tīkla uzbūve jeb arhitektūra ir lielā mērā atkarīga no risināmā uzdevuma, tāpēc tīkla veidotājam jāmāk pareizi novērtēt un iespējams eksperimentāli pārbaudīt dažādas arhitektūras un jāveic to apmācība. Šis process ir laikietilpīgs un darbietilpīgs. Šī procesa automatizēšanai pielieto evolucionārās skaitļošanas metodes, kas ļauj automatizēti pildīt vienu vai abus neirona tīkla veidošanas soļus. Evolucionāro skaitļošanu var pielietot arī gadījumos, kad nav pietiekoši daudz zināšanu par arhitektūru piemērotību konkrētu uzdevumu risināšanai. (Rivero, Dorado u.c., 2010).

Eksistē trīs pieejas neironu tīklu iegūšanai ar evolucionārās skaitļošanas palīdzību (Rivero, Dorado u.c., 2010; Miikkulainen, 2011):

1. Pielāgojot svarus – tīkla topoloģija ir definēta iepriekš un tiek mainīti tikai savienojumu svari, kas parasti tiek iekodēti bināro vai daļskaitļu rindas veidā. Šī pieeja der gadījumiem, kad ir aptuveni zināmas topoloģijas priekšrocības konkrēta uzdevuma risināšanai (Rivero, Dorado u.c., 2010).
2. Pielāgojot arhitektūru – tīkla topoloģijas veidošana un svaru pielāgošana. Tīkla topoloģijas veidošanai pielieto vienu no diviem algoritmiem – konstruktīvo vai destruktīvo. Konstruktīvais algoritms sāk ar mazāko iespējamo tīklu un pakāpeniski veido jaunus savienojumus iegūstot sarežģītāku uzvedību. Destruktīvais algoritms sāk darbojies tieši pretēji, atbrīvojoties no nevajadzīgās uzvedības līdz paliek tikai nepieciešamais. Tīkla savienojumi tiek attēloti matricas veidā, kur katras šūnas vērtība nosaka savienojuma esamību un svaru. Cita tīkla struktūras attēlošanas veidi iekļauj attēlošanu ar kokiem un parametrisko attēlošanu (Rivero, Dorado u.c., 2010).
3. Pielāgojot apmācības likumus – ar evolucionāro procesu tiek iegūti citu apmācības algoritmu parametri (Rivero, Dorado u.c., 2010).

Neiroevolūcija ir stimulētās apmācības veids un tāpēc, tā ir labi piemērota robotu vadības sistēmu automatizētai veidošanai.

3.4. Pielietojums robotikā

Evolucionāro algoritmu pielietojumu robotikā dēvē par evolucionāro robotiku, kas ir salīdzinoši jauns un strauji augošs pētījumu virziens. Evolucionārās robotikas galvenais mērķis ir izveidot metodes robotu vadības sistēmu automatizētai iegūšanai (Wang, Tan u.c., 2006).

Lielākā daļa eksistējošo pētījumu veltīti vienkāršas uzvedības veidošanai, kas nav viegls uzdevums (Nelson, Barlow u.c., 2009):

1. Pārvietošanās un izvairīšanās no šķēršļiem – visbiežāk pētītā un veidotā uzvedība. Robotam ir jāspēj sekmīgi pārvietoties stacionāro un kustīgo šķēršļu klātbūtnē.
2. Gaitas apgūšana – svarīgs uzdevums soļojoša robota izveidē. Pareizas gaitas izveidošana ir sarežģīts un laikietilpīgs uzdevums, kas prasa vairāku izpildmehānismu koordinētu darbināšanu.
3. Meklēšana – robots pārmeklē apkārtējo vidi ar mērķi atrast uzdoto objektu vai vietu.

Eksistē arī mēģinājumi izveidot salīdzinoši sarežģītāku uzvedību, nekā iepriekš minētie piemēri. Šādu vadības sistēmu veidošana prasa ievērojami sarežģītākas piemērotības funkcijas izvedumu un aprēķinu (Nelson & Grant, 2006). Sarežģītas uzvedības piemēri (Nelson, Barlow u.c., 2009):

1. Meklēšana un pārvietošana – robotam ne tikai jāatrod prasītais objekts, bet arī jāpārvieto tas noveiktā vietā. Šis ir viens no sarežģītākajiem uzdevumiem, jo ietver sevī vairākus secīgus procesus, kurus robotam ir precīzi jāizpilda – sameklēt, satvert, pārvietoties uz mērķa stāvokli un izkraut.
2. Mednieks un medījums –robots „mednieks" iemācās tvert citus robots – „medījumus", kuri, savukārt, iemācās izvairīties no mednieka. Bieži vienu no lomām pilda robots ar nemainīgu manuāli izveidotu programmu, kamēr pārējie tiek apmācīti automatizēti.
3. Vairāku kontrolpunktu izbraukšana noteiktā secībā.
4. Robotu grupas koordinēta kustība – spieta robotika.

Eksperimentos izmantotie roboti lielākoties ir maza izmēra (5-20cm diametrā) ar nesaistītiem motoriem un aprīkoti ar infrasarkanajiem attāluma sensoriem, fotoelementiem un taustes sensoriem. Vairumā gadījumu tie tiek veidoti speciāli katram pētījumam. Tomēr eksistē arī gatavi risinājumi, kas ir populāri pētnieku starpā, piemēram Khepera robots. Tam ir modulāra uzbūve, kas ļauj viegli veidot nepieciešamo konfigurāciju. Robota diametrs ir tikai 5 cm un tam ir ierobežoti skaitļošanas resursi (Nelson, Barlow u.c., 2009). Cits populārs risinājums ir Koala robots, tas, līdzīgi kā Khapera robots, ir modulārs, bet ievērojami lielāks izmēros (30 cm diametrā) un jaudīgāks skaitļošanas resursu ziņā. Vairākos eksperimentos izmantoti uz LEGO Mindstorm bāzes veidoti roboti (Nelson, Barlow u.c., 2009).

Evolucionārajā robotikā veiktais darbs lielākoties ir pētniecisks un eksperimentāls. Šī darba rezultāti pierāda, ka autonomu robotu vadības sistēmas noteiktu uzdevumu izpildei var būt iegūtas ar evolucionārās skaitļošanas metodēm. Tomēr joprojām nav pierādīts, ka ar evolucionāro skaitļošanu var izveidot vadības sistēmu sarežģītu un vispārinātu uzdevumu izpildei, jo katrā literatūrā sastopamajā pētījumā robota darbības novērtēšanai tika izmantotas specifiskas derīguma funkcijas, kas pielāgotas konkrētā uzdevuma izpildei noteiktā vidē (Moubarak & Ben-Tzvi, 2012).

Pētījumi notiek arī citā evolucionārās skaitļošanas virzienā – spieta algoritmu pielietošanā reālos robotos. Šādi algoritmi atveido sociālo dzīvnieku uzvedību, piemēram, skudru koloniju vai bišu stropu (Tan & Zheng, 2013). Vairāku robotu kopīga darbība ļauj pildīt uzdevumus, ko viens robots nevar izpildīt vispār, kā arī nodrošina izturību pret robotu bojājumiem. Šāda uzdevuma piemērs ir nūju vilkšanas eksperiments (Auke JI, 2001), kad robotiem bija patstāvīgi jāiemācās pārvietot priekšmetus, kas ir pārāk lieli, lai tos pārvietotu viens robots, bet ir pārvietojami ar divu un vairāk robotu palīdzību (Tan & Zheng, 2013).

Uz šo brīdi pasaulē eksistē vairākas pētnieku grupas, kas veido spieta robotus reālu uzdevumu veikšanai (Tan & Zheng, 2013):

1. Project SI – pētnieku grupa no Šanhajas universitātes izstrādā spietu no modulāriem robotiem eMouse, kas var tikt pielietoti visdažādākajiem uzdevumiem.
2. iRobot Swarm project – Masačūsetsas Tehnoloģiju Institūta pētnieku grupa veido algoritmus spietam ar lielu robotu skaitu, kas varētu pildīt uzdevumus neskatoties uz atsevišķu indivīdu bojājumiem. Komanda izveidoja globālu novērošanas sistēmu ar automātisku uzlādes staciju.

3. Pheromone robotics project – veido spietu no maza izmēra robotiem izlūkošanas, novērošanas, bīstamu apstākļu noteikšanai un ceļa meklēšanai pamatojoties uz skudru komunikācijas principiem.

No visa iepriekšminētā var secināt, ka evolucionārā robotika un neiroevolūcija ir perspektīvi pētījumu virzieni. Tomēr, kā jebkurā jaunā nozarē, eksistē daudzi neizpētīti aspekti (Miikkulainen, 2011). Kā vienu no piemēriem var minēt neironu tīklu topoloģiju piemērotību robotu vadības sistēmās dažādu uzdevumu veikšanai. Literatūrā aprakstītajos pētījumos neironu tīklu topoloģijas izvēle balstījās uz heiristiskām un empīriskām metodēm (Nelson, Barlow u.c., 2009). Tāpēc, saskaņā ar šajā bakalaura darbā uzstādītajiem uzdevumiem, nākošajā nodaļā tiek praktiski analizēta automatizēta robota vadības mehānismu iegūšana pielietojot dažādas neironu tīklu topoloģijas, kuru svari tiek noskaņoti ar ģenētisko algoritmu.

4. ROBOTA VADĪBAS MEHĀNISMA AUTOMATIZĒTA IEGŪŠANA

4.1. Vadības mehānisma iegūšanas plāns

Viens no šī darba uzdevumiem ir izveidot un veikt eksperimentus ar automatizēti iegūtu autonoma robota imitācijas modeļa vadības sistēmu. Lai novērtētu vadības mehānisma efektivitāti, tam tiek uzstādīts uzdevums – veiksmīgi izbraukt cauri koridoram ar šķēršļiem.

Saskaņā ar darba uzdevumiem, jāveido neironu tīklos sakņots vadības mehānisms. Tā kā iepriekš nav zināma šim uzdevumam vislabāk piemērotā neironu tīklu topoloģija, tiek eksperimentāli pārbaudīti vairāki iespējamie varianti (Miikkulainen, 2011; Ruan & Dai, 2012):

1. Perceptroni ar vienu, diviem un trīs slēptajiem slāņiem.
2. Elmana tīkls.
3. Hopfilda tīkls.

Vadības mehānisma iegūšanai izstrādāta specializēta programmatūra, kas realizē neironu tīklu svaru noskaņošanu ar ģenētiskā algoritma palīdzību. Robotu vadības sistēmas novērtēšanā ir iespējama tikai kādā vidē – reālā vai imitācijas modelī. Reāla robota būvēšana ļauj vēl vadības sistēmas veidošanas procesā iekļaut visas tehnikas nepilnības, signālu trokšņus un citus faktorus. Tomēr tas uzliek papildus ierobežojumus – robotu būve ir dārgs un ilgs process, vadības sistēmas izveide var notikt tikai reālā laikā, konstruktīvo nepilnību novēršana rada lielas papildus izmaksas. Imitācijas vides izmantošana ļauj paātrināt robota vadības sistēmas veidošanu un ļauj izvairīties no papildus tēriņiem robota pārbūvei, tomēr imitācijas modelis nebūs pilnībā piemērots izmantošanai reālā vidē un prasīs papildus pielāgošanu (Matveev, Wang u.c., 2012). Pamatojoties uz šī bakalaura darba uzdevumiem, tiek izmantota imitācijas modeļa pieeja.

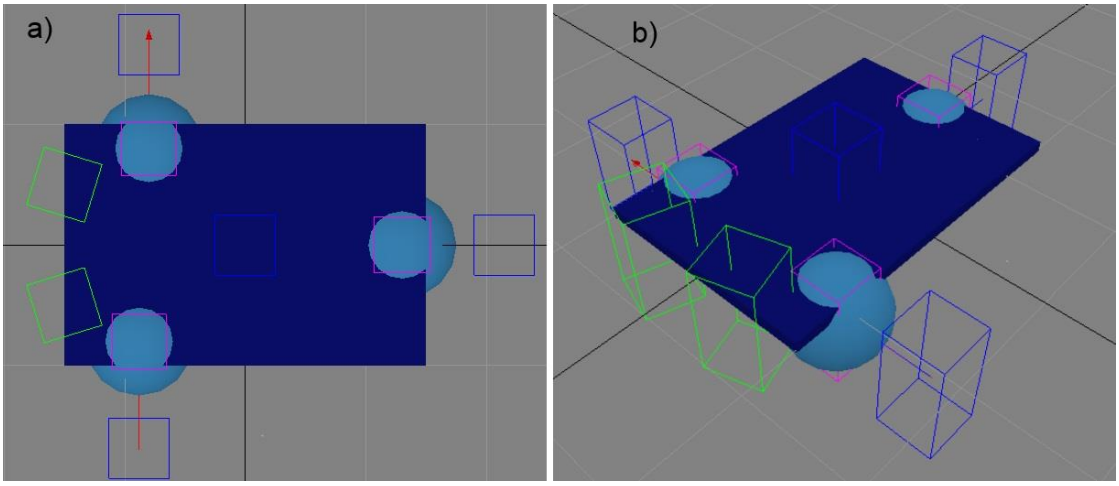
Visu apskatītu tīkla topoloģiju noskaņošana notiek izmantojot vienādus ģenētiskā algoritma parametrus. Rezultātā iegūtie piemērotības grafiki parāda ģenētiskā algoritma efektivitāti katras tīkla topoloģijas gadījumā. Uz šo grafiku pamata var secināt par apskatīto topoloģiju efektivitāti uzstādītā uzdevuma izpildei.

4.2. Imitācijas modelis

Imitācijas modeļa izveidei pielietots AnyKode Marilou trīsdimensiju grafikas redaktors ar fizikas likumu imitācijas iespējām, kas ir izstrādāts speciāli robotikas projektiem. Tajā ir iespēja imitēt eksistējošu sensoru un izpildmehānismu darbību, fizisko ķermeņu mijiedarbību, apgaismojumu. Eksistē bibliotēkas, kas ļauj pieslēgties vides imitācijai un realizēt robotu vadību dažādās programmēšanās valodās – C/C++/C#, VB#, J#, Matlab u.c. Sazināšanās ar imitācijas vidi notiek pielietojot datortīkla savienojumu. Sazināšanās ar imitācijas vidi ļauj izmantot vienu kodu gan imitācijas modeļa, gan reāla robota vadībai (anyKode, 2013).

Eksperimentu veikšanai izveidots robota modelis (4.1. att.s) ar diviem savā starpā nesaistītiem motoriem, diviem uz priekšu virzītiem sensoriem un aizmugurējo riteni konstrukcijas stabilitātes uzlabošanai. Dažādas krāsas paralēlskaldņi apzīmē sekojošas lietas:

- Zilie – ģeometriski ķermeņi ar masu un savienojumiem savā starpā.
- Zaļie – sensori, kas imitē infrasarkanos attāluma sensorus ar attālumu diapazonu no 1 līdz 100 cm un trokšņu līmeni 10%.
- Rozā – motori ar iespēju griezties abos virzienos neatkarīgi viens no otra.



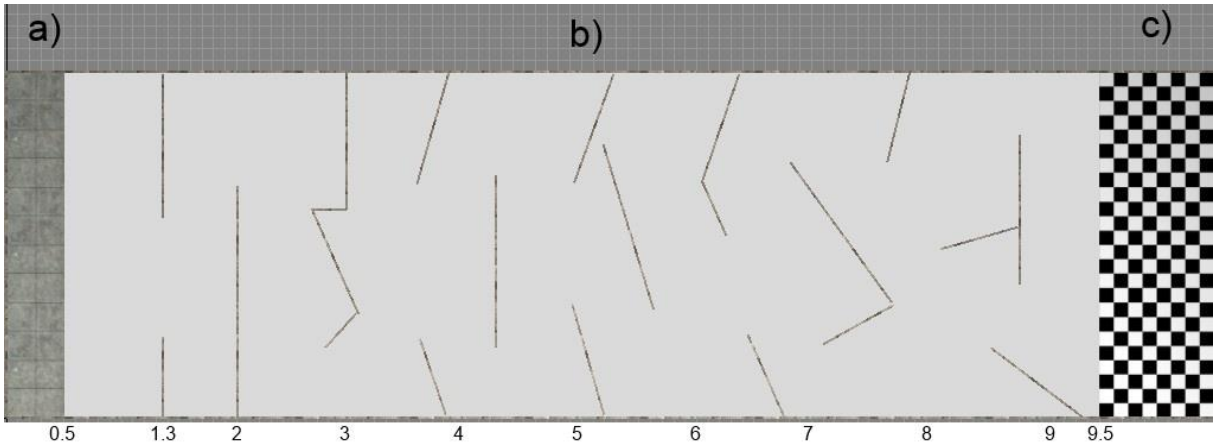
4.1. att. Izmantotā robota imitācijas modelis: a) augšskats, b) izometriskais skats.

Imitētā robota vide ir trijās zonās sadalīts koridors (4.2. att.):

a) Starta zona – katra izmēģinājuma sākumā robots tiek novietots nejaušā pozīcijā šī apgabala robežās. Zonas garums – 0,5m.

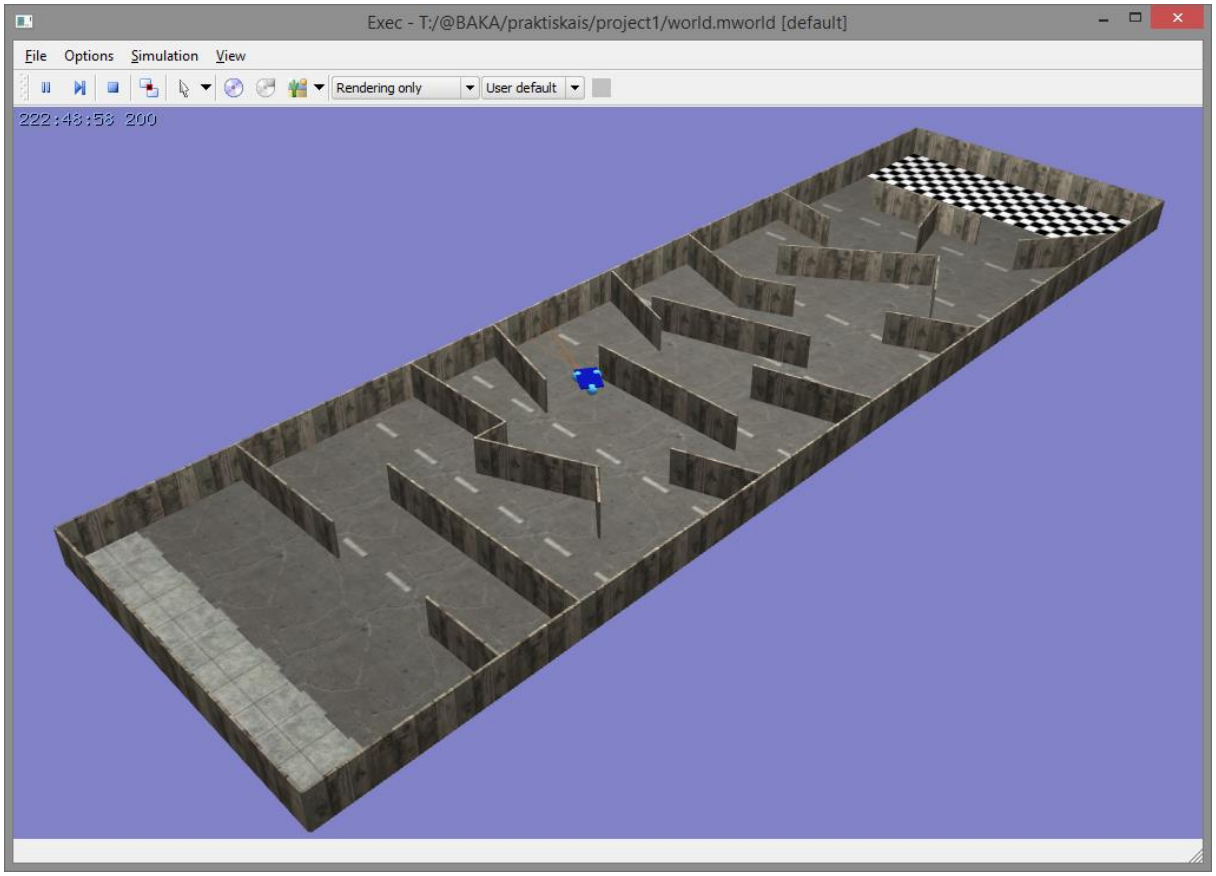
b) Labirinta zona – apgabals ar šķēršļiem, to starp vairāki taisni un šauri stūri, kas rada lielākās problēmas. Zonas garums – 9m.

c) Mērķa zona – apgabals kurā robotam ir jānokļūst. Zonas garums – 1m.



4.2. att. Imitācijas vide: a) starta zona, b) labirinta zona, c) mērķa zona.

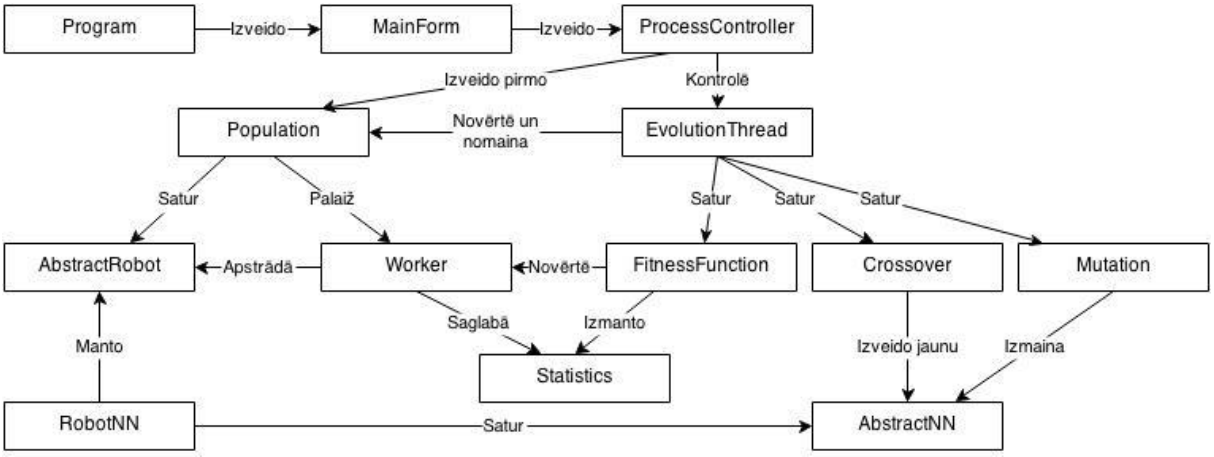
Kopējais labirinta garums ir 10,5 m, platums 3m. Iekšējās sienas ir izvietotas tādā veidā, lai labirinta iziešanai būtu nepieciešama sarežģīta uzvedība. Pirmajos posmos sienas veido tikai taisnus leņķus, vēlāk parādās šauri leņķi, kuru izbraukšana prasa atkāpšanos. Imitācijas modeļa kopējais izskats eksperimenta laikā parādīts 4.3. att..



4.3. att. Imitācijas modelis darbība

4.3. Programmatūra automatizētai vadības mehānisma iegūšanai

Robota vadības sistēmas iegūšanai izveidota specializēta programmatūra, kura realizē neironu tīklu svaru noskaņošanu ar ģenētisko algoritmu. Programmatūras struktūra semantiskā tīkla veidā paradīta 4.4. att.. Shēmā parādītas galvenās programmas klases un to saistība savā starpā. Vienkāršības labad nav parādītas visas atvasinātās klases, šablona klases un palīgklases.



4.4. att. Lietotnes struktūra

Lietotnes ieejas punkts ir klase `Program`, kas tiek izveidota palaišanas brīdī. Šī klase izveido grafisko saskarni, kurā lietotājam ir iespējams iestatīt ģenētiskā algoritma parametrus un palaist algoritma izpildi. Pēc algoritma palaišanas tiek izveidots `ProcessController` objekts, kurš ir atbildīgs par pirmās populācijas izveidi klasē `Population` un ģenētiskā algoritma izpildes palaišanu atsevišķā pavedienā – `EvolutionThread`. Pēc pavediena palaišanas tajā, izmantojot fabrikas šablonu tiek izveidoti ģenētisko operatoru un derīguma funkcijas objekti, kas vēlāk tiek nodoti populācijas klasei attiecīgās darbības veikšanai. `Population` klase satur noteiktu skaitu `AbstractRobot` klases pēcteča objektu, kurus secīgi pārbauda atsevišķā `Worker` pavedienā, novērtē to izmantojot izvēlēto `FitnessFunction` objektu un ar `Crossover` un `Mutation` objektu palīdzību izveido jaunu paaudzi. Šī eksperimenta veikšanai vienīgais `AbstractRobot` klases pēctecis ir `RobotNN` klase, kas implementē neironu tīklos balstītu robotu kontrolieri.

Pēc lietotāja izvēlētā paaudžu skaita pārbaudes, tiek veidota atskaite ar visu paaudžu indivīdu novērtējuma vērtībām. Katrs ieraksts atskaitē ir viena paaudze, lauku skaits ir vienāds ar izvēlēto paaudzes izmēru, ieskaitot eliti, ja tāda ir paredzēta procesa izpildē.

Atskaite tiek saglabāta .xlsx formātā lietotnes mapē, faila nosaukums satur procesa beigu datumu un laiku.

Programmatūra ir veidota tādā veidā, lai tajā būtu iespējams viegli izveidot vairākas savstarpēji aizvietojams algoritmu parametru realizācijas. Piemēram, dažādus krustošanas mehānismus. Tas ir iespējams pateicoties objektorientētās pieejas izmantošanai un statiskas fabrikas modelim. Katrs no procesa parametriem tiek realizēts mantojot attiecīgo abstrakto klasi, parametru objekti tiek veidoti programmas izpildes laikā izsaucot attiecīgās fabrikas `Create()` metodi.

Abstrakto klašu pēcteču veidošanas nosacījumi:

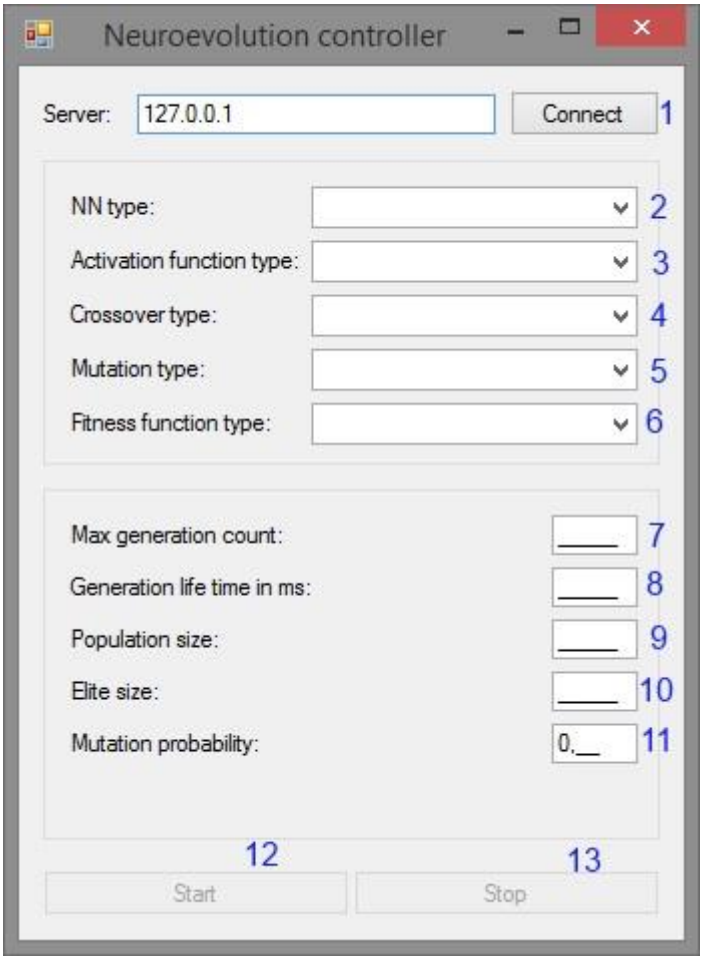
1. `AbstractRobot` – robota vadības sistēmas klase. Ir jārealizē metode `ComputeStep()` tādā veidā, lai tajā vismaz vienu reizi tiktu izsaukta metode `SetNewMotorSpeeds(float[] speeds)`, kas izmaina motoru griešanās ātrumu. Sensoru mērījumus var iegūt ar `GetSensorReadings()` metodi.
2. `AbstractNN` – abstrakts neironu tīkls. Klases konstruktorā parametros `NeuronCount` un `LinkCount` jāieraksta attiecīgi neironu skaitu un savienojumu skaitu. Konstante `HIDDEN_NEURON_COUNT` var tikt izmantota, lai norādītu cik neironu ir katrā slēptajā slānī. Papildus konstruktoram ir jārealizē metode `InitNetwork()`, kurā jāizveido neironi, pievienojot tos sarakstam `Neurons`, un sinapses, ar metodes `CreateSynapse()` palīdzību.
3. `ActivationFunction` – abstrakta neironu aktivizācijas funkcija. Ir jārealize tikai metode `Compute(value)`, kas aprēķina neirona vērtību pie nodotā argumenta.
4. `Crossover` – abstrakts krustošanas mehānisms. Jārealizē tikai `CreateNewPopulation(oldPopulation, mutation)` metode, kuras argumenti ir saraksts ar vecās paaudzes indivīdiem un mutācijas mehānisms un rezultāts ir saraksts ar jauniem indivīdiem.
5. `Mutation` – abstrakts mutācijas mehānisms. Jārealizē metode `Mutate(robot)`, kura veic izmaiņas nodotā robota ģenētiskajā kodā.
6. `FitnessFunction` – abstrakta piemērotības funkcija. Jārealizē metode `Calculate(statistics)`, kas balstoties uz robota darbības statistiku

novērtē tā uzvedību pārbaudes laikā. Statistika iekļauj sevī datus par robota motoru ātrumiem, sensora rādījumiem un pozīciju katrā aprēķinu ciklā.

Pēc šo nosacījumi izpildes ir jāpievieno attiecīgā izveidotās klases identifikatoru pareizajā grupā `Enum` failā, jāizveido nosacījums attiecīgajā fabrikas klasē un jāpievieno ieraksts grafiskās lietotnes attiecīgajā izvēlnē.

Lietotnes grafiskā saskarne (4.5. att.) satur šādus elementus:

1. Imitācijas vides adreses ievades lauks un pieslēgšanās poga.
2. Veidojamā neironu tīkla topoloģija.
3. Veidojamā tīkla neironu aktivizācijas funkcija.
4. Krustošanās mehānismā veids.
5. Mutācijas mehānisma veids.
6. Izmantotās derīguma funkcijas veids.
7. Paaudžu skaits eksperimentā.
8. Paaudzes indivīdu novērtēšanas laiks milisekundēs. Jāņem vērā, ka šeit tiek ierakstīts reālais laiks. Ja imitācijas vidē ir uzstādīts laika reizinātājs, imitētā robota dzīves laiks atšķirsies no reālā.
9. Vienas paaudzes izmērs, neieskaitot eliti, kas saglabājas no iepriekšējās paaudzes.
10. Elites izmērs.
11. Varbūtības pakāpe ar kādu notiek mutācijas.
12. Procesa uzsākšanas poga.
13. Procesa apstādināšanas poga.



4.5. att. Lietotnes grafiskā saskarne.

4.4. Ģenētiskā algoritma realizācija

Tā kā ģenētiskais algoritms ir pielietots neironu tīkla svaru noskaņošanai, tiek izmantots binārs genotipa attēlošanas veids. Katrs atsevišķs svars tiek pārvērsts 16 bitu rindā, tātad ir iespējamas 2^{16} vērtības lai attēlotu skaitļus no -1 līdz 1. Mazākā iespējamā svara vērtība ir $1{,}5 * 10^{-5}$.

Indivīdu novērtēšanai tika izmēģinātas literatūrā minētās piemērotības funkcijas, piemēram, formula (4.1), ar kuras palīdzību tika veiksmīgi iegūta sienas sekošanas uzvedība (Nelson, Barlow u.c., 2009). Tomēr citos eksperimentos izmantotās funkcijas ir pielāgotas konkrētiem apstākļiem un uzrāda sliktus rezultātus.

$$f = s_{ir} - (v_l + v_r - |v_l - v_r|), \tag{4.1}$$

kur s_{ir} – sensoru rādījumu summa,

v_l – kreisā motora ātrums,

v_r – labā motora ātrums.

Neeksistē noteikumi pareizas piemērotības funkcijas veidošanai, tāpēc tika pieņemts lēmums izmantot summas funkciju. Šāda funkcija novērtē tikai paveikto rezultātu, ignorējot rezultāta iegūšanas procesu. Šī eksperimenta ietvaros, tas nozīmē, ka par vērtējumu tiek uzskatīts maksimālais attālums, kuru ir nobraucis apskatītais indivīds. Uzstādītais uzdevums ir pietiekami vienkāršs, lai robots varētu daļēji to izpildīt, tātad palaišanas problēma veiktajos eksperimentos nav aktuāla.

Kā krustošanas mehānisms tiek izmantota vienveidīgā krustošana – katrs atsevišķs gēns ar vienādu varbūtību var nākt no jebkura no vecākiem. Šāda pieeja ir izdevīga, jo tā vienādi labi piemērota gan binārajam, gan skaitliskam genotipa attēlojumam. Krustošana nodrošina labvēlīgo iezīmju saglabāšanu, tajā pašā laikā ieviešot pietiekamu dažādību jaunu iezīmju veidošanai. Lai pilnībā nodrošināt labu iezīmju saglabāšanos tiek izmantots elitisms – daļa no populācijas ar visaugstākajiem vērtējumiem tiek saglabāta nemainīga. Elites izmēru nosaka lietotājs pirms eksperimenta uzsākšanas. Elitisma atslēgšanai pietiek uzstādīt elites izmēram nulles vērtību.

Mutācijai pielietota nejaušu bitu apgriešana – ar lietotāja ievadīto varbūtību jebkura bita vērtība var tikt mainīta uz pretējo. Ir jāatceras, ka šāds mutācijas mehānisms var tikt pielietots tikai bināram genotipa attēlojumam.

Neeksistē striktu vadlīniju ģenētiska algoritma parametru izvēlei katram noteiktam uzdevumam, tāpēc pirms eksperimentu uzsākšanas tika veikti vairāki izmēģinājumi, ar mērķi noteikt dažādu parametru ietekmi uz algoritma izpildes efektivitāti. Uz šo izmēģinājuma pamata izvēlēti šādi algoritma parametri:

1. Paaudžu skaits – 150. Visos veiktajos izmēģinājumos ievērojami rezultāti tika sasniegti jau pie simtās paaudzes, tāpēc izvēlētā vērtība ir pietiekama kopējās algoritma izpildes gaitas novērtēšanai.
2. Indivīdu skaits paaudzē – 35.
3. Vienas paaudzes dzīves reālais laiks – 2 sekundes, kas ir 20 sekundes simulētā laika. Pietiekams laiks, lai pat vidējs indivīds varētu pilnībā izbraukt labirintu.
4. Elites izmērs – 5, jeb 15% no visas populācijas. Saglabāto indivīdu skaits ir pietiekams, lai, pielietojot savstarpējo krustošanos, izveidot pilnu jaunu paaudzi. Tas nodrošina labvēlīgo iezīmju saglabāšanu vairāku secīgu paaudžu nomaiņas laikā, pat ja visi jaunie indivīdi izrādās ievērojami sliktāki par esošajiem.

5. Mutācijas varbūtībā – 10%. Izvēlētais skaitlis ir ievērojami lielāks, nekā literatūrā minētais (Wang, Tan u.c., 2006), lai kompensētu elites izmēru un saglabāt dažādību populācija.

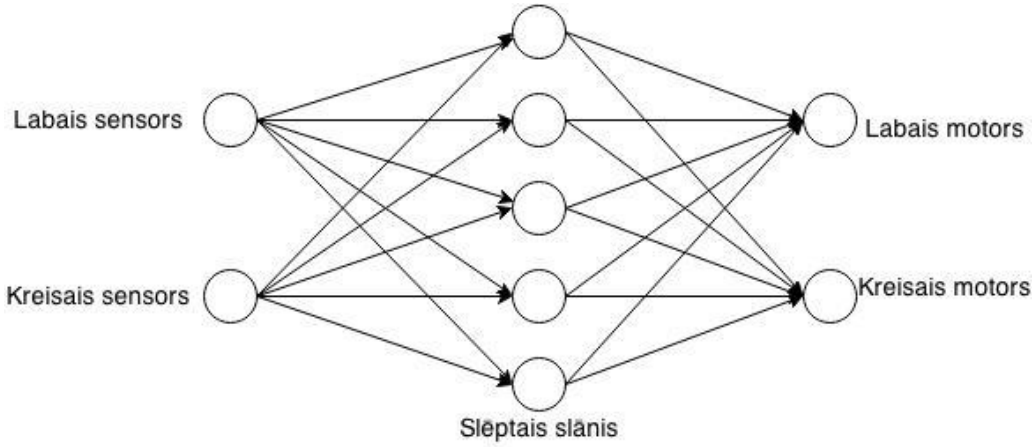
4.5. Neironu tīklu topoloģijas

Tīkliem ir 2 ieejas neironi un 2 izejas neironi. Ieejas saņem sensoru rādījumus tādā veidā, ka 1 atbilst sensoru maksimāli iespējamajam signālam un 0 minimālajam. Izejas neironi ir savienoti ar motoriem, neirona vērtība 1 apzīmē maksimālo ātrumu braucot uz priekšu, -1 – maksimālo ātrumu atpakaļgaitā un 0 – apstāšanos.

Visi tīklu neironi izmanto sigmoidālu aktivizācijas funkciju, tātad to vērtības ir intervālā [0,1]. Sinapšu svari var pieņemt vērtības intervālā [-1,1]. Neironu tīklu klasifikācija un topoloģiju īpašības ir sīkāk aprakstītas 2.3. nodaļā.

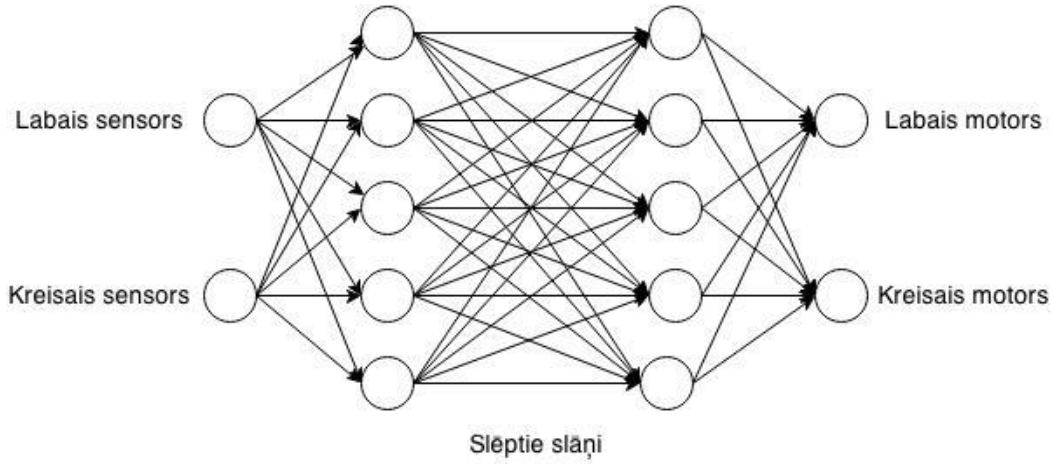
Eksperimentu veikšanai tika izvēlētas šādas neironu tīklu topoloģijas:

1. Perceptrons ar vienu slēpto slāni no pieciem neironiem slēptajā slānī. Neironu skaits slēptajā slānī ir izvēlēts empīriskā ceļā no veiktajiem izmēģinājumiem. Precīza neironu tīkla uzbūve parādīta 4.6. att.. Šis perceptrons izmantots par pamatu pārējām pētītajām tīklu topoloģijām.



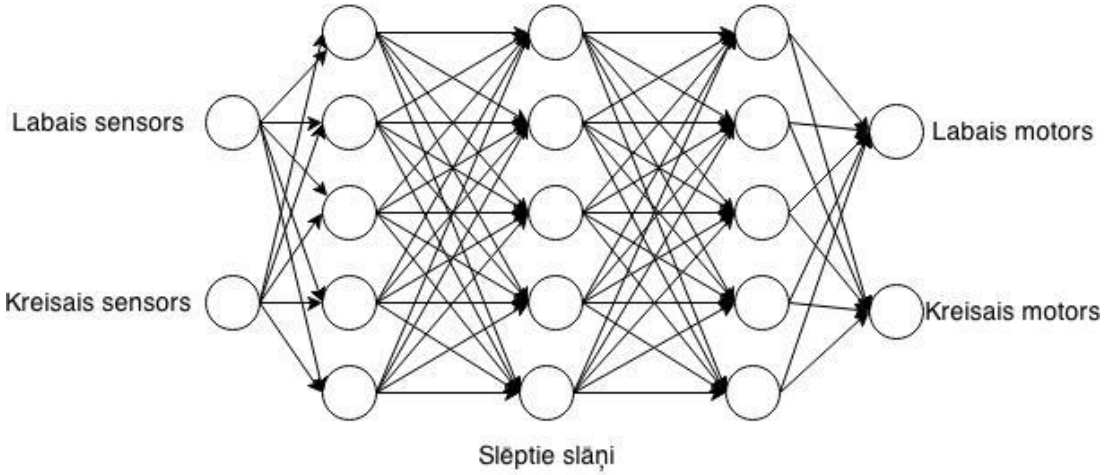
4.6. att. Pētītā viena slēptā slāņa perceptrona uzbūve.

2. Perceptrons ar diviem slēptajiem slāņiem no pieciem neironiem. Vairāki slāņi ļauj izveidot sarežģītāku, nelineāru uzvedību. Tajā pašā laikā ievērojami lielāks sinapšu skaits var apgrūtināt apmācības procesu. Precīza divu slāņu perceptrona uzbūve parādīta 4.7. att..



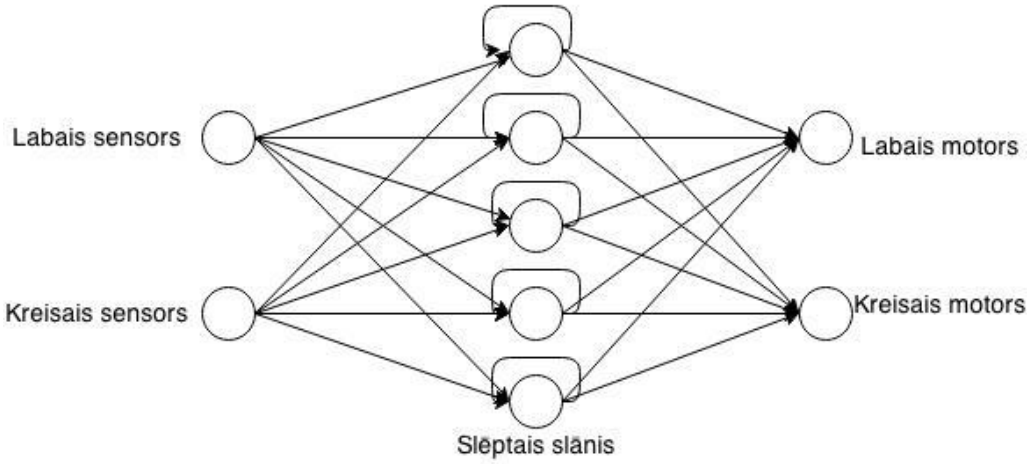
4.7. att. Pētītā divu slāņu perceptrona uzbūve.

3. Perceptrons ar trīs slēptajiem slāņiem no pieciem neironiem. Līdzīgi iepriekšējam neironu tīklam, vairāki slēptie slāņi dod iespēju veidot sarežģītāku uzvedību. Papildus slāņa pievienošana ļauj novērtēt slāņu skaita ietekmi uz ģenētiska procesa rezultātiem. Precīza izmantotā trīs slāņu perceptrona uzbūve parādīta 4.8. att.



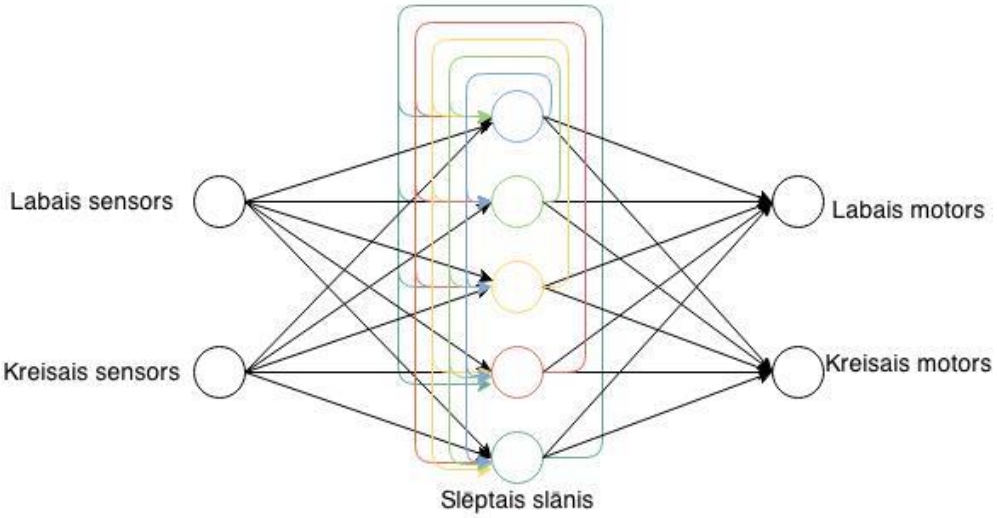
4.8. att. Pētītā trīs slāņu perceptrona uzbūve.

4. Elmana tīkls ar pieciem neironiem slēptajā slānī. Atgriezeniskās saites neironos ļauj tiem atcerēties savu iepriekšējo vērtību, veidojot atmiņu par iepriekšējo soli. Pateicoties iekšējai atmiņai, tīkli ar atgriezenisko saiti spējīgi apstrādāt signālu virknes. Tā kā tīkla aprēķins notiek secīgi, nav nepieciešams veidot atsevišķu slāni iepriekšējo vērtību glabāšanai, tā vietā neironu izejas tiek tiešā veidā savienotas ar savām ieejam. Precīza izmantotā Elmana tīkla uzbūve parādīta 4.9. att..



4.9. att. Pētītā Elmana tīkla uzbūve.

5. Hopfilda tīklam līdzīga struktūra ar pieciem neironiem slēptajā slānī. Atšķirībā no klasiskā Hopfilda tīkla, neironu vērtības nav tikai bināras. Šī iemesla dēļ ierobežojumi, kas parasti tiek uzstādīti Hopfilda tīkliem var netikt ievēroti, tomēr kopējā tīkla struktūra tiek saglabāta. Atgriezeniskās saites ļauj atcerēties visa tīkla vērtības iepriekšējā solī, veidojot sarežģītāku atmiņu. Precīza tīkla uzbūve parādīta 4.10. att.



4.10. att. Pētītā Hopfilda tīkla uzbūve.

4.6. Eksperimenti un to rezultātu analīze

Katra no nodaļā 4.5 aprakstītajām neironu tīklu topoloģijām tika izmantota robota vadības sistēmas izveidei, pielietojot ģenētisko algoritmu ar nodaļā 4.4. aprakstītajiem

parametriem. Katra izmēģinājuma rezultāti apkopoti piemērotības grafika veidā ar iezīmētu tendenci.

4.6.1. Perceptrons ar vienu slēpto slāni

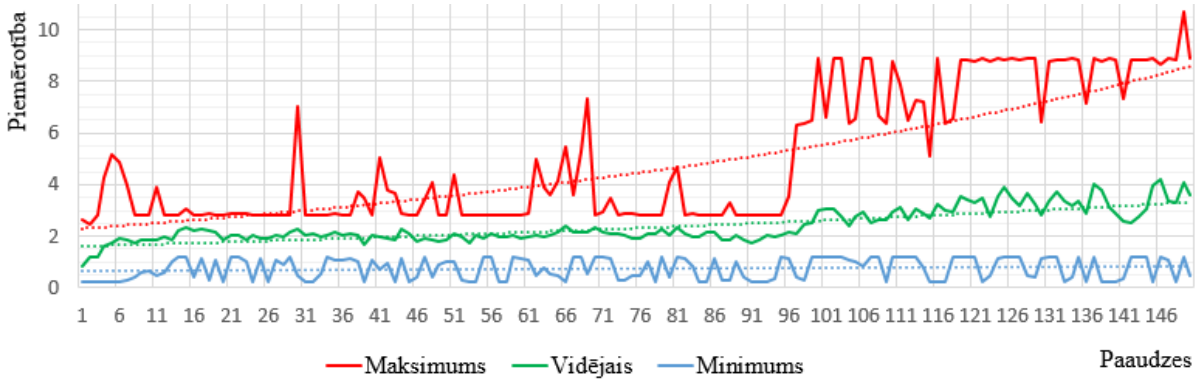
Vienslāņa perceptrona piemērotības grafiks ir parādīts 4.11. att. Process nenotika vienmērīgi un tajā var izdalīti trīs posmus:

- No 1. līdz 16. paaudzei notiek straujš vidējās vērtības kāpums, kas liecina par sekmīgu atlasi un jaunu iezīmju rašanos. Izveidojas vairāki indivīdi, kas spēj nonākt līdz labirinta vidum, tomēr maksimālās vērtības kāpums ir tikpat liels cik kritums. Šāds lēciens var tikt izskaidrots ar potenciāli laba indivīda veiksmīgu novietojumu starta zonā.
- No 17. līdz 98. paaudzei attīstība ir praktiski nemanāma, paaudzes vidējais novērtējums nenozīmīgi svārstās un pat samazina vērtību. Līdzīga tendence redzama arī pie maksimālajām vērtībām – gandrīz visas vērtības, ar retiem izņēmumiem, ir tuvu 3, kas ir tieši attālumā, kādā ir pirmais stūris, kura apbraukšanai nepieciešams atgriezties atpakaļ (4.2. att.). Retie lēcieni norāda drīzāk uz veiksmīgu pozicionēšanu sākuma zonā, nekā uz labvēlīgu iezīmju rašanos. Šo pieņēmumu atbalsta arī tas fakts, ka vidējā vērtība praktiski netiek ietekmēta, tātad nekāda labvēlīga iezīme netiek nodota nākošajām paaudzēm.
- No 99. paaudzes redzami vairāki secīgi maksimālās vērtības lēcieni, aiz kuriem, pēc svārstību perioda, iestājas pietiekami stabils augstas vērtības periods. Tas liecina par ārkārtīgi labvēlīgu iezīmju parādīšanos un nostiprināšanos populācijā. Vērtība, pie kuras stabilizējas maksimālās vērtības grafiks, atbilst pēdējam stūrim pirms finiša zonas (4.2. att.). Vienlaikus ar straujo maksimālās vērtības lēcienu, vidējās vērtības grafiks atsāk pakāpeniski pieaugt, kas nozīmē labvēlīgo gēnu izplatīšanos populācijā.

Tikai pirmspēdējās paaudzes pārstāvis spēja nokļūt finiša zonā. Ņemot vērā 97.-120. paaudzēs notikušo, var uzskatīt, ka pēc 20-50 paaudzēm varētu izveidoties indivīds, kas ar 100% varbūtību nonāktu finiša zonā.

Minimālā vērtība visa procesa laikā svārstījās robežās no 0 līdz 1,3, kas faktiski atbilst attālumam līdz pirmajai sienai (sk. 4.2. att.).

Eksperimenta rezultāti parāda, ka ar ģenētisko algoritmu ir iespējams iegūt pietiekami labu uz perceptrona ar vienu slēpto slāni balstītu vadības sistēmu. Tomēr process ir ielā mērā atkarīgs no nejaušiem notikumiem, kas nozīmē, veiksmīgs rezultāts nebūs sasniegts visos gadījumos. To var kompensēt ar lielu paaudžu skaitu.



4.11. att. Viena slāņa perceptrona piemērotības grafiks

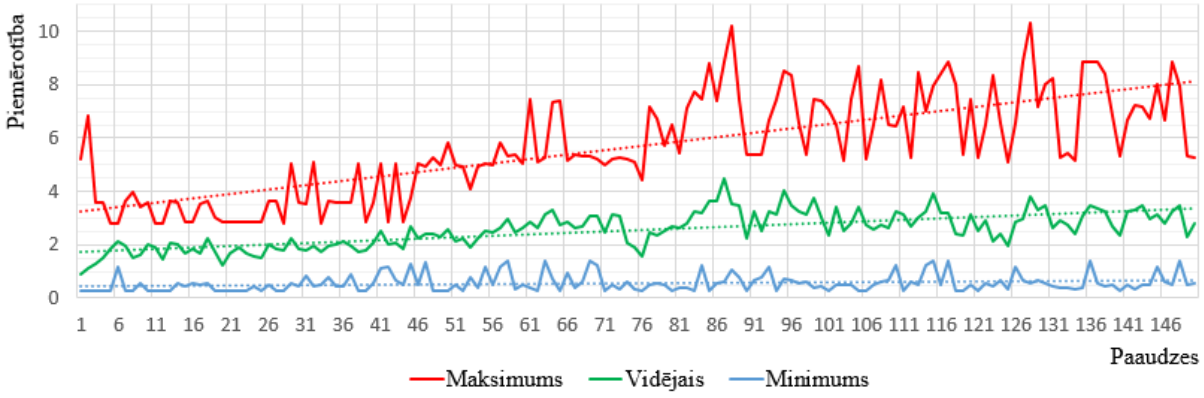
4.6.2. Perceptrons ar diviem slēptiem slāņiem

Divu slāņu perceptrona piemērotības funkciju grafiki parādīti 4.12. att. Kopumā gan maksimālās, gan vidējās vērtības grafikos ir saskatāms pakāpenisks pieaugums līdz aptuveni 88. paaudzei. Pēc tās abi grafiki sāk stipri svārstīties un ievērojamu pieaugumu vairs nav.

Maksimālās vērtības grafiks sākas ar lēcienu, kas, ņemot vērā tā tik pat strauju kritienu, ir veiksmīgas pozīcijas rezultāts. Jau pie 5. paaudzes grafikas stabilizējas ap vērtību 3, līdzīgi kā tas notika viena slāņa perceptrona gadījumā. Pie 28. paaudzes parādās pirmās veiksmīgās mutācijas, kas pilnībā nostiprinās populācijas genotipā aptuveni 15 paaudžu laikā. Līdzīgas mutācijas notiek 62. un 77. paaudzēs, un ja pirmo reizi tām neizdodas nostiprināties, otrajā mēģinājuma labvēlīgās iezīmes tomēr izplatās populācijā un ir redzams ievērojams funkcijas pieaugums, kura rezultātā novērtējuma funkcijas sasniedz iespējamo maksimumu – parādās indivīdi, kas spējīgi izbraukt visu labirintu. Pēc šī punkta notiek straujš kritiens un funkcija vairs nenostiprinās pie kādas vērtības. Tas var nozīmēt, ka sasniegtais kāpums ir tikai veiksmīga sagadīšanās. Tomēr vidējās vērtības grafiks atkārto šādu pašu kāpienu, kas nozīmē, ka labvēlīgo iezīmju nostiprināšanās populācijā tomēr notika.

Vidējās vērtības grafiks visa procesa garumā līdzinās maksimālās vērtības grafikam, kas liecina par vienmērīgu gēnu izplatīšanos starp populācijas indivīdiem. Līdzīgi maksimumam, vidējā vērtība pieaug līdz 72.-88. paaudzei un tad sāk svārstīties, atkārtojot maksimuma grafika uzvedību.

Minimuma grafiks visā garumā svārstās starp 0 un 1,3, līdzīgi attiecīgajam viena slāņa perceptrona grafikam.



4.12. att. Divslāņu perceptrona piemērotības grafiks

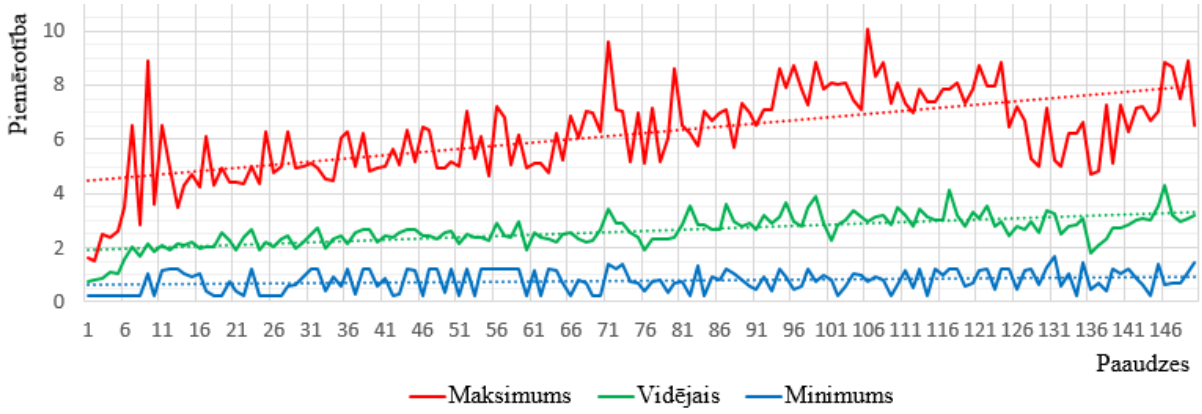
4.6.3. Perceptrons ar trīs slēptiem slāņiem

Trīs slāņu perceptrona piemērotības grafiks parādīts 4.13. att. No iepriekš apskatītajiem piemēriem, šis grafiks atšķiras ar nepārtrauktu maksimālās vērtības svārstību, pie tam saglabājot augšanas tendenci līdz pat 126. paaudzei, kur notiek straujš kritums. Lēciens pie 10. paaudzes ir kārtējais veiksmīgas pozīcijas gadījums, jo tas praktiski neietekmē vidējās vērtības grafiku. Savukārt tāds pats lēciens pie 72. paaudzes visticamāk ir krustošanas rezultāts, jo tajā pašā paaudzē ir novērojami lēcieni arī vidējā vērtībā. Izveidojusies labvēlīgā iezīme nespēj iestiprināties populācijas gēnos, jo nākošajās paaudzēs abos grafikos ir redzama strauja grafika dilšana.

Tomēr neskatoties uz nestabilo maksimālo vērtību, vidējās funkcijas grafiks pakāpeniski pieaug gandrīz visā procesa garumā. Šāda situācija rodas tīkla topoloģijas īpatnību dēļ. Liels slēpto neironu skaits nozīmē ievērojami lielāku savienojamu skaitu un attiecīgi arī garāku genoma virkni. Jo garāks ir genoms, jo sarežģītāk mutācijas mehānismam izveidot jaunas labvēlīgas iezīmes. No otras puses pastāv mazāka varbūtība, ka kādas iezīmes tiek zaudētas.

Minimālo vērtību grafiks uzvedas līdzīgi iepriekš apskatītajiem – svārstās robežās no 0 līdz 1,3.

Kopumā eksperiments parāda, ka veidotā vadības sistēma uzvedas neparedzami, kaut arī tā izskatās stabilāka par divu slāņu perceptronu. Tomēr vidējās vērtības augšana ļauj pieņemt, ka pie daudz lielāka paaudžu skaita sistēma konverģētu uz labu atrisinājumu.



4.13. att. Trīsslāņu perceptrona piemērotības grafiks

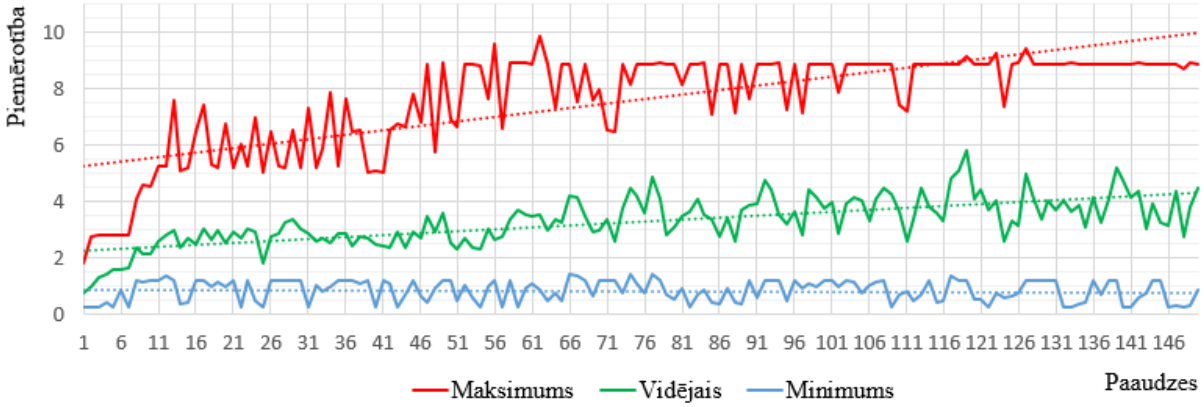
4.6.4. Elmana tīkls

Uz Elmana tīkla balstītas vadības sistēmas piemērotības grafiks parādīts 4.14. att. Maksimālās vērtības grafikā ir divi strauji pieaugumi līdz 15. paaudzei un no 40. līdz 50 paaudzei. Tas liecina par vairākām secīgam krustošanām vai mutācijām. Pēc 50. paaudzes vērtība pakāpeniski stabilizējas pie vērtības 9. Tas nozīmē, ka pāris īpaši veiksmīgi indivīdi nostiprinās elitē, kas saglabājas nemainīga atlikušajā procesa laikā.

Vidējās vērtības grafiks pirmajās 15 paaudzēs strauji aug, atkārtojot maksimālās vērtības grafiku. Pēc tam augšanas ātrums stipri samazinās, tomēr augšanas tendence saglabājas. No 72. paaudzes vidējās vērtības grafiks sāk stipri svārstīties, šis moments sakrīt ar maksimuma nostabilizēšanos. Šāds efekts rodas tāpēc, ka labāko indivīdu genotips pakāpeniski izplatās populācijā, bet augsts mutācijas līmenis šim procesam stipri traucē.

Minimālās vērtības grafiks, kā visos iepriekšējos gadījumos, svārstās intervālā 0-1,3.

No eksperimenta rezultātiem izriet, ka ar ģenētisko algoritmu var ātri iegūt Elmana tīklu robota vadības sistēmai, kas parādītu pietiekami labus un stabilus rezultātus. Tomēr ir jāpārskata algoritma parametri. Īpašu uzmanību pievēršot mutācijas līmenim, jo veiktajā eksperimentā tieši šis faktors ieviesa nestabilitāti populācijā, bet tajā pašā laikā nedrīkst pilnībā atmest mutāciju, jo tā nodrošina populācijas dažādību.



4.14. att. Elmana tīkla piemērotības grafiks

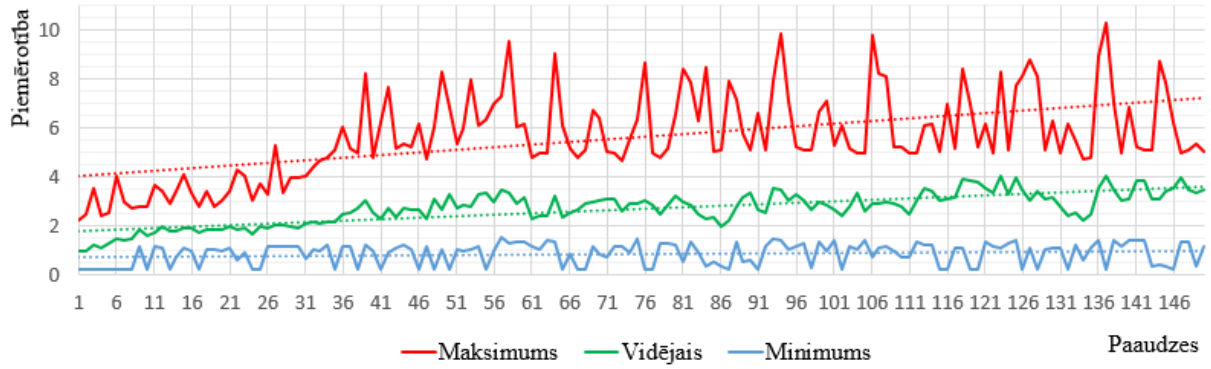
4.6.5. Hopfilda tīkls

Hopfilda tīklā balstītas vadības sistēmas iegūšanas piemērotības grafiks parādīts 4.15. att. Maksimuma vērtības grafiks var tikt sadalīts divos posmos: līdz 40. paaudzei tas ir augošs, un sasniedz maksimālo vērtību 8,5, pēc 40. paaudzes grafiks sāk svārstīties lielās robežās, kas nozīmē, ka labākie indivīdi nav spējīgi saglabāt un izplatīt pozitīvās iezīmes visā populācijā.

Vidējās vērtības grafiks saglabā augšanas tendenci praktiski visa procesa garumā, ar dažiem nelieliem kritumiem pie 80. un 125. paaudzes. Vidējās vērtības augšana norāda uz nelielu skaitu pozitīvo gēnu, kas lēni nostiprinās populācijā, pat neskatoties uz nestabilo labāko indivīdu uzvedību.

Minimālās vērtības uzvedas līdzīgi pārējiem grafikiem – svārstās robežās no 0 līdz 1,3.

Eksperimenta rezultāti parāda, ka Hopfilda tīklā balstītu vadības sistēmas iegūšana ar ģenētisko algoritmu prasa lielu paaudžu skaitu un vienalga negarantē laba tīkla iegūšanu.



4.15. att. Hopfilda tīkla piemērotības grafiks

4.7. Eksperimentu rezultātu kopsavilkums

Salīdzinot ģenētiskā algoritma darbību perceptroniem ar dažādu slēpto slānu skaitu, var pamatoti apgalvot, ka lielāks slēpto slāņu skaits nedod acīmredzamas priekšrocības. Visos trijos gadījumos ir redzams vidējās vērtības pieaugums, tomēr maksimālās vērtības ir nostabilizējušas tikai vienslāņa perceptronam. Tātad tikai šāds perceptrons piedāvā paredzamu uzvedību. Palielinot slāņu skaitu, ievērojami palielinās savienojumu skaits, kas apgrūtina jauno gēnu nostiprināšanos populācijas genotipā, bet tajā pašā laikā nodrošina jau esošu iezīmju saglabāšanu. Papildus neironu ieviešana veido sarežģītāku uzvedību, kas ir daudz jūtīgāka pret sīkām ieejas vērtību svārstībām. Vienslāņu perceptrons divās ļoti līdzīgas situācijās uzvedīsies vienādi, kamēr daudzslāņu perceptrons var katrā situācijā izpildīt atšķirīgas darbības. Cik tas ir labi vai slikti ir lielā mērā atkarīgs no uzstādītā uzdevuma. Šajā konkrētajā eksperimentā priekšroka ir vienkāršībai.

Starp rekurentajiem tīkliem labāku rezultātu uzrādīja Elmana tīkls. Tas īsā laika periodā sasniedz pietiekami augstas derīguma funkcijas vērtības un saglabā tās. Hopfilda tīkls nebija spējīgs izveidot vadības sistēmu ar stabilu rezultātu. Tas ir viegli izskaidrojams ar palielināto savienojumu skaitu, līdzīgi kā tas notiek ar daudzslāņu perceptroniem.

Salīdzinot vienslāņa perceptrona un Elmana tīkla rezultātus, var pamanīt, ka grafiki ir līdzīgi – abos gadījumos notiek vidējās vērtības pieaugums, kamēr maksimuma grafiks vienā brīdī nostabilizējas ap maksimālo vērtību. Tajā pašā laikā ir redzams, ka maksimuma vērtība Elmana tīkla gadījumā pieaug pakāpeniski no procesa paša sākuma, kas liecina par sekmīgu evolucionāro procesu. Vienslāņa perceptrona gadījumā rezultāts tiek sasniegts lēciena veidā, kas ļauj domāt, ka progress ir sasniegts drīzāk veiksmīgas sagadīšanās rezultātā, nekā evolucionārā ceļā.

Pamatojoties uz iepriekšminētajiem secinājumiem, izriet, ka vislabāk robota neironu tīklos balstītas vadības sistēmas izveidei ir piemērots Elmana tīkls. Šo automatizēti iegūtu tīklu var sekmīgi pielietot navigācijas uzdevumu risināšanai.

SECINĀJUMI

Kopsavilkums par darbā padarīto. Darbā apkopota robotu vadības, neironu tīklu un evolucionārās skaitļošanas teorētiskā bāze,. kā arī identificētas esošās pieejas mobilo robotu vadības mehānismu automatizētai iegūšanai ar evolucionārās skaitļošanas metodēm. Izstrādāts un darbībā pārbaudīts mobila robota imitācijas modelis ar kura palīdzību realizēta neironu tīklos sakņotu robota vadības mehānismu iegūšana ar ģenētisko algoritmu.

Darba mērķa sasniegšanas pakāpe un darba uzdevumu izpildes novērtējums. Izvirzītais darba mērķis ir sasniegts – ir identificēti un analizēti evolucionārās skaitļošanas pielietojumi robota vadības mehānismu automatizētai iegūšana. Mērķa sasniegšanai uzstādītie uzdevumi arī ir izpildīti: izveidots teorētiskās bāzes apkopojums, identificēti risinājumi vadības mehānismu iegūšanai, izstrādāts un darbībā pārbaudīts robota imitācijas modelis un automatizētas vadības mehānismu iegūšanas rīks, ar ko veikti eksperimenti un izdarīti atbilstošie secinājumi.

Darbā sasniegtie rezultāti. Darba teorētiskās daļas rezultāts ir apkopojums par populārākajiem pētījumu virzieniem evolucionārajā robotikā. Eksistējošie pētījumi notiek divos savstarpēji saistītos virzienos – robota uzvedības veidošana, izmantojot evolucionāros algoritmus un liela robotu skaita koordinēta kustība, pamatojoties uz spieta algoritmiem. Kamēr pirmajā virzienā notiek tikai akadēmiskie pētījumi, spieta robotikai ir atrasti vairāki risinājumi izlūkošanas un novērošanas jomā.

Darba praktiskās daļas rezultāti ir ar izstrādāto imitācijas modeli un programmatūru veiktie eksperimenti robotu automatizētas vadības mehānismu iegūšanai ar ģenētisko algoritmu noskaņojot atbilstošos neironu tīkla svarus. Eksperimenti veikti pielietojot dažādas neironu tīklu topoloģijas - viena, divu un trīs slāņu perceptronus, Elmana tīklu un Hopfilda tīklu. Visos gadījumos tika iegūts vismaz viens tīkls, kuru pielietojot kā vadības mehānismu robots spēj veiksmīgi pārvietoties nezināmā vidē apejot šķēršļus. Tomēr, neskatoties uz veiksmīgo gala rezultātu visos izmēģinājumos, algoritma darbības efektivitāte dažādu topoloģiju iegūšanai ievērojami atšķiras. No apskatītajām tīklu topoloģijām, vislabākais rezultāts tika iegūts ar Elmana tīklu. Veiktie eksperimenti ļauj spriest par evolucionārās skaitļošanas efektivitāti automātiskai robotu vadības mehānismu veidošanai. Elmana tīkla apmācībai navigācijas uzdevuma risināšanai nepieciešamas tikai 150-200 paaudzes, citām

tīklu topoloģijām līdzīgas uzvedības iegūšanai nepieciešams ievērojami lielāks paaudžu skaits.

Turpmāko pētījumu virzieni. Viens no turpmāko pētījumu virzieniem ir pētījums par ģenētiskā algoritma parametru ietekmi uz evolucionārā procesa efektivitāti. Tas iekļauj ne tikai skaitliskos parametrus (paaudžu skaits un lielums, elites izmērs, mutācijas varbūtība), bet arī dažādus genotipa attēlojuma veidus un atšķirīgos ģenētiskos operatorus (selekcija, krustošanās, mutācija).

Literatūrā tiek minēti arī citi robotu vadības sistēmu veidi, piemēram, nestriktās loģikas vadības mehānismi, kuru iegūšana ar ģenētiskā algoritma palīdzību arī ir iespējama. (Wang, Tan u.c., 2006). Nepieciešams izpētīt šo vadības mehānsmu iegūšanu ar ģenētisko algoritmu un veikt salīdzinājumu ar neironu tīklos sakņotajiem. Vēl viens potenciāls pētījumu virziens ir dažādu vadības mehānismu apvienojumu iegūšana un izmantošana, piemēram, nestriktās loģikas un neironu tīklu apvienojums.

Evolucionārā skaitļošana ietver arī citas pieejas, ne tikai ģenētisko algoritmu. Piemēram, ģenētiskā programmēšana tiek bieži pieminēta neironu tīklu un citu vadības sistēmu iegūšanas kontekstā (Rivero, Dorado u.c., 2010). Cits perspektīvs pētījumu virziens ir spieta algoritmi gan viena robota vadības sistēmās, gan vairāku autonomu robotu vadībai (Tan & Zheng, 2013).

LITERATŪRA

Abiyev, R., Ibrahim, D. & Erin, B. Navigation of mobile robots in the presence of obstacles. *Advances in Engineering Software,* 41**,** 1179-1186, 2010.

Angelo, J. A., *Robotics : A Reference Guide to the New Technology,* Westport, Conn, Greenwood Press, 2007,

Anykode. 2013. *Marilou - Key Features* [Online]. Available: http://www.anykode.com/mariloukeyfeatures [Accessed March 2014].

Auke Ji, A. M., Aude B, Gambardella Lm. Collaboration through the exploitation of local interactions in autonomous collective robotics. The stick pulling experiment. *Autonimous Robot,* 149, 2001.

Banzhaf, W., *Evolutionary Computation and Genetic Programming* in: *Engineered Biomimicry,* Boston, Elsevier, 2013.

Bekey, G. A., *Robotics : State of the Art and Future Challenges,* London, Imperial College Press, 2008,

Brownlee, J., *Clever Algorithms: Nature-Inspired Programming Recipes* Lulu.com, 2011,

Buhmann, M. D., *Radial Basis Functions : Theory and Implementations,* Cambridge, Cambridge University Press, 2003,

Cook, G., *MOBILE ROBOTS: Navigation, Control and Remote Sensing,* USA, Wiley-IEEE Press, 2011,

Corne, D. & Fogel, G., *Evolutionary Computation in Bioinformatics,* Amsterdam, Morgan Kaufmann Publishers, 2003,

Eberhart, R. C. & Shi, Y., *Computational Intelligence : Concepts to Implementations,* Amsterdam, Elsevier/Morgan Kaufmann Publishers, 2007,

Floreano, D. & Mattiussi, C., *Bio-Inspired Artificial Intelligence : Theories, Methods, and Technologies,* Cambridge, MA, USA, MIT Press, 2008,

Graupe, D., *Principles of Artificial Neural Networks (3rd edition),* Singapore, WSPC, 2013,

Yang, Y. W., Xu, J. F. & Soh, C. K. An evolutionary programming algorithm for continuous global optimization. *European Journal of Operational Research,* 168**,** 354-369, 2006.

Yoo, S. J. Adaptive neural tracking and obstacle avoidance of uncertain mobile robots with unknown skidding and slipping. *Information Sciences,* 238**,** 176-189, 2013.

Yoon, Y. & Kim, Y.-H. Geometricity of genetic operators for real-coded representation. *Applied Mathematics and Computation,* 219**,** 10915-10927, 2013.

Karafotias, G., Haasdijk, E. & Eiben, A. E. An Algorithm for Distributed On-line, On-board Evolutionary Robotics. 8, 2011.

Khalil, W. & Dombre, E., *Modeling, Identification & Control of Robots,* London, Kogan Page Science, 2004,

Kohonen, T., *Self-Organizing Maps (Third Extended Edition),* New York, Springer, 2001,

Lin, J. & Lian, R.-J. Hybrid fuzzy-logic and neural-network controller for MIMO systems. *Mechatronics,* 19**,** 972-986, 2009.

Matarić, M. J., *The Robotics Primer,* MIT Press, 2007,

Matveev, A. S., Wang, C. & Savkin, A. V. Real-time navigation of mobile robots in problems of border patrolling and avoiding collisions with moving and deforming obstacles. *Robotics and Autonomous Systems,* 60**,** 769-788, 2012.

Miikkulainen, R. 2011. Evolving neural networks. *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation.* Dublin, Ireland: ACM.

Moubarak, P. & Ben-Tzvi, P. Modular and reconfigurable mobile robotics. *Robotics and Autonomous Systems,* 60**,** 1648-1663, 2012.

Nelson, A. L., Barlow, G. J. & Doitsidis, L. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems,* 57**,** 345-370, 2009.

Nelson, A. L. & Grant, E. Using direct competition to select for competent controllers in evolutionary robotics. *Robotics and Autonomous Systems,* 54**,** 840-857, 2006.

Petrič, T. & Žlajpah, L. Smooth continuous transition between tasks on a kinematic control level: Obstacle avoidance as a control problem. *Robotics and Autonomous Systems,* 61**,** 948-959, 2013.

Qian, L. Regularized Radial Basis Function Networks: Theory and Applications. *Technometrics,* 44**,** 294, 2002.

Rivero, D., Dorado, J., Rabuñal, J. & Pazos, A. Generation and simplification of Artificial Neural Networks by means of Genetic Programming. *Neurocomputing,* 73**,** 3200-3223, 2010.

Ruan, X. & Dai, L. Vehicle Study with Neural Networks. *Physics Procedia,* 25**,** 814-821, 2012.

Samsudin, K., Ahmad, F. A. & Mashohor, S. A highly interpretable fuzzy rule base using ordinal structure for obstacle avoidance of mobile robot. *Applied Soft Computing,* 11**,** 1631-1637, 2011.

Shi, Z., *Advanced Artificial Intelligence,* Singapore, World Scientific, 2011,

Sibi, P., Jones, S. A. & Siddarth, P. Analysis of different activation functions using back propogation neural networks. *Journal of Theoretical & Applied Information Technology,* 47**,** 1344-1348, 2013.

Simon, D., *Evolutionary Optimization Algorithms,* Wiley, 2013,

Stenning, B. E. & Barfoot, T. D. Path planning with variable-fidelity terrain assessment. *Robotics and Autonomous Systems,* 60**,** 1135-1148, 2012.

Tan, Y. & Zheng, Z.-Y. Research Advance in Swarm Robotics. *Defence Technology,* 9**,** 18-39, 2013.

Tzafestas, S. G., *Introduction to Mobile Robot Control* in: *Introduction to Mobile Robot Control,* Oxford, Elsevier, 2014.

Wang, Y.-J. & Zhang, J.-S. Global optimization by an improved differential evolutionary algorithm. *Applied Mathematics and Computation,* 188**,** 669-680, 2007.

Wang, L., Tan, K. C. & Chew, C. M., *Evolutionary Robotics : From Algorithms to Implementations,* New Jersey, World Scientific Publishing, 2006,

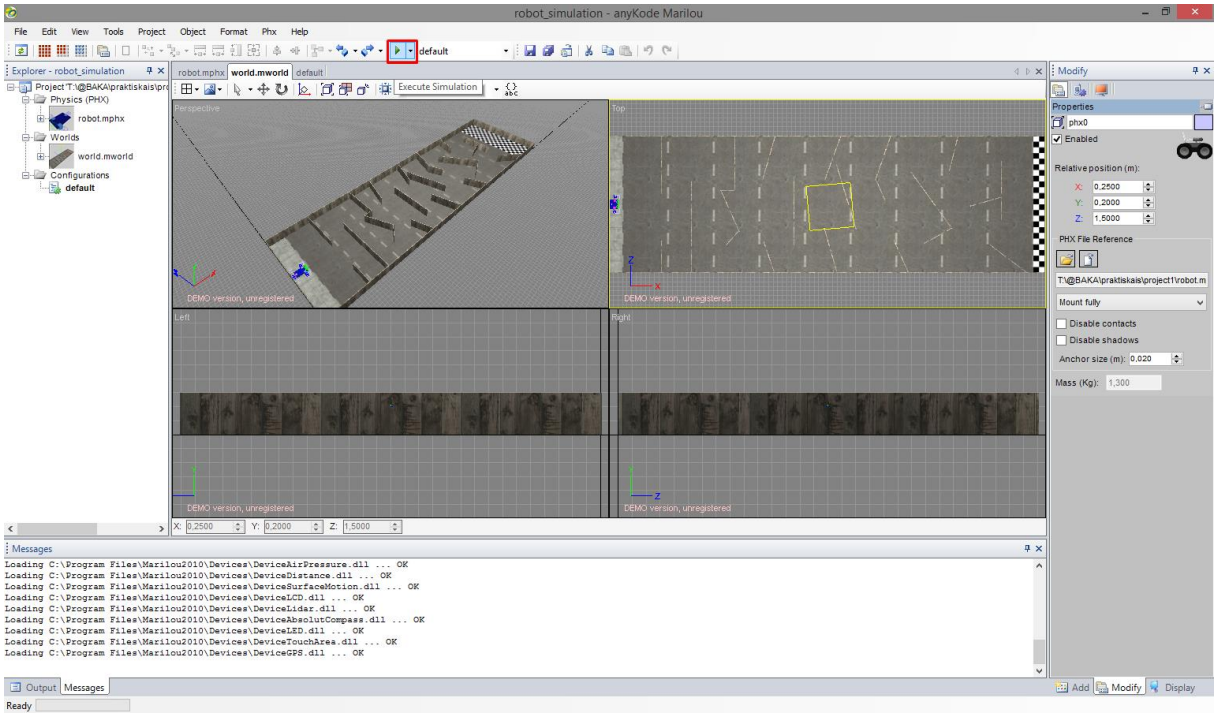
Zhang, W., *Computational Ecology : Artificial Neural Networks and Their Applications,* Singapore, World Scientific, 2010,

PIELIKUMI

1. pielikums. Izstrādātās lietotnes lietošanas instrukcija

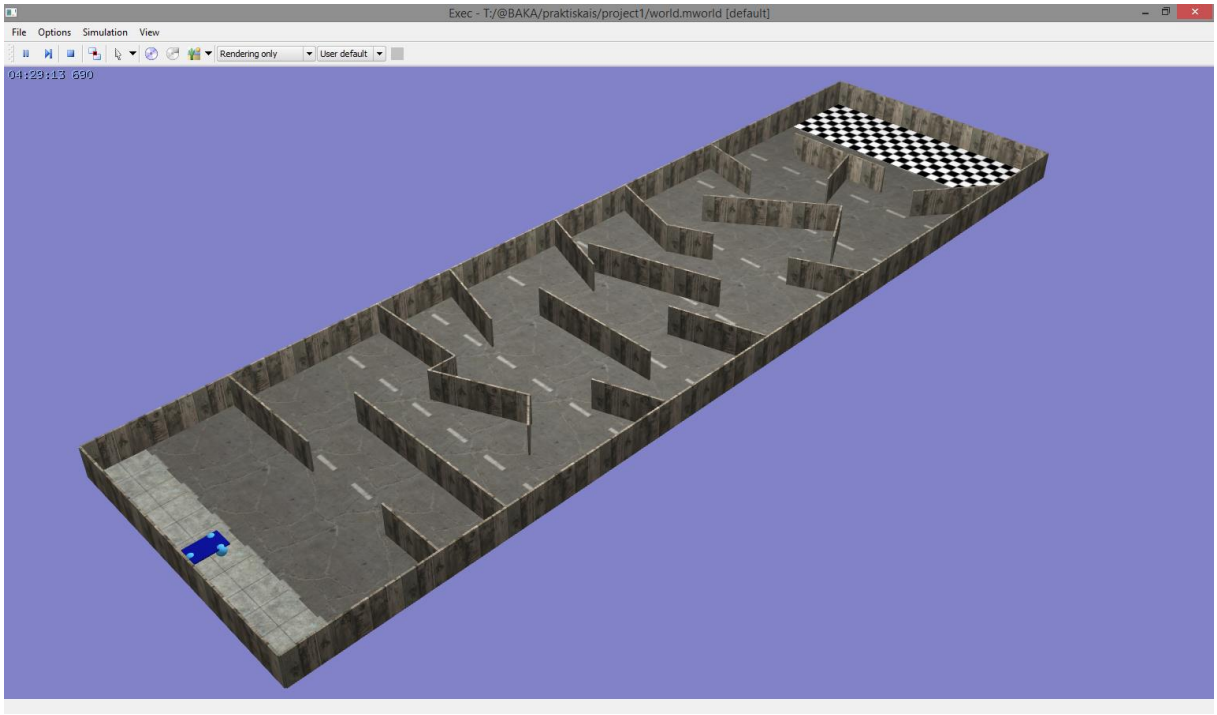
"Neuroevolution controller" lietotnes pilns pirmkods kopā ar pasaules un robota imitācijas modeļiem ir pieejams Git versiju kontroles sistēmas repozitorijā https://bitbucket.org/sluhmirin/bakalaura-darbs/src. Imitācijas modeļa palaišanai ir nepieciešama anyKode Marilou 2010 (vai jaunāka versija). Kods ir veidots kā Microsoft Visual Studio 2012 projekts, un programmu ir ieteicams palaist atkļūdošanas režīmā.

Pirms lietotnes izmantošanas nepieciešams palaist vides imitāciju anyKode Marilou imitācijas rīkā. Pasaules imitācijas palaišanai nepieciešams nospiest attēlā P.1 atzīmēto pogu.



P. 1 AnyKode Marilou imitācijas rīks

Palaistas imitācijas pasaules logs parādīts attēlā P.2. Augšējā kreisajā stūrī atrodas laika kontroles pogas, peles režīma izvēle, video un ekrānattēlu uzņemšanas pogas. Šajā brīdī imitācijas modelis ir gatavs darbam un lietotne ir spējīga tam pieslēgties caur tīkla portu.

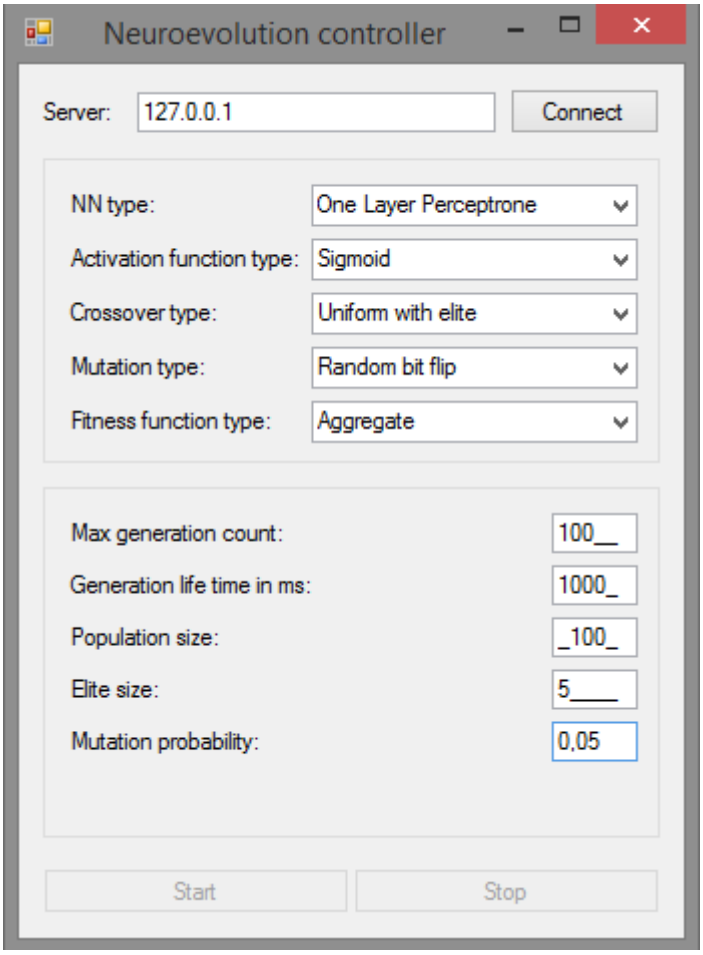


P. 2 Palaists pasaules imitācijas modelis

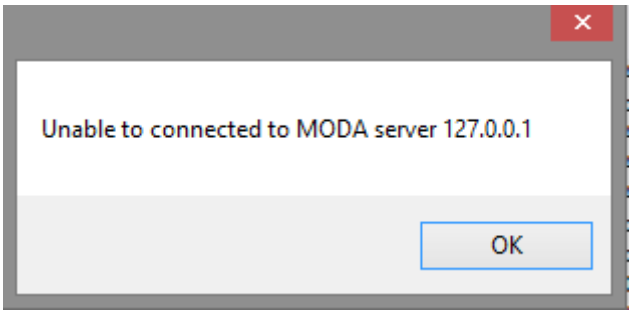
Lietotnes saskarne ir parādīta attēlā P.3. Pēc programmas palaišanas attiecīgajā laukā ir jāievada IP adrese datoram, uz kura tiek darbināts imitācijas modelis. Parasti tas ir lokālais dators, tāpēc pēc noklusējuma tiek piedāvāta lokālā datora adrese „127.0.0.1”. Par veiksmīgu pieslēgšanos liecinās “Start” pogas aktivizēšanās, bet kļūdas gadījumā tiks parādīts attiecīgs paziņojums (attēls P.4).

Pēc pieslēgšanās lietotājam ir jāievada eksperimenta parametri – pirmajā grupā ir jāizvēlas no piedāvātajiem variantiem, otrajā jāievada skaitļi. Pēc parametru ievadīšanas lietotājam jāspiež “Start” poga.

No šī brīža lietotājs var tikai apstādināt procesu ar pogas “Stop” palīdzību. Procesa apstāšanās nenotiek momentāni, pašreizējās paaudzes pārbaude tiek pabeigta līdz galam.



P. 3 Lietotnes grafiskā saskarne



P. 4 Neveiksmīgas pieslēgšanas kļūdas paziņojums

Eksperimenta rezultāti tiek ierakstīti .xlsx failā programmas mapē ar nosaukumu "finished_<datums>_<laiks>.xlsx". Fails satur vienu darba lapu ar tabulu, kuras rindas satur vienas paaudzes kārtas numuru pirmajā kolonnā un katra paaudzes indivīda novērtējumus pārējās kolonnās.