KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

TAUTVYDAS LOKYS

GENETIŠKAI APMOKYTO NEURONINIO TINKLO PRITAIKYMAS ĖJIMAIS PAREMTO ŽAIDIMO PLĖTROJE

Magistro darbas

Darbo vadovas
doc. dr. A. Ostreika

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

TAUTVYDAS LOKYS

GENETIŠKAI APMOKYTO NEURONINIO TINKLO PRITAIKYMAS ĖJIMAIS PAREMTO ŽAIDIMO PLĖTROJE

Magistro darbas

Darbo vadovas
doc. dr. A. Ostreika
2013-05-

Recenzentas:
prof. dr. B. Paradauskas
2013-05-

Atliko:
IFM-1/1 gr. studentas
Tautvydas Lokys
2013-05-

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

20..... - ..... - ..... d.
Kaunas

Autoriai,_____
(vardas, pavardė)

_____,

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis bakalauro (magistro) darbas (toliau vadinama – Kūrinys)_____
(kūrinio pavadinimas)

_____

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autoriai

_____ (vardas, pavardė)	_____ (parašas)
_____ (vardas, pavardė)	_____ (parašas)
_____ (vardas, pavardė)	_____ (parašas)
_____ (vardas, pavardė)	_____ (parašas)

SANTRAUKA

Jau senai dirbtinis intelektas ir iš jo kilę agentai yra pritaikomi įvairių žanrų žaidimuose. Agentų elgsenai pagerinti naudojami įvairūs optimizacijos algoritmai, tokie kaip genetinis algoritmas. Tam kad agentas sugebėtų įvykdyti užduotą tikslą, jis turi įvertinti aplinką ir kitų agentų veiksmus.

Darbo tikslas – pritaikyti dirbtinio intelekto metodus ėjimais paremto žaidimo „Warlike“ agentų elgsenai pagerinti. Darbe pritaikyti dirbtinio intelekto metodai: genetiniu algoritmu apmokytas neuroninis tinklas, įtakos žemėlapiai, antikūnių gamybos planuotojas. Šie metodai praplečia agento pasirinkimų kiekį, bei kareivių (žaidimo vienetų, kuriais operuoja agentai) tarpusavio veiksmų derinimą.

Tyrimo tikslas – ištirti ir palyginti pritaikytus dirbtinio intelekto metodus, ir nustatyti, kurie iš jų duoda geriausius rezultatus ėjimais paremtam žaidime.

Darbo rezultatus galima panaudoti panašių ėjimais paremtų žaidimų agentų elgsenos gerinimui bei elgsenos klaidų radimui. Taip pat ir kitokių karinių žaidimų dirbtinio intelekto kūrimui.

APPLICATION OF GENETICALLY TRAINED NEURAL NETWORK IN TURN BASED GAME DEVELOPEMENT

SUMMARY

Agents controlled by an artificial intelligence have been used in games for a long time. Agent behaviour can be improved using optimization algorithms (such as genetic algorithm). To achieve a goal, an agent needs to understand his environment and actions made by other agents.

The system implements the following methods for artificial intelligence: genetic algorithm, artificial neural network, influence maps. These methods are used to model actions made by an agent, and to coordinate those actions with other agents. They are applied specifically to solve agent related problems in a „Warlike“ turn-based game.

The goal of this study is to compare the artificial intelligence methods used in a turn-based game.

The results achieved in this study can be used to find balancing errors in similar turn-based games and to develop turn-based game artificial intelligence.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLĖLIŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

NT (angl. *Neural Network*) – neuroninis tinklas.
GA (angl. *Genetic Algorithm*) – genetinis algoritmas.
GTP – Genetinis tinklo programavimas
IGTP - Imunitetu tobulinamas genetinis tinko programavimas.
GTPIPM - genetinio tinklo naudojant imuniteto prisitaikantį mechanizmą.
ĮŽ – (angl. *Influence Map*) įtakos žemėlapiai.
DI – (angl. *Artificial Intelligence*) dirbtinis intelektas.
GP – gamybos planuotojas.

ĮVADAS

1.1 Problemos aktualumas

Dirbtinis intelektas jau senei yra tiriamas ir taikomas skirtingų tipų žaidimams. Dabar agentų elgesys yra pagerinamas naudojant optimizacijos algoritmus (tokius kaip genetinis algoritmas), kurie randa ir nustato parametrus, kad agentų našumas žaidimuose pagerėtų. Genetinis algoritmas (GA) turi begalę teigiamų pliusų.

Neuroninis tinklas (NT) naudoja paprastą struktūrą skirtą informacijos apdorojimui ir šablonų atpažinimui, kuris sukonstruotas remiantis biologiniais neuroniniais tinklais.

Augant kompiuterių procesorių galimybėms, galima pasitelkti įdomesnes, arčiau realybės esančias imitacijas. Šiuo metu yra sukaupta pakankamai informacijos apie: genetiką (paveldimumą, kryžminimą, selekciją ir mutaciją); imuninę sistemą (antikūnius ir antigenus). Informacija apie genetika ir evoliucija padeda pagreitinti skaičiavimo algoritmus, kurie įprastai užima daug laiko, nes turi didele veiksmu aibę. Dažnai naudojant genetiniu algoritmu apmokomą neuroninį tinklą, galima pasiekti geresnių rezultatų. Tai leidžia kurti tikroviškesnį ir efektyvesnį žaidimuose naudojamą dirbtinį intelektą.

Daugybės agentų sistema problemai spręsti naudoja kelis ar daugiau agentų [6]. Kaip agentų evoliucijos metodą, naudosime bendrosios-evoliucijos (angl. *co-evolution*) metodą, norint paskatinti kiekvieno kario individualius sprendimus.

Kad agentas sugebėtų pasiekti tikslą, jis turi suvokti aplinką ir agentų veiksmus [6]. Todėl šiame darbe taikant kai kuriuos metodus, agentai geba suvokti supančią aplinką, bei įtakoti kitų agentų veiksmus. Valdyti kelis agentus, kurie siekia to pačio tikslo, yra ganėtinai sunku, nes kiekvienas veiksmas įtakoja kitų agentų veiksmus ir vieni kitiems trukdyti.

Darbe ištirti metodai skirti dirbtiniam intelektui realizuoti: genetiniu algoritmu apmokytas neuroninis tinklas, įtakos žemėlapių struktūra ir gamybos planuotojas. Šie metodai praplečia agento pasirinkimų kiekį, bei tarpusavio kareivių veiksmų derinimą. Daugybės tyrimų nagrinėjančių dirbtinį žaidimų intelektą rezultatai gauti, pritaikius tų tyrimų metodus tik konkretiems žaidimams. Naujų tyrimų prisireikia atrandant naujus metodus ar jų derinius, ar taikymą konkretiems uždaviniams spręsti.

Tyrimo objektas - trijų dirbtinio intelekto metodų pritaikymas konkrečiam ėjimais paremto žaidimui.

Tyrimo tikslas: palyginti ir išsiaiškinti, kurie iš trijų dirbtinio intelekto metodų pritaikytų ėjimais-paremtam žaidimui duoda geriausius rezultatus.

1.2 Darbo uždaviniai

Pagrindiniai darbo uždaviniai:

- atlikti panašių realizuotų sprendimų analizę
- atlikti žaidimuose taikomų dirbtinio intelekto metodų analizę ir išanalizuoti:
    - genetinį algoritmą
    - įtakos žemėlapius
    - mikro ir makro strategijas
    - neuroninį tinklą
- mokslinės literatūros analizavimas, siekiant įsigilinti į matematinį bei programinį algoritmų realizavimą
- sukurti dalį ėjimais paremto žaidimo sistemos, kurios užtektų tyrimui atlikti
- palyginti tirtų metodų gautus rezultatus
- įvertinti naudojantis gautais palyginimo rezultatais, kuris iš metodų geriausiai tinka „Warlike“ ėjimais paremto žaidimo dirbtiniam intelektui

1.3 Mokslinis naujumas

Šiuo metu jau yra įvairių genetiniu algoritmu apmokytų neuroniniu tinklu valdomų dirbtinių žaidimų intelektų. Tačiau dauguma jie būna pritaikyti ir apmokyti išskirtinai konkrečiam žaidimui.

Šis darbas yra pritaikytas konkretiems „Warlike“ ėjimais paremto žaidimo dirbtinio intelekto metodams palyginti ir įvertinti.

2 ANALIZĖ

Šioje darbo dalyje yra išsamiai išanalizuojami metodai, atliekama panašių sprendimų ir atliktų tyrimų analizė. Atliekama panašių sprendimų lyginamoji analizė, išsiaiškinami privalumai ir trūkumai.

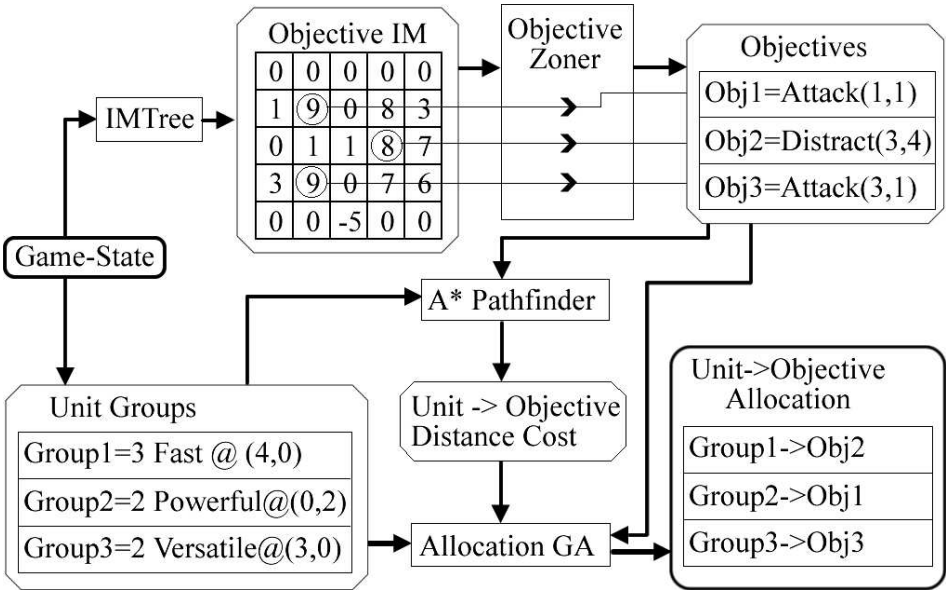
2.1 Panašių sprendimų analizė

Panašių sprendimų analizėje analizuojami panašūs tyrimai ir egzistuojančios sistemos. Vertinant kolegų taikomus metodus, atkreipiamas dėmesys į jų gautus darbo rezultatus. Darbe netaikoma eksperimentais pagrįsta analizė, nors ji tinka ir algoritmų tyrimui, tačiau norint patikrini algoritmo veikimą, kiekvienam algoritmui realizuoti reikia sukurti programinę įrangą.

2.2 Strateginio žaidimo intelektas paremtas įtakos žemėlapiais

Autorių tikslas buvo ištirti realaus-laiko kompiuteriniuose žaidimuose panaudojamą genetinį algoritmą. Autoriai naudodami įtakos žemėlapius, sudarė aplinkinių sektorių blokus. Šie blokai buvo saugomi apsirašytose medžių struktūrose, kaip komplikuotos žaidimo strategijos [7]. Autorių atliktame tyrime palyginta bendradarbiaujančios evoliucijos ir paprasto (angl. *hand-coded*) intelekto pasiekti rezultatai.

Nagrinėjamam tyrime kiekvienas iš kareivių turi savo privalumus ir trūkumus. Pirmiausia formuojamas įtakos žemėlapis suteikiantis motyvaciją aplinkoje. Po to naudojamas genetinis algoritmas karių pasiskirstymo žemėlapyje problemai išspręsti (2.1 pav.):



2.1 pav. Žaidimo agento valdymo architektūra (Chris; Sushil, 2006)

Veikimo eiga pagal schemą viršuje:

1) Formuojamas įtakos žemėlapis.
2) Tikslų parinkėjas (angl. *Objective zoner*) konvertuoja įtakos žemėlapio (angl. *Objective IM*) duomenis į svarbiausius žemėlapio tikslus (angl. *Objectives*), kurie yra skirti kareiviams. Kiekvienas tikslas gauna ir veiksmo tipą: pulti nurodyti objektą, ginti nurodytą objektą, ar eiti į nurodytą vietą. Taip pat kiekvienam taikiniui priskiriami ir kareivių tipai, kurie

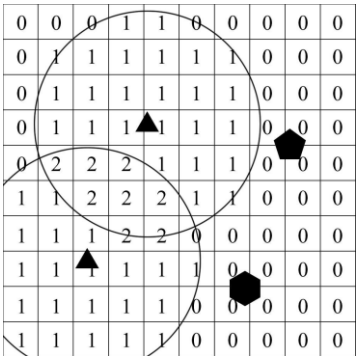
sugebėtų įvykdyti paskirtą užduotį pvz.: jei taikinys yra kavalerija, tai prie taikinio bus pridėtas papildomas ietininkų tipas. Nes ietininkai specializuojasi kavalerijos neutralizavimu.

3) Genetinis algoritmas ,,apdoroja“ kareivių grupes (angl. *Unit Groups*) stengdamasis rasti kuo geresnius taikinius, skirtus kiekvienai grupei individualiai. Tokiu būdu gaunami geresni grupių rezultatai.
4) A* kelio ieškotojas (angl. *Pathfinder*) nusako pasiskirstymo svorius teritorijose. Teritorijos esančios toli arba labai pavojingose vietose turi didelius svorius, priešingai nei artimos ir saugios.
5) Skirstytojas (angl. *Allocation*) tikrina kiek parinkti taikiniai grupėms yra tinkami ir kaip lengvai grupėms seksis pasiekti taikinius.

Šis procesas gali būti kartojamas tol kol visi žaidėjo kareiviai gauna po individualų taikinį.

Autorių pritaikyto įtakos žemėlapio matricos kiekviena celė turi savo reikšmę, kuri dažniausiai priklauso nuo konkrečios funkcijos. Įtakos žemėlapiai naudojami ir vienam sunkiausių dirbtiniam intelektui įveikiamų žaidimų ,,Go“. Nors šis stalo žaidimas vienas seniausių pasaulyje, vis dar nėra realizuota sistema, prieš kurią žaisdamas ,,Go“ žaidimo profesionalas pralaimėtų, priešingai nei kitam senam stalo žaidimui ,,Šachmatai“ [1][2].

Dažnai įtakos žemėlapiai sujungiami kartu tam, kad suformuotų erdvinę sprendimų priėmimo strategiją. ĮŽ-ių funkcijos suma gali atspindėti resursus, atstumą iki priešų žemėlapyje, aplink esančių draugiškų karių ar tikslo pozicijas (žr. 2.5 pav.).



2.2 pav. Žaidimo pasaulio įtakos žemėlapis (Chris; Sushil, 2006:3)

Viršuje esančiam paveiksle pavaizduoti trikampiai atspinti oponento kareivius. Aplinkui kiekvieną iš trikampių padidinamos įtakos žemėlapio reikšmės. O jų regionų sankirtoje reikšmės sumuojamos ir tai gali atspindėti labai patrauklią arba priešingai nepatogią poziciją.

ĮŽ apdorojami iki atskirų karių grupių įsakymų:

- visi ĮŽ išanalizuojami
- atrinkti langeliai konvertuojama į įsakymus
- įsakymai priskiriami atskiroms kareivių grupėms

Dažniausiai naudojamas ne vienas, o keli ĮŽ. Jie gali būti skirti apibrėžti įvairiai informacijai: žemėlapiams, karių pozicijoms, strategijoms. ĮŽ skirti saugoti priešininko grupių įtakai, kur dažnai saugomos didelės neigiamos reikšmės aplink galingus oponento kareivius ir didelės teigiamos aplink silpnus. Visi įtakos žemėlapiai sumuojami į vieną ir didžiausios teigiamos reikšmės yra patrauklios puolimui išvengiant galingų oponento kareivių.

Autorių Chris ir Sushil tyrimo rezultatai parodė, kad naudojamas įtakos žemėlapis padeda koordinuoti kareivių grupes tarpusavyje judančias karo lake [7].

2.2.1 Bendradarbiaujančios ir Nebendradarbiaujančios Evoliucijos Įtaka

Hirotaka Itoh, Naoki Ikeda ir Kenji Funahashi tyrime buvo lyginami bendradarbiaujančios (angl. *co-evoliution*) ir nebendradarbiaujančios evoliucijos (angl. *non-co-evoliution*) įtaka trims skirtingiems metodams.

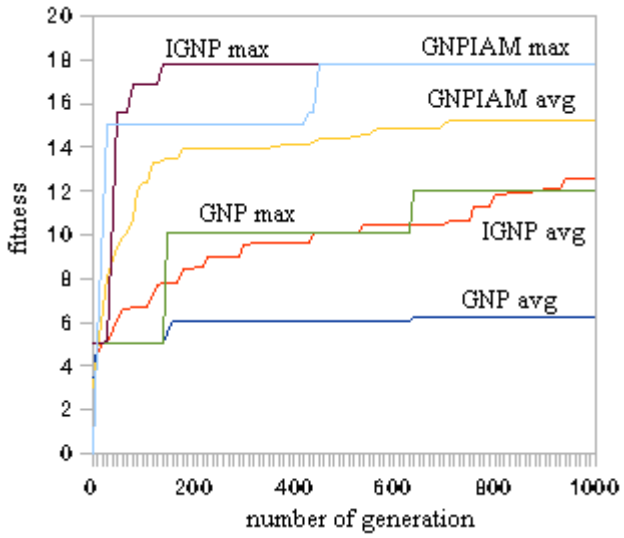
Tyrime buvo naudotas Genetinio Tinklo Programavimas (GTP) kaip generavimo technika heterogeniniai daugiagenetiniai sistemai. Tyrimo metu pastebėta, kad viena iš priežasčių dėl pagerėjusių problemos sprendimo buvo dviejų ar daugiau agentų panaudojimas užduočiai, kurią galėtų išspęsti ir vienas agentas ir naudojant bendradarbiaujančią evoliuciją. Geresniam rezultatui pasiekti, straipsnio autoriai skatino agentus kooperuotis bendrai užduočiai atlikti [6].

Autoriai agentų veiksmus sugeneravo automatiškai, naudojant genetinį tinklo programavimą. Genetinis tinklo programavimas (GTP) tai yra patobulinta Genetinio Programavimo (GP) technika. GP technika yra gera tuomet, kai duomenys statiški ir retai kinta. GTP naudojama tinklo struktūra yra pritaikoma dirbant su duomenimis kintančiais ,,realiu laiku".

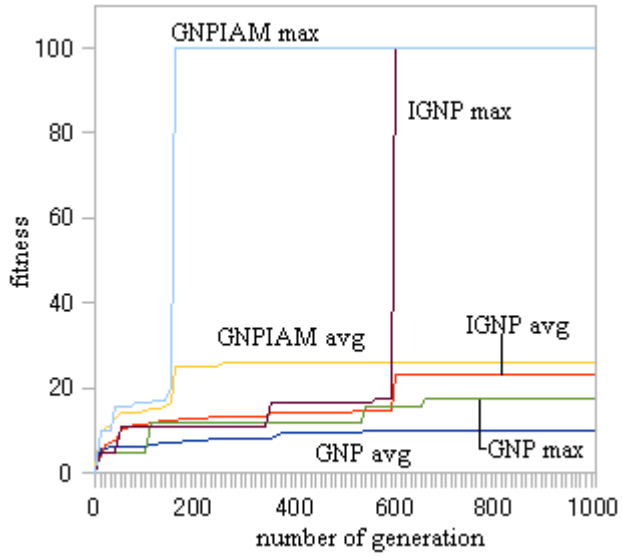
GTP evoliucinis algoritmas yra toks pats kaip GP. Pasak straipsnio autorių ir jų gautų rezultatų, egzistuoja ir geresnė evoliucijos technika. Ji vadinama (GAIPM) genetinis algoritmas su imunitetiniu Prisitaikančiu Mechanizmu (angl. *Genetic Algorithm with Immune Adjustment Mechanism*). Pastebėta, kad naudojant GAIPM evoliucijai, galima kur kas pagreitinti GA veikimą (2.3 pav).

Taip pat, Imuninis algoritmas (IA) gali būti panaudotas kaip evoliucijos algoritmas genetinio tinklo programavime. Tokiu atveju gaunamas Imunitetu tobulinamas genetinis tinko programavimas (IGTP). Kur galiausiai tai pritaikoma genetinio tinklo programavimui naudojant imuniteto prisitaikantį mechanizmą (GNPIPM), kuris naudoja GAIPM evoliucijai.

Tyrėjų tikslas buvo palyginti trijų GTP, IGTP ir GTPIPM genetinių algoritmų bendradarbiaujančios ir nebendradarbiaujančios evoliucijos apmokytos neuroninio tinklo sistemos darbo rezultatus.



2.3 pav. Apsimokymo rezultatai naudojant nebendradarbiaujančią evoliuciją (Hirotaka Itoh, Naoki Ikeda, Kenji Funahashi, 2008:6)



2.4 pav. Apsimokymo rezultatai naudojant bendradarbiaujančią evoliuciją (Hirotaka Itoh; Naoki Ikeda; Kenji Funahashi, 2008:6)

Aukščiau pavaizduoti daugybės agentų apsimokymo genetiniu algoritmu rezultatai, čia:

Fitness- tinkamumo funkcija, kuri ir apibrėžia rezultatų tinkamumą;

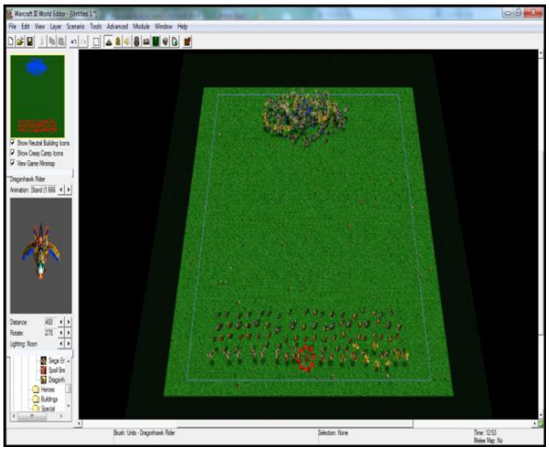
Number of generation – genetiniu algoritmo panaudotų generacijų skaičius;

Nebendradarbiaujančios evoliucijos (angl. *non-co-evolution*) geriausi gauti rezultatai buvo GTPIPM ir IGTP. Tris kartus prasčiau pasirodė naudojant GTP (žr. 1pav.). Bendradarbiaujančios evoliucijos (angl. *co-evolution*) geriausi gautų rezultatų vidurkiai buvo GTPIPM ir IGTP ir du kartus prasčiau pasirodė naudojant GTP (2.2 pav.).

Iš autorių pateiktų tyrimo rezultatų matyti, kad taikant bendradarbiaujančią evoliuciją rezultatai gaunami visais trimis naudotais metodais. aukštesni naudojant net ir GTP.

2.2.2 Evoliucinis Neuroninis Kontroleris Naudojantis GA „WarCraft III" Žaidime

Chang Kee Tong, Chin Kim On, Jason Teo, ir Aroland MConie Jilui Kiring autorių tikslas buvo pritaikyti genetinio algoritmo evoliuciją dirbtinio neuroninio tinklo svoriam rasti, kuris atliktų karių gamybos ir valdymo kontrolerio funkciją [6]. Suprojektuotas kontroleris, kiekvieną kartą turi nuspręsti kiek ir kokius kareivius gaminti, realaus laiko strateginiam žaidime „Warcraft3", tam kad įveiktų oponento armiją (2.5 pav.).



2.5 pav. Žemėlapis naudojant ,,WorldEdit“ editorių (Chang; Chin; Jason; Aroland, 2011:3)

Naudotas genetinisalgoritmas Chang; Chin; Jason; Aroland, [2011:3]:

,,1.0) Start.

2.0) Initialize the population (10 chromosomes) by generate random real value between N(-1,1) that representing the weight of the FFANN.

3.0) Loop.

3.1) Randomly spawn enemy unit based on food limit. Those values are then used as input for FFANN.

3.2) An Individual is loaded into the FFANN and output is calculated. Output is then decoded into integer value that represents the required amount of unit that should be spawned for each type of unit. After all units had been spawned, the game starts by ordering both groups of armies to move and attack at the centre of the map.

3.3) Game ends if the food from either one of the side reaches zero or the time of the game elapses 180.0 second. Then, the fitness of the individual will be calculated and the remained unit will be cleared away from the map.

3.4) Step 3.1, 3.2 and step 3.3 are repeated until all individuals in the current population have been evaluated.

4.0) Elitism is applied to find the individual that having highest fitness and keep the best fit individual as the first offspring of the next generation.

5.0) Loop.

5.1) Tournament selection is used to select parents (parent one, P1 and parent two, P2) to generate offspring for the next population.

5.2) A random real value is generated between N(0,1) as the mating chance.

If (mating chance <= MATING RATE)

then Crossover P1 and P2 with extension combination operator [13][14] to create a new individual as offspring. Mutate new offspring.

else

skip step 5.3

EndIf

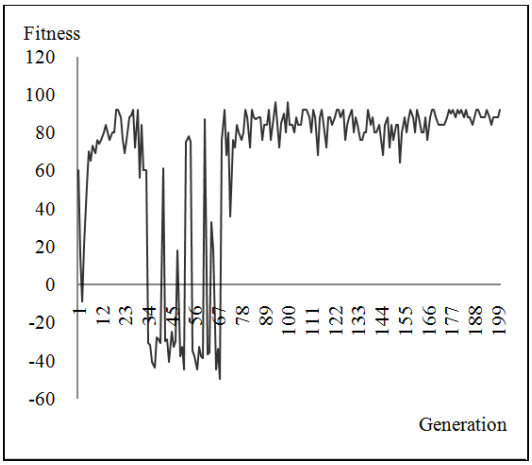
5.3) Add offspring into next population.

5.4) Step 5.1, 5.2 and 5.3 are repeated until all the offspring have been generated.

6.0) Step

Step 3.0, 4.0, and 5.0 are repeated until generation 200."

FFANN- maitinimo pirmyn dirbtinis neuroninis tinklas (angl. *Feed Forward Artificial Neural Network*);
Crossover – kryžminimas;
Offspring – palikuonis;
[13] - Nuoroda į D. Beasley, D. R. Bull, and R. R. Martin, (1993) "An overview of genetic algorithms: Part 1, fundamentals", Univ. Comput., vol. 15, pp. 58 – 69.
[14] – nuoroda į D. Beasley, D. R. Bull, and R. R. Martin,(1993) "An overview of genetic algorithms: Part 2, research topics", Univ. Comput., vol. 15, pp. 170 - 181.



2.6 pav. Rezultatai iš pirmojo eksperimento (Chang; Chin; Jason; Aroland, 2011:3)

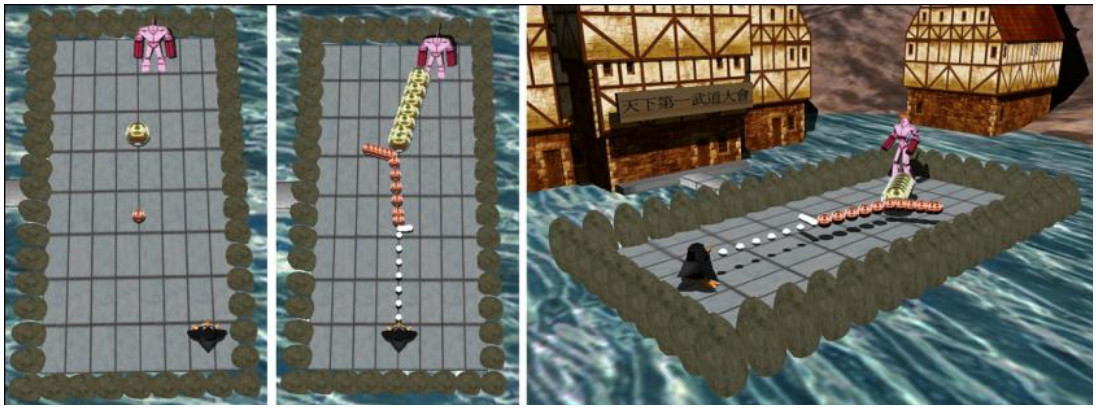
Fitness- tinkamumo funkcija, kuri ir apibrėžia rezultatų tinkamumą;
Generation – genetiniu algoritmo panaudotų generacijų skaičius;

Iš autorių pateiktų rezultatų matyti, kad apmokant neuroninį tinklą genetiniu algoritmu - pastovi tinkamumo funkcija nusistovi tik po maždaug 67 generacijų (2.6 pav.). Į tai reikia atkreipti dėmesį, nes generuojant viską atsitiktinai, pastovumas nusistovi tik po kiek laiko.

2.2.3 „Snowball Shooting" žaidimas

„Shoot Ball" tai studentų sukurtas žaidimo variklis, kuriame naudojamas neuroninio tinklas apmokomas Genetiniu Algoritmu (GA) [8]. Žaidimo variklio kontroleris atsakingas už sėkmingą kamuolio šūvį į oponentą. Šiam tikslui pasiekti neuroninis tinklas apmokomas naudojant neprižiūrimąją (angl. *unsupervised*) būdą.

Užduoties pasunkinimui buvo panaudoti du papildomi tarpiniai kamuoliai. Užduotis pakeista, taip kad visi trys kamuoliai turi pataikyti vienas į kitą eilės tvarka, ir kad būtent paskutinis kliudytų oponentą (2.7 pav.).



2.7 pav. Žaidimo kadrai, kur pirmas demonstruoja visus žaidimo objektus; sekantis demonstruoja iššautą sniego kamuolį ir jo poveikį kitiem kamuoliam; pakutintas - perspektyvą (SAI-Keung; Shih-Wei, 2012:5)

Neuroninio tinklo apmokymui panaudoti šeši įėjimai ir trys kampai (kiekvieno kamuolio judėjimo trajektorijos) ir *d* atstumas nuo veikėjo iki varinio rutulio. Atstumas *d* yra pagalbinė reikšmė stimuliuojanti pingvino veikėją judėti į šonus, o trys kampai naudojami šūvio trajektorijos apskaičiavimui.

Šis neuroninis tinklas turi tris išėjimus. Pirmasis išėjimas naudojamas apspręsti pingvino judėjimo kryptį (kairėn/dešinėn). Antrasis išėjimas naudojamas nusprendimui ar jau laikas pingvinui šauti sniego kamuolį į priešininką, ar dar ne. Paskutinis išėjimas naudojamas nuspręsti šaunamo kamuolio greičiui.

Tinkamumo funkcija „Ft“ (ankl. *fitness*) apskaičiuojama naudojant formulę [9]:

$$\mathbf{Ft = c(\theta_4)s(p1, p2, p3, p_{robot}, n_{ch}, ng)\varphi(dis(p3, p_{robot})),} \quad (1)$$

kur funkcija „s“ yra lygi:

$$s(p1, p2, p3, p_{robot}, n_{ch}, ng) = \omega(ng)(\alpha h(p1, p2) + \beta h(p2, p3)) + \lambda + \gamma h(p3, p_{robot}) + \kappa n_{ch}$$

p1, p2, p3 - kamuolių pozicijos plokštumoje (sniego, varinio ir auksinio),

p_{robot} - roboto pozicija,

$h(x, y)$ – nusako ar „x” pataikė į „y”,

n_{ch} – skaičiuoja iš eilės einančių pataikymų skaičių į oponento robotą,

λ – dydis skatinantis pingviną judėti į šonus,

ng – dydis nusakantis kelinta vykdoma apsimokymo generacija, pagal jo dydį nustatomas $\omega(ng)$, kuo generacijos numeris didesnis, tuo šio koeficiento įtaka yra mažinama, nes iš pradžių norima, kad pingvinas sėkmingai pataikytų į pirmąjį rutulį (bronzinį). Laikui prabėgus pingvinas tai sugeba atlikti gerai, todėl dėmesio kreipimas į „ar mes pataikėm į bronzinį rutulį, ar ne“ mažinamas, nes stengiamės išmokinti pataikyti bronzinį į auksinį,

ng – maksimalus generacijų skaičius,

α, β, γ – atspindi pataikymo į kamuolius įtakos svorius, kur $\alpha < \beta < \gamma$, nes pataikymas auksinio kamuolio į oponento robotą yra svarbiausias,

κ - visų svorių suma $\alpha + \beta + \gamma$,

θ_4 – kampas tarp auksinio kamuolio ir vektoriaus į oponento robotą,

$c(\theta_4)$ – tikslumo bonusas,

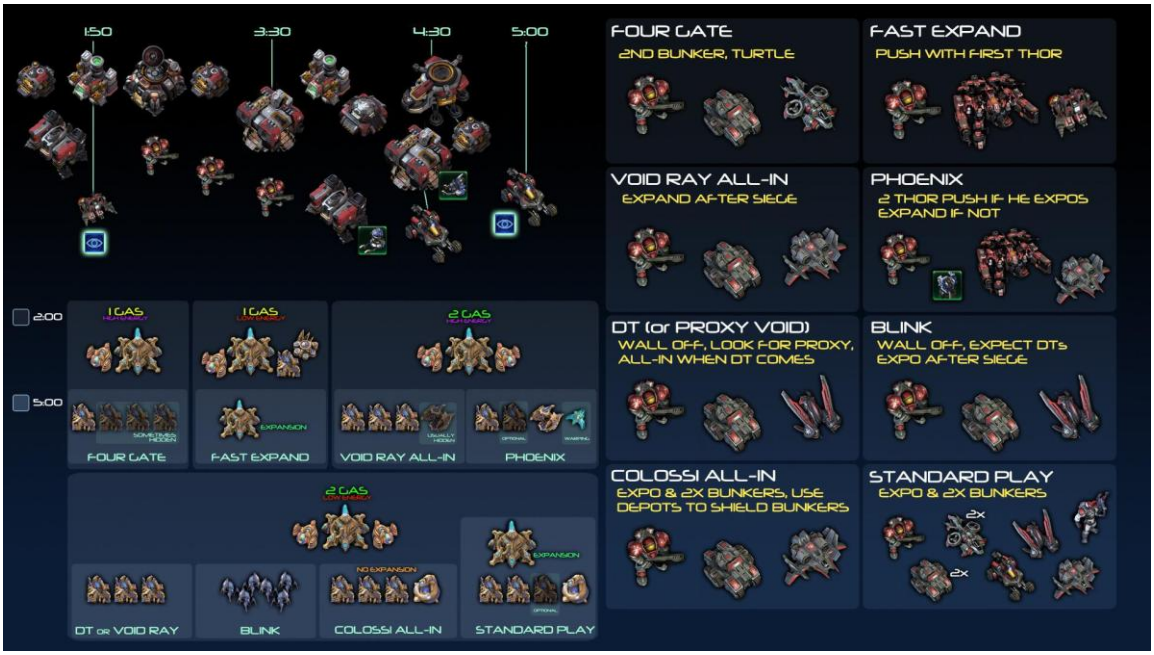
dist(p3, p_{robot}) –atstumas tarp p3 (auksinio kamuolio) ir p_{robot},

φ(dist(p3, p_{robot})) – lygi formai 1/dist(p3, p_{robot}).

Apmokyto neuroninio tinklo rezultatai stebina. Net 99% tikimybė, kad užduoties tikslas bus pasiektas. 1% prarandamas, kai oponento robotas stovi vienam iš aplinkos kampų. Tokiu atveju pingvinas apsimoko prasčiau naudojant anksčiau minėtą įtakos funkciją. Taip pat iš autorių gautų rezultatų pastebėta, kad tyrimo metu naudojant kintančią įtakos funkciją, kuri orientuojasi į nuoseklų tinklo apsimokymą, geri rezultatai pasiekiami kur kas greičiau.

2.2.4 „Starcraft II“ žaidimo dirbtinis intelektas

„Star Craft II“ šiuo metu yra vienas tarp populiariausių pasaulyje strateginių kompiuterinių žaidimų. Sukurti šiam žaidimui dirbtinį intelektą yra labai sunku, nes žaidimas apima makro ir mikro strategijų junginį (2.8pav).



2.8 pav. Žaidimo „Starcraft II“ skirtingos karo ir kareivių gaminimo strategijos

Kiekvienas žaidėjas pradeda žaidimą turėdamas bazę, kurioje gali gamintis paprastus kareivius. Žaidimo eigoje žaidėjai tobulina savo bazę: plėsdami resursų sistemą ir įsigydami geresnius kareivius. Žaidimo tikslas pašalinti savo priešininkų bazes.

Žaidimo idėja paremta skirtingais kareivių tipais, kur vieni sėkmingai muša antrus, antri – trečius, o treti pirmus, pvz.: oro pajėgos sėkmingai muša žemės pajėgas, o oro pajėgos neatsilaiko prieš karius, kurie skirti kautis tik su oro pajėgomis.

Makro strategija

Sukurti sėkmingai žaidžiantį ir taisyklėmis paremtą dirbtinį intelektą yra sunku. Dar sunkiau, jei jis atsakingas už visus strateginio žaidimo veiksmus: karių statybą, resursų planavimą, karo strategijas ir atskirų karių valdymą. Tai gali tapti beveik neįmanoma užduotimi, dėl per didelio kareivių pasirinkimo ir susidarančių kintamų kombinacijų skaičiaus. Tokiomis aplinkybėmis pasitarnauja statybų planuotojas apmokytas genetiniu algoritmu (2.9 pav).

2.9 pav. Žaidimo „Starcraft II“ planuotojo darbas

Statybų planuotojo darbas - išrinkti kuo geresnį spendimo variantą įvertinant kuo daugiau aplinkybių. Šiai užduočiai atlikti gali būti panaudoti prioritetai, kurie atkreips dėmesį į turimų resursų kiekį, susidariusia situaciją žemėlapyje, įvertinant priešininkų ir savo karių kiekį, tipą ir poziciją [9].

Mikro strategija

Mikro strategija – atsakinga už karių valdymą. Norint padidinti mikro strategijos efektyvumą, patartina naudoti kuo paprastesnius kareivius. Judėjimo krypties apsisprendimui palengvinti dažnai naudojami įtakos žemėlapiai. Keičiant potencialų reikšmes žemėlapyje, galima priversti karius judėti į norimą vietą, reaguoti į oponentų karių priartėjimą ir jų tipą. Dažnai potencialai gali padidinti ar sumažinti įtakos žemėlapio duomenis, priklausomai nuo sutiktų karių stiprumo.

Iš darbe patiektų rezultatų pastebėta, kad statybų planuotojas išsprendžia gana sunkią resursų valdymo problemą. Derinant makro ir mikro strategijas pasiekiami žymiai geresni dirbtinio intelekto rezultatai. Taip pat, autoriai mini apie statybų planuotojo darbo efektyvumo pagerėjimą, jei jis atsižvelgia į oponento naudojamą žaidimo strategiją.

2.2.5 Biblioteka LibGDX

Libgdx biblioteka pasirinkta todėl, kad būtų sumažinti programinės įrangos kūrimo kaštai. Bet tai nėra vienintelis Libgdx teigiamas aspektas. Šio darbo programos kodas ateityje gali būti pritaikomas ir kitose platformose. Tai suteiks programai universalumo ir lankstumo. Vartotojas nėra priverstinai pririšamas prie vienos operacinės sistemos.

Libgdx yra daugiaplatformė žaidimų kūrimo biblioteka parašyta Java kalba. Biblioteka apjungia Windows, Android, iOS ir HTML5 žaidimų kūrimo. Jos grafika yra pagrįsta standartais kaip OpenGL ES / WebGL. Bibliotekos pagalba kuriama programa pilnai Windows aplinkoje, ir 6 kodo eilučių pagalba, ji gali būti paleista ant ,,Android" ar HTML5.

Biblioteka pasirūpina dauguma žemo lygio dalykų, todėl galima sutelkti dėmesį į žaidimo sistemos, ar pagrindinio variklio programavimą ir nesirūpinti žemais dalykais, kaip OpenGL veikimo principas atvaizduojant 2D paveikslėlius, perspektyvos apskaičiavimu ir pan. Taip pat, šios bibliotekos naudojimas neužkerta kelio prieiti ir prie žemo lygio naudojimo. Biblioteka suteikia puikią terpę pradėti darbą [10].

Biblioteka turi kelis atsakingus moduliu:

- Grafikos modulis, kuris atvaizduoja duomenis į ekraną
- Modulis skirtas muzikos grojimui ir garso efektams

- Įvesties modulis skirtas gauti vartotojo įvestis iš: pelės, klaviatūros ar lietimui jautrus ekrano
- Failą I / O modulis skirtas skaityti ir rašyti duomenis

Biblioteka pasirinkta dėl trijų jos privalumų:

- nemokama ir atviro kodo;
- bibliotekoje realizuotas darbas su failais;
- ateityje programos kodą galima panaudoti įvairiose platformose (Windows, Android, iOS);

2.2.6 Genetinis algoritmas

Genetinis algoritmas (GA) - tai „evoliucionuojantis“ algoritmas paremtas biologijos žiniomis apie gyvybės evoliuciją. GA naudoja gamtoje egzistuojančius gyvybės evoliucinius mechanizmus: paveldėjimą, mutaciją, natūraliąją atranką ir kryžminimą.

GA metodas yra skirtas analizuoti duomenis, ir jo dėka galima rasti apytikslį užduoties sprendimą, naudojant evoliucinį ciklą, veikiantį kaip ir gamtoje. Genetinis algoritmas suteikią priemonę kompiuteryje atkartoti tam tikros populiacijos ir sprendimų kandidatų evoliuciją, artėjant prie geriausio užduoties sprendimo [6][11][13].

Bendradarbiaujančios evoliucijos technika (angl. *co-evoliution*) yra metodas, kuriame evoliucijos reikšmė priklauso nuo užduoties sunkumo skirtos kiekvienam agentui atskirai. Dėl šios priežasties kiekvienas agentas evoliucionuoja individualiai. Iš kitos pusės nebendradarbiaujančios evoliucijos technika (angl. *non-co-evoliution*) – metodas, kuriame visi agentai evoliucionuoja priklausomai nuo evoliucijos reikšmės priskirtos visai sistemai.

Detaliau Genetinio Algoritmo mechanizmą sudaro [12]:

Atranka

Iš populiacijos atrenkama dalis tinkamiausių sprendinių (sėkmingiausių „palikuonių“), iš kurių bus kuriama naujoji populiacija. Atrankos kriterijų aprašo tinkamumo funkcija (angl. *fitness*), kuri ir lemia atranką. Dažniausiai taikomas metodas, kai iš visos populiacijos atrenkami labiausiai tinkami sprendiniai. Tačiau yra ir kitas atrankos metodas, kurio metu yra vertinami tik atsitiktinai populiacijos sprendiniai. Antrasis metodas, yra mažiau efektyvus nei pirmasis, kadangi šiuo metodu tiksliausio sprendimo paieška trunka žymiai ilgiau.

Dauguma atrankos funkcijų yra tikimybinės ir sukurtos taip, kad atrinkimo metu, nedidelė atrinktų spendimų dalis būtų paimti burtų keliu. Tokiu būdu užtikrinama, kad išliktų įvairovė. Dažniausi ir daugiausiai išstudijuoti atrankos metodai: ruletės ir turnyrinė atranka.

Mutacija ir kryžminimas

Antrosios ir kitų populiacijų kūrimą sudaro genetinių mechanizmų: kryžminimo ir mutacijos – pritaikymas. Kryžminimas ir mutacija yra biologinių mechanizmų analogai.

Kiekvieno naujo sprendinio sukūrimui iš populiacijos imama pora sprendinių „tėvų“, kurie sukryžminami ir mutacijos ar kryžminimo būdu gaunamas sprendinys „vaikas“. Toliau imama kita „tėvų“ pora ir vėl sukuriamas „vaikas“ ir taip tęsiama kol pasiekiamas tam tikras naujos kartos individų skaičius, sukuriama pilna nauja sprendinių populiacija.

Galiausiai po šių procesų naująsias kartas sudarys individai, kurių chromosomos (sandara) yra visiškai skirtinga nei pradinės populiacijos. Bendras populiacijos tinkamumo vidurkis taip pat pakils, kadangi išliks tik geriausi individai (sprendiniai) ir dalis mažiau tinkamų individų.

Mutacija

Mutacija yra vienas iš būdų, kaip išlaikyti genetinę įvairovę genetiniuose algoritmuose (2.10 pav.). Kadangi jos mechanizmas buvo kuriamas pagal biologinę mutaciją. Mutacija padeda išvengti populiacijos chromosomų supanašėjimo, nes tai vestų į evoliucijos sulėtėjimą. Šis reiškinys, taip pat paaiškina, kodėl GA sistemos vengia naudoti tokį atrinkimo būdą, kai atrenkami vien tik geriausi tėvai, nes po kiek laiko tėvai visados kartosis .

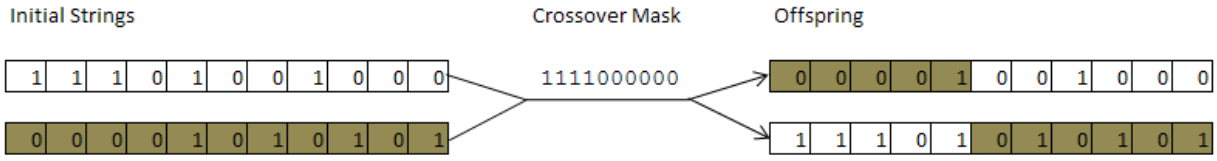
2.10 pav. Taško mutacijos pavyzdys (Chong-U Lim, 2009:33)

Paveikslėlyje pavaizduotas mutacijos procesas naudojant vieną tašką, kur:
Initial String – pradiniai duomenys;
Offspring – po mutacijos gauti duomenys;

Klasikinį mutacijų operatoriaus veikimą sudaro atsitiktinai generuojami skaičiai, kurie nurodo įvyks ar neįvyks mutaciją tam tikrai duomenų daliai.

Kryžminimas

Kryžminimas yra genetiniuose algoritmuose naudojamas būdas perteikti vienos chromosomos ar kelių chromosomų sandarą į kitą kartą. Jis yra biologinės kryžminimo analogas, nes yra paremtas būtent biologiniu procesu.



2.11 pav. Vieno taško kryžminimo pavyzdys (Chong-U Lim, 2009:32)

čia,
Initial String – pradiniai duomenys;
Crossover Mask – kryžminimo kaukė;
Offspring – po kryžminimo gauti duomenys;

Paveikslėlyje pavaizduotas kryžminimo procesas naudojant vieną perskyrimo tašką, kur vieno tėvo duomenų dalis sukeičiama su kito tėvo. Duomenys yra sukeičiami būtent ten, kur nurodo kryžminimo kaukė.

Yra nemažai kryžminimo būdų, kurie skirtingai perteikia biologinės kryžminimo esmę:

Vienoda ir pusiau vienodas Kryžminimas. Abiejų technikų atvejais „tėvų“ duomenų pagrindu sukuriami du nauji „vaikai“. Pirmuoju atveju „tėvų“ duomenys sulyginami tarpusavyje ir sukeičiami vietomis su 50 proc. tikimybe. Pusiau vienodos perskyrimo atveju sukeičiama vietomis pusė nesutampančių tėvų duomenų.

Su vienu perskyrimo tašku. Pasirenkamas perskyrimo taškas tėvų duomenyse. Vėliau visi „tėvų“ duomenys, buvę už taško, apkeičiami vietomis. Rezultatas – „vaikai“, kurie turi dalį vieno ir kito tėvo (genetinių) duomenų.

Su dviem perskyrimo taškais. Pasirenkami du perskyrimo taškai tėvų duomenyse. Vėliau visi „tėvų“ duomenys, esantys tarp šių taškų sukeičiami vietomis. Rezultatas – „vaikai“, kurie turi dalį vieno ir kito tėvo duomenų

„Pjaustymas“ – kryžminimo technika, kai keičiamas „vaikų“ duomenų ilgis. Skirtumas nuo anksčiau minėtų technikų yra tas, kad čia tėvų duomenyse pasirenkami taškai skirtingose vietose.

Taikant „pilną“ Genetinį Algoritmą, galima gerai įvertinti ir analizuoti duomenis ir jo dėka pakankamai greitai išeina rasti apytikslį užduoties sprendimą.

Tinkamumo funkcija

Genetinis algoritmas gali tik optimizuoti tinkamumo funkcijos charakteristikas. Pagal tinkamumo funkcijos reikšmes, generuojamos sekančios kartos, t.y., atrenkami individai dauginimuisi ir palikuonių generavimui. Populiacijos dydį, kryžminimo ir mutacijos apimtis galima priskirti vidinei genomų evoliucijai, dauginimasis ir tinkamumo parinkimo technika priskiriami išorinei atrankai.

$$F(x1,x2) = W - L, \qquad (1)$$

čia:

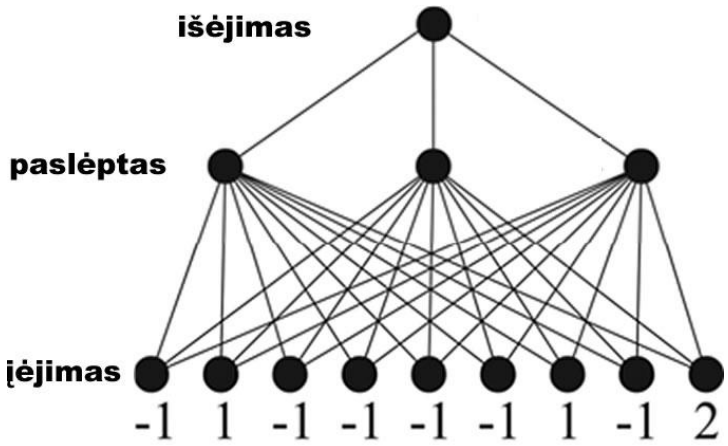
F – tinkamumo funkcija, W – laimėjimų įtaka, L – parlaimėjimų įtaka [8].

Vidinė evoliucija stebima keičiant parametrus: populiacijos, kryžminimo, mutacijos dydžius. Išorinė atranka – dauginimosi ir tinkamumo parinkimo, kuriuos įtakoja funkcijos svoriai.

2.2.7 Neuroninis Tinklas

Neuroninis tinklas (NT) - naudoja paprastą struktūrą skirtą informacijos apdorojimui ir šablonų atpažinimui. Pasitelkiant Genetiniu Algoritmu apmokomą neuroninį tinklą, dažnai galima pasiekti gerų sprendimo paieškos rezultatų.

Dažniausiai NT vidiniai skaičiavimai yra pritaikyti konkrečioms problemoms spręsti. Kad NT galėtų priimti logiškus sprendimus, jam privalo gauti duomenų, kuriuos galėtų įvertinti ir kuriais besiremdamas gražintų „išvadas“. Neuroninio tinklo struktūra yra sudaryta iš: įvesties, išvesties ir paslėptą. Įvertinęs duomenis NT grąžina išvestį.



2.12 pav. Neuroninio Tinklo struktūra

NT yra apmokomas naudojant Genetinį algoritmą, kuris padeda ,,optimaliau parinkti ir įvertinti“ įėjimo duomenis. Paslėptajame sluoksnyje atliekami visi įvesties apdorojimo ir vertinimo procesai, kurių pagalba suformuojamas trečiasis sluoksnis – išvestis

NT yra sudarytas iš daugybės neuronų. Visi neuronai tarpusavyje yra susieti (koeficientais) ir įvairiomis charakteristikomis. Kaip ir žmogaus smegenys, NT stengiasi išmokti svorių reikšmes. Susiejimo koeficientai yra keičiami atliekant NT apmokymo procesą.

2.2.8 Dijkstros algoritmas

Trumpiausio kelio radimui nesvoriniame grafe pakanka naudoti paiešką į plotį arba paiešką į gylį. Svoriniame grafe dažniausiai naudojamas Dijkstros algoritmas. Dijkstros algoritmas naudojamas trumpiausio kelio radimui tarp dviejų viršūnių kryptiniame svoriniame grafe kur briaunos svoris yra teigiamas skaičius. Šis algoritmas dažnai naudojamas žaidimuose, kurie turi labai paprastą žemėlapio struktūrą.

Be grafo informacijos, tokios kaip briaunų koeficientai ir viršūnių ryšiai algoritmas papildomai skaičiavimuose naudoja neaplankytų viršūnių sąrašą U, atstumų nuo starto viršūnės S iki kiekvienos viršūnes masyvą D ir rodyklių į ankstesnes viršūnes masyvą P. Iš pradžių nustatoma kad visos viršūnės yra neaplankytos, atstumai yra begaliniai, o P viršūnes pažymimos kaip nenustatytos (tuščios). S viršūnės atstumas nustatomas į 0. Dijkstros algoritmo pseudo kodas:

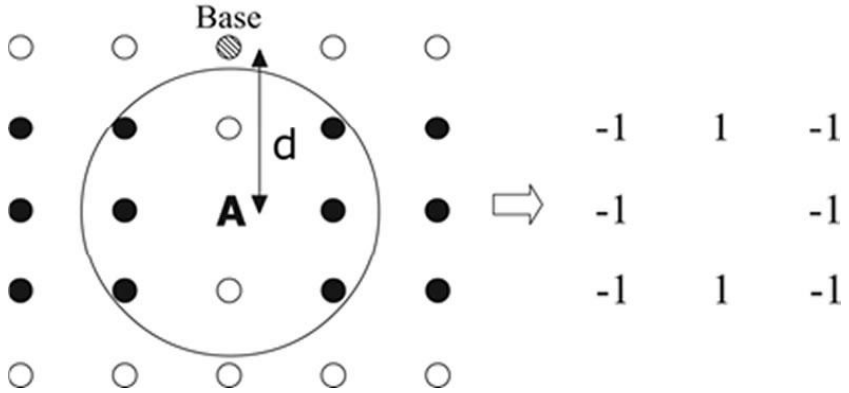
```
Dijkstros algoritmas
// n – grafo viršūnių skaičius
// svoriai[n] – briaunų svorių masyvas
// D[n] – atstumų nuo kiekvienos viršūnės i starto viršūnės masyvas
// U[n] – neaplankytų viršūnių masyvas
// P[n] – rodyklių į ankstenses viršūnes masyvas
// S – starto viršūnė
// F – tikslo viršūnė
// V – einamoji grafo viršūnė
// V2 – gretima neaplankyta grafo viršūnė

for i = 0 to n
   D[i] = begalybė;
   P[i] = tuščia;
   U[i] = neaplankyta;
endfor;
D[S] = 0;

while U turi neaplanytų viršūnių
   V = viršūnė iš neaplanytų su mažiausiu atstumu;
   U[V] = aplankyta;
   for V2 = gretima neaplankyta viršūnė V viršūnei
      if (D[V] + svoriai(V, V2) < D[V2])
         D[V2] = D[V] + svoriai(V, V2); // pažymime naują mažesnį atstumą
         P[V2] = V; // pažymime, kad atėjome iš V
      endif;
   endfor;
endwhile;
```

Kai visos viršūnės tampa aplankytos iš P masyvo galima atsekti trumpiausią kelią nuo bet kurios grafo viršūnės iki starto viršūnės.

Įtakos Žemėlapiai (ĮŽ)

Įtakos žemėlapiai dažniausiai aprašomi naudojant matricas, kur kiekviena celė atspindi žemėlapio langelį ar net visą bloką. Celėse saugomos reikšmės įtakoja įvairiausius veiksmus: nuo strategijos pasirinkimo iki konkrečios judėjimo vietos išrinkimo.

Dažnai įtakos žemėlapiai sujungiami kartu, tam kad suformuotų erdvinę sprendimų priėmimo strategiją (2.5 pav.).



2.13 pav. Pasiekiamumo įtakos žemėlapio sudarymas

Pagal viršuje esantį paveikslėlį, 8 taškai esantys aplink tašką A yra užkoduojami. „-1“ reikšmės reprezentuoja barjerus, žemėlapyje nuspalvintus juodai. „1“ atspindi pasiekiamus laisvus langelius atstumu „d“ nuo taško „ A“ [4].

Panaudojus įtakos žemėlapius lengviau įvertinti žemėlapyje susidariusią situaciją, įvertinti oponento armiją ir išsirinkti optimalius taikinius.

2.3 Analizės išvados

Iš analizuotų tyrimų pastebėta, kad didžiojoje dalyje analizuotų tyrimų, naudojami įtakos žemėlapiai, kurie palengvina karių grupinį darbą žaidimo žemėlapyje. Genetiškai apmokytas neuroninis tinklas leidžia įvertinti ir pateikti sprendimus, kur tradiciniai paieškos metodai ir optimizacijos yra per lėtos. Iš Hirotaka Itoh, Naoki Ikeda ir Kenji Funahashi tyrimo pastebima, kad bendradarbiaujanti evoliucija padeda greičiau apsimokyti neuroninius tinklus.

Remiantis šiais metodais bus sudaryta ėjimais paremto žaidimo sistema. Atlikus analizę buvo išskirti keli siūlomi problemos sprendimą lengvinantys metodai: įtakos žemėlapiai - palengvinantys žemėlapyje susidariusios situacijos įvertinimą; apmokytas neuroninis tinklas; bendradarbiaujanti evoliucija, kuri pagreitins NT apmokymą.

Analizės dalyje buvo apžvelgta keletas dirbtinio intelekto metodų. Žaidimo sistemai pritaikyti buvo pasirinktas GA apmokomas NT. Lyginant su kitais metodais buvo išskirti šie NT privalumai: paprastas formalus aprašymas sumažinantis klaidingo interpretavimo tikimybę, sėkmingas panaudojimas daugelyje skirtingų sričių, greitas veikimas, nors prieš tai sugaištamas daug laiko jo apmokymui.

Įtakos žemėlapiai buvo pasirinkti, dėl: galimybės lengvai kaupti susiklostančių situacijų duomenis ir paprasto ĮŽ naudojimo.

3 PROJEKTAVIMAS IR REALIZAVIMAS

Šiam skyriuje yra aptarta kas buvo suprojektuota ir realizuota norint įvykdyti užsibrėžtus tikslus.

3.1 Reikalavimai

3.1.1 Funkciniai sistemos reikalavimai:

- Turi būti realizuoti šie metodai:
    1 įtakos žemėlapiai
    2 genetinis algoritmas
    3 neuroninis tinklas
    4 antikūnių gamybos planuotojas
- Realizuota dirbtino intelekto panaudojimo aplinka – dalis ėjimais paremto žaidimo sistemos
- Žaidimo dirbtinis intelektas turi interpretuoti žaidimo ribas taip, kad sugebėtų atlikti žaidėjo pagrindinius veiksmus:
    1 rasti kelia iki oponento bazės;
    2 apginti savo bazę;
    3 užimti strategiškai patogias pozicijas;
    4 įveikti kuo daugiau oponento karių;
    5 gaminti papildomus karius;
- Turi pateikti statistinę informaciją apie metodų veikimą:
    1 GA apmokymo rezultatus (tinkamumo funkcijos ir svorių)
    2 laimėtų ir pralaimėtų žaidimo varžybų skaičius

3.1.2 Nefunkciniai reikalavimai

Reikalavimai aparatūrai ir sistemos veikimui:

- Turi veikti Windows operacinėje sistemoje
- Turi naudoti atviro kodo bibliotekas
- Turi naudoti nemokamus metodus

3.2 Bendros Sistemos Projektavimas

Kuriama sistema susideda iš susijusių blokų:

- Ėjimais paremto žaidimo sistemos. Šioje sistemoje, yra atliekami visi dirbtinio intelekto testavimai, dirbtiniam intelektui griežtai laikantis žaidimo taisyklių;
- Neuroninio tinklo projektavimo;
- Įtakos žemėlapių struktūros;
- Dirbtino intelekto apmokymas naudojant GA;

3.3 Sprendimo projektavimas

Programinė įrangos kūrimo priemonės

Kuriant sistemą buvo naudojama Java programavimo kalba. Ši kalba pasirinkta tam, kad būtų sumažinti programinės įrangos kūrimo laiko kaštai. Nes Java kūrėjai nemokamai leidžia naudotis šios kalbos kompiliatoriumi ir transliatoriumi. Java platforma veikia ir kitose operacinėse sistemose. Todėl šio darbo programos kodas ateityje gali būti pritaikomas tiek Windows, Linux, Android ar kitai operacinei sistemai. Tai suteiks programai universalumo ir lankstumo.

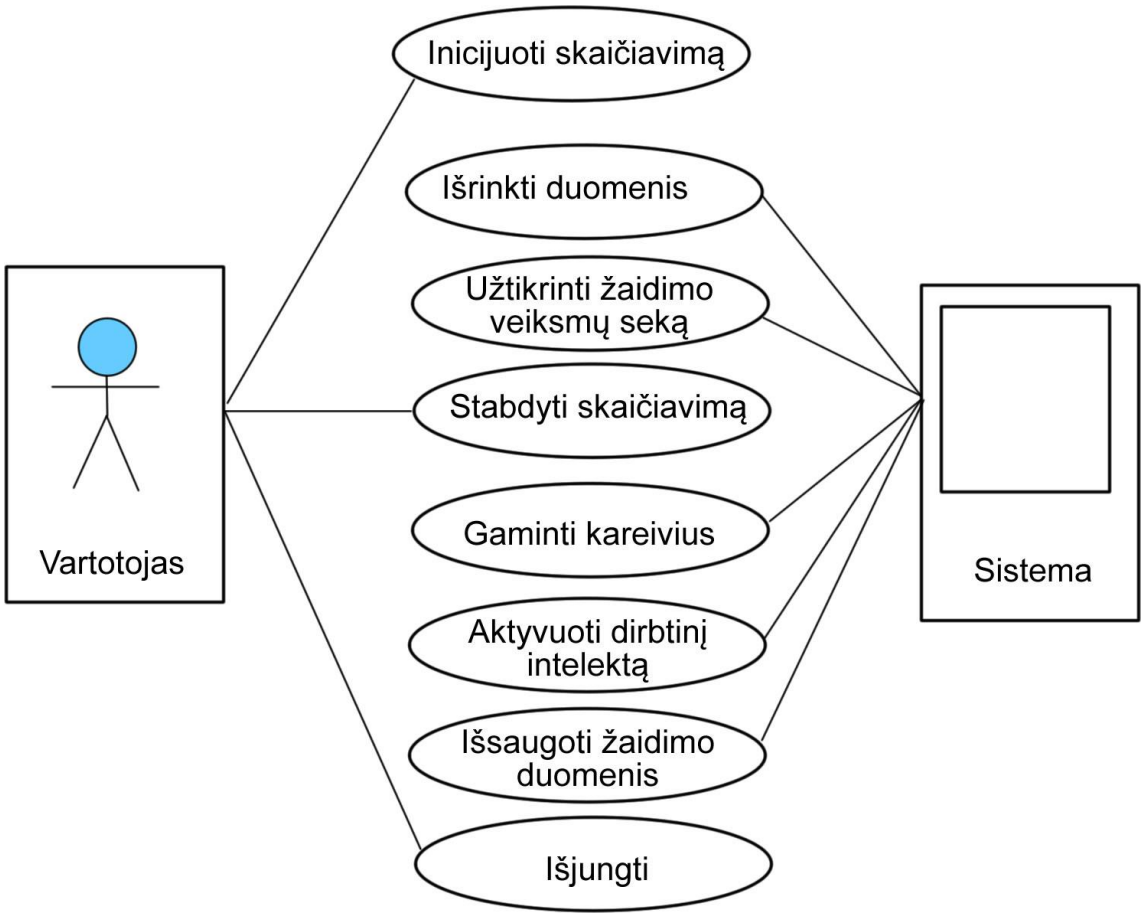
Kuriant dvimačius ar trimačius vaizdus generuojančią programinę įrangą taip pat labai svarbu pasirinkti tinkamą trimatės grafikos apdorojimo standartą. Prieš atliekant praktinę šio darbo dalį buvo palyginti trys grafiniai standartai: Java 3D, OpenGL ir LibGDX biblioteka. Bandymu metu paaiškėjo, kad programinė įranga parašyta panaudojant Java 3D bibliotekas reikalauja žymiai didesnių kompiuterio resursų, nei panaudojant OpenGL ar LibGDX bibliotekas. LibGDX biblioteka pasirūpina dauguma žemo lygio dalykų, todėl galima sutelkti dėmesį į žaidimo, ar pagrindinio variklio sistemos.

Trumpai apžvelgsime LibGDX – atvirą grafinę biblioteką. Programos, sukurtos panaudojant šią biblioteką, taip pat nėra pririšamos prie konkrečios platformos. .

LibGDX pagrindinės savybės:

- sparta
- realizuotas atvaizdavimo scenos grafas
- darbas su tekstūromis

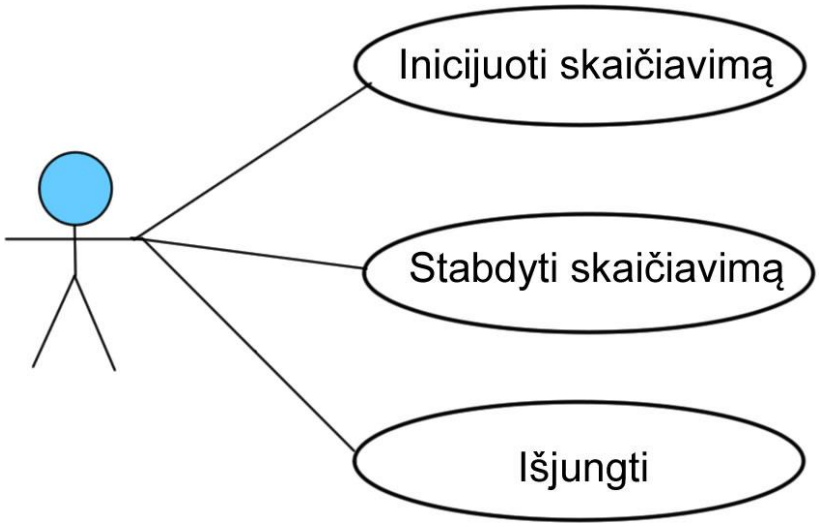
Sistemos ir vartotojo galimybės



3.1 pav. Panaudojimo atvejai

Panaudojimo atvejai

Panaudojimo atvejų diagrama



3.2 pav. Panaudojimo atvejų diagrama

Aprašoma pagrindinė informacija apie „GameScreen“ klasę, kuri atsakinga už sistemos duomenų pasiėmimą, skaičiavimus ir nereikalingų resursų atlaisvinimą.

Sistemos darbo tikslai – Pradinių duomenų nuskaitymas ir programos išvedimas į darbinį režimą. Užbaigus skaičiavimus, atlikti resursų atlaisvinimą.

Pagrindinai sistemos metodai:

- **Dispose**() – skirtas atlaisvinti pasiimtus programos darbui reikalingus resursus.
- **Update**() – skirtas kintančių duomenų kitimui užtikrinti .
- **Resize**() – skirtas programos lando dydžio kitimui.
- **Render** () – skirtas programos aplinkos atvaizdavimui.
- **GameScreen**(MoreGame) – konstruktorius, skirtas generuoti programos aplinką, bei apibendrinti ir paleisti programą.
- **Load**() – skirtas programos visiem duomenims iš failų užkrauti, tai yra tekstūros, šriftams, žemėlapio ir kitų objektų užkrovimui .
- **Init**() – skirtas programos kintamiesiems priskirti išankstines reikšmes .
- **ButtonTouches**() – skirtas programos vartotojo sąsajai užtikrinti.

3.4 Ėjimais paremto žaidimo sistema

„Warlike“ ėjimais-paremtas kortų žaidimas yra žaidžiamas ant 5x5 langelių dydžio matricos lauko. Šiam strateginio žanro žaidime, kiekvienas žaidėjas turi po asmeninę kortų kaladę. Žaidimo metu, kiekviena žaidėjo panaudota korta tampa kareiviu.

Žaidimo tikslas: pereiti visą žemėlapio matricą į priešininko pusę ir padaryti oponento bazei 5 žalos.

Žaidimą sudaro: žaidimo lenta (5x5) ir dvi 20-30 asmeninės žaidėjų kortų kaladės.

3.3 pav. Žaidimo kortos – kario pavyzdys

Žaidime naudojamos penkios kareivių rūšys. Tarpusavyje visi kareiviai skiriasi rodikliais: kaina, puolimo galia, gyvybių kiekiu, judėjimo greičiu ir specialiaisiais įgūdžiais.

3.1 lentelė Kareivių tipai

Kario NR.	Kaina	Puolimas galia	Gyvybės	Judėjimo Greitis	„FA“ Puolimas	Dvigubas Puolimas
1	3	2	1	2	+	-
2	3	1	3	2	-	-
3	2	3	1	2	-	-
4	1	1	1	3	-	-
5	2	1	1	2	-	+

Iš kareivių tipų lentelės matyti, kad kuo kareiviai brangesni – tuo jų turimos statistiko yra didesnės. Atidžiau pasižiūrėjus į kareivių puolimo ir gyvybių santykius matyti, kad jie gali būti naudojami skirtingoms užduotims atlikti. Šia prielaida remiasi gamybos planuotojo metodas.

3.2 lentelė Žaidimo karo laukas (5x5)

„222“ žaidėjo Bazė				
„111“ žaidėjo Bazė				

Nupirkti nauji kareiviai, žaidimą pradeda viename iš penkių šalia bazės esančių laukelių. Nuo sekančio ėjimo (pagal žaidimo taisykles), kareiviai gali vaikščioti žaidimo lenta stengdamiesi pasiekti gautus tikslus ir vykdyti savo strategijas. Strategijų pasirinkimas dažniausiai priklauso nuo aplinkos situacijų.

Ėjimais-paremto žaidimo nuosekliai einančių veiksmų seka:

1) Sekančio žaidėjo ėjimo pradžia
2) Kareivių paruošimas (atsigavimas)
3) Naujų kortų traukimas
4) Karių judėjimas
5) Naujų karių pirkimas
6) Pinigų gamyklų statymas
7) Žalos priskyrimas priešininko bazei
8) Ėjimo pabaiga

Pagrindinės žaidimo taisyklės:

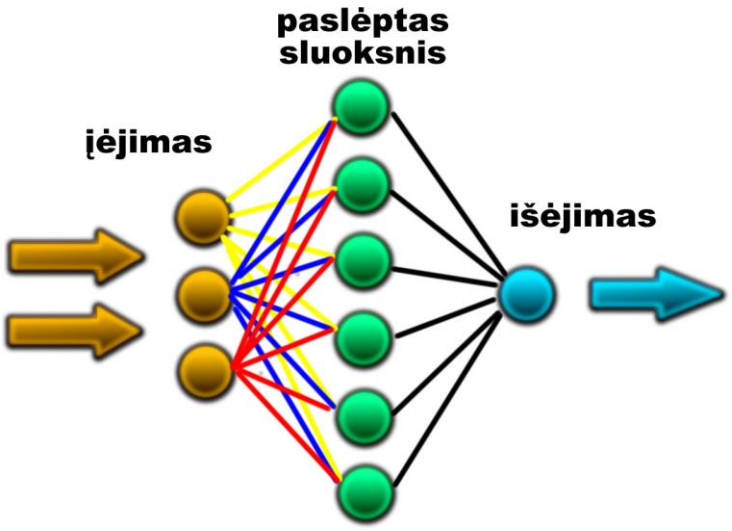
- Žaidimas žaidžiamas ėjimais, tai - visus savo veiksmus žaidėjui atliekant iš eilės ir pabaigus užleidžiant eilę sekančiam žaidėjui
- Pradžioje žaidimo visi žaidėjai gauna po 7 kortas
- Kiekvieno ėjimo gale, žaidėjas gali atsisakyti vienos iš savo rankose laikomų kortų ir pasistatyti papildomą pinigų gamyklą
- Kareiviai negali vaikščioti kiaurai oponento karius
- Žemėlapyje vaikštoma tik vertikalia ir horizontalia kryptimi
- Kovos metu sužeisti kareiviai ir lieka sužeisti iki žaidimo galo
- Kareiviai į žaidimą nuperkami pavargę ir negali judėti iki sekančio ėjimo

3.5 Neuroninio Tinklo projektavimas

Kuriamas neuroninio tinklo modelis, rėmėsi šabloniška NT schema, kurios vidiniai skaičiavimai pritaikyti konkretaus ėjimais paremto žaidimo kareivių judėjimo optimalios pozicijos įvertinimui. Dirbtinis intelektas priimdamas sprendimus naudoja neuroninį tinklą. Kad dirbtinis intelektas galėtų priimti logiškus sprendimus, neuroninis tinklas privalo gauti duomenis, kuriuos galėtų įvertinti ir kuriais besiremdamas padarytų ,,išvadas“. Tam neuroniniam tinklui paduodama įvestis, kurią sudaro duomenys apie supančią aplinką. Įvertinęs duomenis NT grąžina išvestį.

3.5.1 Panaudotas Neuroninis Tinklas

Realizuoto neuroninio tinklo struktūra yra sudaryta iš: įėjimo, paslėptojo, išėjimo sluoksnių ir šešių koeficientų. Apmokytas genetiniu algoritmu neuroninis tinklas, padeda optimaliau parinkti ir įvertinti įėjimo duomenis. Paslėptajame sluoksnyje atliekami visi įvesties apdorojimo ir vertinimo procesai, kurių pagalba suformuojamas trečiasis sluoksnis – išvestis.



3.4 pav. Neuroninio tinklo schema

Paveikslėlyje pavaizduotas NT schema, kuri įėjimui naudoja pasirašytą įvesties vektorių. O išvesčiai naudojamas išvesties vektorius.

Realizuoto NT struktūra ir jame vykstantys skaičiavimo procesai, sudaro pasirinkto žaidimo dirbtinio intelekto pagrindą. Įvertinus neuroniniam tinklui išvesties duomenis, galima rasti optimalių problemos sprendinį.

3.5.2 NT išvesties formavimas

Neuroninio tinklo išvestys apskaičiuojamos ,,paslėptajam sluoksnyje“. Prieš paduodant duomenų vektorių į neuroninį tinklą, yra suformuojamas duomenų vektorius. Paduodamų duomenų vektoriaus formavimo skaičiavimai padalinti į dvi grupes: duomenų apdorojimo skaičiavimus ir kareivį supančių duomenų atrinkimą ir suglaudinimą.

Paslėptajame sluoksnyje atliekami paduotų duomenų vektoriaus vertinimas ir suformuojama išvestis. Naudojant neuroninio tinklo išvesties duomenis, atliekamas tikimybių įvertimą ir kareivio judėjimo taško parinkimas.

Duomenų apdorojimo skaičiavimai:

Prieš paduodant duomenų vektorių NT, duomenys yra apdorojami panašiu principu kaip formuojamos histogramos. Duomenų vektorius formatas yra taikinių duomenų struktūra. Apdorojant duomenų vektorių naudojamas tik kiekvienos celės įverčio dydis. Visi įverčio dydžiai sutraukiami į ,,vertikalų“ ir horizontalų“ vektorius.

Apdorojimas atliekamas pasitelkiant sistemoje realizuotus ,,getHistogramH“ ir ,,getHistogramV“ metodus.

Kareivį supančių duomenų atrinkimas ir suglaudinimas:

Šiame žingsnyje atliekamas duomenų, esančių apink kareivio koordinates atrinkimas iš prieš tai suformuotų vertikalių ir horizontalių duomenų vektorių. Atrinkti duomenys yra sutraukiami į šešis kintamuosius, kurie sistemoje užvadinti: A, B, C, a, b, c ir surašomi į įėjimo duomenų vektorių.

Įėjimo vektoriaus vertinimas:

Prieš vertindamas paduotą įėjimų vektoriaus rezultatus, NT įvertina kiekvieno briaunų koeficientą. Tada, pagal apsirašytą taikomą funkciją, apskaičiuojamas išėjimų vektorius.

Pseudo kodas skirtas išėjimų vektoriaus formavimui:

```
Išėjimų vektoriaus formavimas
aroundInfo[0] – „A”, kuris atsakingas už kareivio viršaus duomenų pasiskirstymą;
aroundInfo[1] –„B”, kuris atsakingas už kareivio eilės duomenų pasiskirstymą;
aroundInfo[2] –„C”, kuris atsakingas už kareivio apačios duomenų pasiskirstymą;;
aroundInfo[3] – „a”, kuris atsakingas už kareivio kairės pusės duomenų pasiskirstymą;
aroundInfo[4] –„b”, kuris atsakingas už kareivio esamo stulpelio pusės duomenų pasiskirstymą;
aroundInfo[5] –„c”, kuris atsakingas už kareivio dešinės pusės duomenų  pasiskirstymą;
PLeft – pirmas išėjimas
PRight - antras išėjimas
PStayRL  - tracias  išėjimas
PTop  - ketvirtas išėjimas
PDown  - penktas išėjimas
PStayTD  - šeštas  išėjimas

 PLeft = aroundInfo[3]-aroundInfo[4];
PRight =  aroundInfo[5]-aroundInfo[4];
PStayRL =  (aroundInfo[4] - aroundInfo[3]) + (aroundInfo[4] - aroundInfo[5]);
PTop = aroundInfo[0]-aroundInfo[1];
PDown = aroundInfo[2]-aroundInfo[1];
PStayTD = (aroundInfo[1] - aroundInfo[0]) + (aroundInfo[1] - aroundInfo[2]);
```

Kareivio judėjimo taško parinkimas

Šiam žingsnyje apskaičiuojamas problemos spendimas, į kurį lentos tašką perkelti pasirinktą kareivį. Siekiant sumažinti nenaudingų veiksmų pasirinkimo galimybę, apskaičiuojamas ir taikomas NT išėjimų duodamo vektoriaus tikimybių vidurkis. Tada, burtų keliu išrenkama optimali judėjimo kryptis, kurios pasirinkimo tikimybės priklauso nuo agento įvertintų NT išvesčių rezultatų. Ši kryptis talpinama į galimų taikinių struktūros sąrašą. Vėl tikrinama: ar dar yra optimalių ėjimų remiantis išėjimų vektoriaus įvertintais duomenimis. Jei yra, jie taip pat talpinami į išvesties duomenų sąrašą.

Taikinių sąrašas formuojamas taip, kad „geriausias“, tai – pirmas sprendimas būtų sąrašo priekyje, o prasčiausias gale. Tokiu būdu, sistemoje tikrinant ar kareivis sugebės pasiekti savo taikinį, tikrinimas bus pradėtas nuo geresnių sprendimo variantų.

3.6 Dirbtinio intelekto mikro ir makro strategijos

Šiame darbe stengiamasi suprojektuoti žaidimo sistemą ir ištirti joje taikomų dirbtinio intelekto metodus. Pagal žaidimų teoriją, dirbtinis intelektas turi pasižymėti strategijos laikymųsi.

Dirbtinio intelekto mikro ir makro strategijos:

1. Pasiekti oponento bazę ir padaryti 5 žalos
2. Apginti savo bazę, surenkant kuo mažiau žalos (nepasiekiant 5)
3. Užimti strategiškai svarbius taškus
4. Užtikrinti nuolatinį žaidimo karo aprūpinimą naujais kariais

Pirmi trys punktai yra mikro strategijos, kurios susijusios su kareivių judėjimu žemėlayje. Ketvirtoji strategija yra makro. Ji atsakinga už bendra karo eigą, kurią sudaro: resursų disponavimas ir jų plėtimas žaidimui progresuojant; karių kiekio užtikrinimas karo lauke. Makro strategija turi apsispręsti kiek reikės pinigų gamyklų, kad galėtų užtikrinti ne tik pastovią bazės gynybą, bet ir oponento bazės apgultį.

3.7 Neuroninio tinklo apmokymas naudojant GA

Bendroje žaidimo sistemoje vienas iš dirbtinį intelektą metodų naudoja neuroninio tinklo apmokymą genetiniu algoritmu.

GA realizuotas taip, kad panaudotų daugumą į evoliucijos procesą įeinančių veiksmų: kryžminimą, mutaciją ir atranką, kuri naudojasi tinkamumo funkcija. Būtent genetinio algoritmo pagalba apmokytas NT, sugebės atlikti sprendimus, kurie priartėtų prie pergalės savo suprojektuotoje ėjimais paremto žaidimo sistemoje.

Genetinis algoritmas pritaikomas kiekvieną kartą, po susitarto sužaistų žaidimo partijų skaičiaus. Šeši neuroninio tinklo įtakos koeficientai panaudojami kaip GA kintamieji.

Pradžioje, GA pritaiko tinkamumo funkciją (angl. *fitness*), tam kad įvertintų dirbtiniu intelektu valdomo žaidėjo tinkamumą. Tinkamumas vertinamas pagal agento asmeniškai pasiektus nuopelnus žaidime. Įvertinus tinkamumą, vykdoma elitų atranka naudojant turnyro struktūrą. Turnyro struktūra iš visų galimų agentų atrenka turinčiu didžiausią tinkamumo funkc. reikšmę. Šiems žaidėjams pritaikomas tarpusavio kryžminimas ir mutacija, naudojant sugeneruotas kryžminimo ir mutacijos kaukes. To pasėkoje, gaunamas visiškai naujas agentas, tikintis kad jis yra geresnis už pirmtakus savo tėvus.

3.7.1 Panaudota tinkamumo funkcija

Realizuotas genetinis algoritmas evoliucijos principu stengiasi optimizuoti tinkamumo funkcijos charakteristikas. Dauginimasis ir tinkamumo parinkimo technika yra priskiriami išorinei atrankai. Pagal gautas tinkamumo funkcijos reikšmes - generuojamos sekančios kartos, t.y., atrenkami individai dauginimuisi ir palikuonių generavimui.

Apsirašytai tinkamumo funkcijai įtakos turi: laimėtų ir pralaimėtų žaidimo partijų skaičius. Taip pat, funkcijos teigiama reikšmė didėja nuo nužudytų oponentų kareivių skaičiaus, sužaistos partijos ilgio ir panašių veiksnių.

$$F = Pabaiga + T - N, \tag{2}$$

čia:

F – tinkamumo funkcija,

Pabaiga – laimėjimų ir pralaimėjimų funkcijos įtaka,

T – teigiamai įtakojančių bruožų funkcijos vertė,

N – neigimai įtakojančių bruožų funkcijos vertė.

Tinkamumo „*F*“ f-ja yra įtakojama „*Pabaigos*“ f-jos, kurios reikšmė priklauso nuo pergalių ir pralaimėjimų kiekio.

$$Pabaiga = W * 50 + L * (-50), \quad (3)$$

čia:

Pabaiga – laimėjimų ir pralaimėjimų suma,

W – laimėtų žaidimų skaičius,

L – parlaimėtų žaidimų skaičius.

„*Pabaigos*” f-jos laimėtų ir pralaimėtų žaidimo partijų kiekių svoriai yra lygūs ir pakankamai dideli, kad atsvertų „*T*“ ir „*N*“ f-jų reikšmes tuo atveju, kai nei vienas iš agentų nepasiekė pergalės. Nei vienam agentui nepasiekus pergalės mūšio metu, laimintys agentai užsitikrina kur kas didesnę tikimybę, kad bus atrinkti į elitą ir susilauks savo vaikų. Nes laiminčių agentų tinkamumo funkcijos reikšmė yra didesnė. Taip iškeliamas prioritetas ne tik dirbtinio intelekto žaidėjams žaisti gerai, bet ir stengtis dažnai laimėti.

$$T = pagarytaŽala * 10 + nužudymai * 2 + draugai * 3 + (100 / (ėjimai - 2)), \quad (4)$$

Čia:

T – teigiamai įtakojančių bruožų funkcijos vertė,

pagarytaŽala – oponento bazei padarytas žalos skaičius,

nužudymai – nukautų oponento karių skaičius,

draugai – draugiškų kareivių skaičius žaidime, žaidimui pasibaigus,

ėjimai – panaudotų ėjimų skaičius, kol buvo pasiekta pergalė. Šis kaičius sumažinamas dviem, nes pirmi du žaidimo ėjimai atliekami statant pinigų gamyklas.

Teigiamam „*T*“ f-jos dydžiui, didžiausią įtaką turi oponento bazei padaryta žala. Tris kartus mažiau įtakoja – po žaidimo pabaigos likusių draugiškų kareivių kiekis, karo lauke. O mažiausiai „*T“* f-ją įtakoja nužudytų oponento karių skaičius ir sužaisto žaidimo greitis.

$$N = gautaŽala * 10 + praradimai * 3 + priešininkai * 3, \quad (5)$$

čia

N – neigiamai įtakojančių bruožų funkcijos vertė,

gautaŽala – savo bazės sužalojimų skaičius,

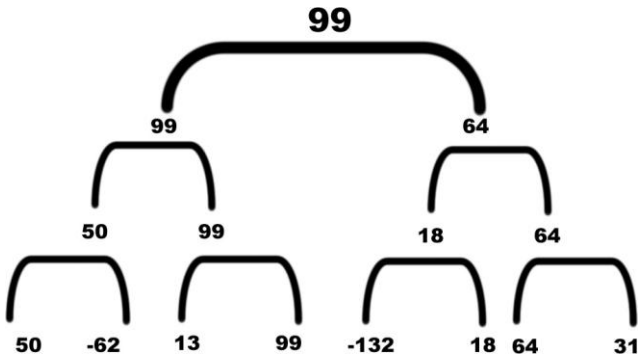
praradimai – prarastų savo karių skaičius,

priešininkai – oponento kareivių skaičius žaidime, žaidimui pasibaigus.

Neigimam „*N“* f-jos dydžiui, didžiausią įtaką turi sužeidimų skaičius savo bazei. Tris kartus mažiau įtakoja šią f-ją - prarastų kareivių ir likusių gyvų oponento kareivių skaičiui. Kaip galima pastebėti ir „*T“* ir „*N“* funkcijų, prarastų kareivių neigiama įtaka yra kur kas didesnė nei nužudytų. Tokiu būdu - skatinama pasirinkti racionalesnius kovos būdus ir nekovoti ten kur tikrai pralaimėsi.

3.7.2 Panaudota atranka

Atrankos kriterijų aprašo tinkamumo funkcija, kuri ir lemia agentų atrankos procesą. Atrankai atlikti naudota turnyrinė atranka, kai visi žaidėjai yra lyginami tarpusavyje ir išrenkami didžiausią tinkamumo f-jos reikšmes turintys agentai.

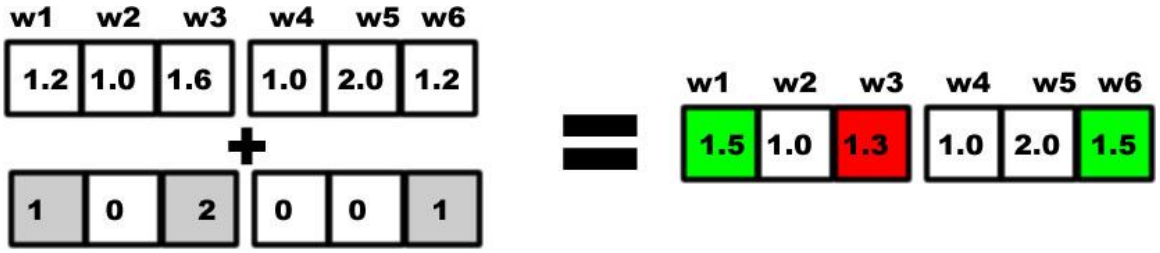


3.5 pav. Turnyrinės atrankos pavyzdys

Paveikslėlyje pavaizduota, kaip taikant turnyrinę atranką randami elitiniai agentai. Atranka vykdoma tol, kol randamas užsibrėžtas elitinių agentų kiekis. Bendras populiacijos tinkamumo vidurkis taip pat pakyla po kiekvienos atrankos pritaikymo, kadangi išliks tik „geriausi“ agentai.

3.7.3 Panaudota mutacija

Agentams, kurie praėjo atranką sėkmingai, pritaikoma mutacija. Darbe realizuotas mutacijos procesas panaudoja burtų keliu sugeneruotą mutacijos kaukę. To pasėkoje yra palaikoma genetinė įvairovė.



3.6 pav. Panaudotos mutacijos pavyzdys

Paveikslėlyje pavaizduotas mutacijos procesas naudojant tris taškus, kur kitimo dydis lygus 0,3. „*w1, w2, w3, w4, w5, w6*“ yra kintamųjų koeficientų vardai, paduoti genetiniam algoritmui. Prie jų pridedama burtų keliu sugeneruota kaukė. Kaukė sugeneruojama naudojant sveikus skaičius nuo 0 iki 2. Ten, kur kaukės reikšmė lygi „1“, koeficiento reikšmė padidinama. Ten, kur kaukės reikšmė lygi „2“, koeficiento reikšmė sumažinama. Koeficiento kitimas vyksta per nustatytą sistemoje kitimo dydį. O ten, kur kaukės reikšmė lygi „0“ – koeficientai nekis.

Realizuoto mutacijos proceso taikymas padeda išvengti žaidėjų chromosomų supanašėjimo. Nes supanašėjimas vestų į evoliucijos sulėtėjimą ar net visišką sustingimą. Kadangi mutacijos mechanizmas buvo kuriamas pagal biologinę mutaciją, visi išsigimimai (reikšmės nukrypusios žemiau nulio) yra automatiškai atmetami, kaip panašiai kaip gamtoje.

3.7.4 Panaudotas kryžminimas

Realizuotas kryžminimo procesas, kurį taiko genetinis algoritmas, padeda išlaikyti ne tik genetinę įvairovę, bet ir jos kokybę. Nes sujungus kelis gerus agentus į vieną, tikimybė gauti trečią gerą agentą yra kur didesnė. Kiekvieno naujo agento sukūrimui, iš populiacijos paimame pora agentų „tėvų“, iš kurių dalis informacijos yra sumaišoma tarpusavyje. Tokiu būdu gaunamas naujas agentas „vaikas“.

3.7 pav. Trijų taškų kryžminimo pavyzdys

Paveikslėlyje pavaizduotas kryžminimo procesas naudojant tris perskyrimo taškus, kur: „w1, w2, w3, w4,w5,w6“ yra kintamųjų vardai, paduoti genetiniam algoritmui. Pagal pavyzdį viršuje matyti, kad atliekant kryžminimą, buvo paimti du „tėvai“ ir jiems sugeneruota kryžminimo kaukė. Kryžminimo kaukės veikimas yra paprastas: „0“ išsaugo pirmojo tėvo duomenis, o kitos reikšmes – antrojo tėvo. Taip gaunamas visiškai naujas agentas - „sukurtas Vaikas“, tikintis, kad jis yra dar geresnis žaidėjas, nei buvo jo tėvai.

Kryžminimo taikymas padeda greitai rasti apytikslį užduoties sprendimą. Galiausiai po šių procesų naująsias kartas sudarys agentai, kurių chromosomos (sandara) yra visiškai skirtinga nei pradinės populiacijos.

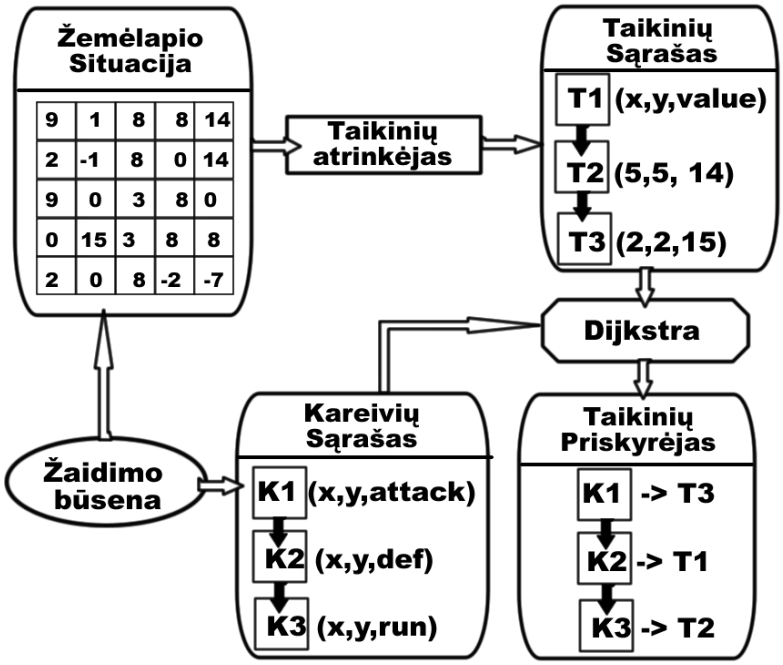
3.8 Ėjimais paremto žaidimo valdymo sistemos projektavimas

Sukurta konkretaus ėjimais paremto žaidimo valdymo sistema yra atsakinga už visus su žaidimu susijusius veiksmus:

- Žemėlapio situacijos įvertimą
- Kareivių tarpusavio santykių įvertinimą
- Optimalių taikinių radimą
- Trumpiausio kelio radimą
- Taikinių paskirstymą kareiviams
- Naujų kareivių gamybą

3.8.1 Pritaikytas bendrasis karių judėjimo procesas

Realizuota žaidimo sistema stengiasi užtikrinti karių judėjimą žemėlapyje. Karių judėjimo procesas prasideda nuo žemėlapio situacijos įvertinimo ir karių motyvavimo judėti tikslo link. Bendras proceso modelis išrenka optimalius taikinius iš žemėlapyje susidariusios situacijos įvertinęs karių tipus. Tada naudojant trumpiausius kelius, kiekvienam kareiviui priskiriamas po individualus taikinys (3.10pav.).



3.8 pav. Bendras karių judėjimo proceso modelis

Veikimo eiga pagal schemą viršuje:

1) Formuojama žemėlapio situacija. Žemėlapio situacija įvertinama naudojant įtakos žemėlapius ir kareivių tarpusavio santykius. Visi įtakos žemėlapiai susumuojami ir gaunama žaidimo situacijos matrica, kuri yra sudaryta iš taikinių duomenų struktūros reikšmių.
2) Taikinių atrinkėjas, iš žemėlapio situacijų matricos atrinka taikinius ir talpina juos į taikinių duomenų sąrašą. Po to, sąrašas rikiuojamas pagal taikinių svarbos reikšmes. Rikiavimas atliekamas tam, kad didžiausios svarbos taikiniai būtų sąrašo pradžioje.
3) Einant per kareivių sąrašą tikrinama, kuriuos taikinius kareivis pasiektų iš taikinių sąrašo. Pasiekiamumas apskaičiuojamas naudojant Dijkstros algoritmą.
4) Taikinių priskyrėjas iš pasiekiamų kareivio taikinių, atrenką geriausiai individualiai jam tinkantį taikinį. Jei šį taikinį jau turi išsirinkęs kitas karys, tokiu atveju jam priskirimas mažiaus tinkamas taikinys.

Šis procesas kartojamas tol, kol visi agento kareiviai gauna po individualų taikinį.

3.8.2 Kareivių tarpusavio santykiai

Kareiviai tarpusavyje skiriasi statistikomis ir specialiaisiais įgūdžiais. Įvertinus kareivių puolimo ir gyvybių santykius matyti, kad skirtingus parametrus turintys kareiviai turėtų būti skirti skirtingoms užduotims vykdyti [žr. 3.3lentelė].

Žaidimo sistemoje realizuotus kareivių tarpusavio santykius atspindi karių santykių lentelė:

3.3 lentelė Kareivių tarpusavio santykių lentelė

Puolimas/Gyvybės, įgūdis	**2/1, FA**	**1/3**	**3/1**	**1/1**	**1/1, DP**
2/1, FA	-4	-6	+12	+12	-4
1/3	-12	-4	-6	+6	+6
3/1	-12	-4	-4	-4	-12
1/1	-12	-10	-4	-4	-12
1/1, DP	-4	-6	+12	+12	-4

Į lentelę žiūrima iš kairės į dešinę. Iš kareivių tarpusavio santykių lentelės matyti, kad vieni kareiviai skatinami kautis su antrais ir vengti trečių. Pavyzdžiui: „1/3“ kareivis skatinamas kautis prieš „2/1, FA“, bet jam patartina vengti „3/1“. Atidžiau pasižiūrėjus į kareivių puolimo ir gyvybių santykius matyti, kad jie yra skirti skirtingoms strategijoms vykdyti.

3.8.3 Panaudoti įtakos žemėlapiai

Realizuota ėjimais paremto žaidimo sistema naudoja visą grupę įtakos žemėlapių, kurie atsakingi už skirtingus įverčius. Pradedant nuo vieno svarbiausių „žaidimo tikslo motyvavimo“ ĮŽ, kuris skatina kirsti karo lauko matricą į kitą pusę ir nueiti iki oponento bazės.

1	1	1	1	1
2	2	2	2	2
4	4	4	4	4
8	8	8	8	8
12	12	12	12	12

3.9 pav. Žaidimo tikslo motyvavimo įtakos žemėlapis

Paveikslėlis atspindi karių motyvaciją eiti į apačią, nes langelių esančių apačioje kur kas didesnės reikšmės.

Karo laukas yra įtakojamas visų oponento kareivių. Aplink kiekvieno kareivį daroma įtaka, priklauso nuo aplink esančių oponento kareivių, tarpusavio santykių lentelės ir kareivių judėjimo greičio.

0	0	-6	0	0
0	-6	-6	-6	0
-6	-6	-12	-6	-6
0	-6	-6	-6	0
0	0	-6	0	0

3.10 pav. Oponento ,,2/1, FA“ kareivio įtaka karo laukui atsidūrus prieš ,,3/1“ kareivį

Paveikslėlyje matyti, kaip įtakojamas karo laukas, kai ,,3/1“ kareivis susitinka oponento ,,2/1, FA“ kareivį. Aplink ,,2/1“, kurio judėjimo atstumas lygus dviem, esančios reikšmės yra mažinamos per pusę ,,-6“, nei nurodomos kareivių tarpusavio santykių lentelėje. O ,,2/1“ užimama pozicija yra lygi šios lentelės pilnam reikšmės dydžiui ,,-12“.

Sekantis panaudotas ,,gynybos motyvavimo“ įtakos žemėlapis. Gynybos motyvatorius suveikia kiekvieną kartą, kai pasiekiamu atstumu prie bazės priartėja oponento kareivis. Tam atsitikus, viso kelio nuo oponento kareivio iki bazės, žemėlapio įtakos reikšmės padidėja. Reikšmės didinamos priklausomai nuo kareivio puolimo jėgos, kur jėga padauginama iš 10.

0	0	0	0	0
0	0	0	0	0
0	0	„1/1“	0	0
0	0	10	0	0
0	0	10	0	0

3.11 pav. Žaidimo gynybos motyvavimo įtakos žemėlapis

Paveikslėlyje matyti, kaip įtakojamas karo laukas, kai oponento ,,1/1“ kareivis priartėja prie bazės. Gynybos motyvavimo įtakos žemėlapis, skatina kareivius apsispręsti ar ginti savo bazę, nebent puolimas yra labai silpnas.

,,Apgulties išlaikymo“ įtakos žemėlapis, skatina kareivius sėkmingai pribėgus prie oponento bazės ten ir likti, atliekant manevravimus į kairę ir dešinę, kurie padeda išvengti oponento kareivių.

30	„3/1“	30	30	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

3.12 pav. Žaidimo apgulties išlaikymo įtakos žemėlapis

Paveikslėlyje matyti, kaip įtakojamas karo laukas, kai savas kareivis sėkmingai pasiekia oponento bazę. Apgulties išlaikymo įtakos žemėlapis, skatina kareivį manevruoti į šonus, nes tai padeda išvengti oponento kareivių persekiojimo. Manevravimo motyvacija priklauso nuo kareivio puolimo jėgos padaugintos iš 10. Pavyzdžiui, kaip matyti viršuje esančiam paveikslėlyje – kareiviui ,,3/1“ pribėgu prieš oponento bazės, jis skatinamas judėti į šonus ,,30“ verte (3puolimo galia * 10 = 30).

3.8.4 Panaudotas antikūnių gamybos planuotojas

Realizuotas antikūnių gamybos planuotojo atlieka žemėlapyje esančių kareivių analizę. Remdamasis analizės duomenimis, padaromos išvados: kiek ir kokių kareivių yra ant žaidimo lentos; kokius kareivius yra patartina gaminti.

Gamybos planuotojas stengiasi palaikyti harmoniją, išlaikant kuo panašesnį visų rūšių kareivių skaičių, bet atsižvelgiant ir į oponento kareivių pasiskirstymą. Pavyzdžiui, jei oponentas gamins vien greitus kareivius, realizuotas gamybos planuotojas pradės gaminti daugiau ,,anti-greitų“ kareivių ir atvirkščiai.

3.9 Išvados

Projektavimo ir realizacijos metu buvo atlikta:

1 realizuoti šie metodai:
   - genetinis algoritmas
   - neuroninis tinklas
   - įtakos žemėlapiai
   - antikūnių gamybos planuotojas
2 suprojektuota ir realizuota dirbtino intelekto panaudojimo aplinka – dalis ėjimais paremto žaidimo sistemos dirbtinio intelekto metodų tyrimui atlikti
3 dirbtinis intelektas pritaikytas konkretaus žaidimo taisyklėms laikytis

Realizuota ėjimais paremto žaidimo sistema yra tinkama aplinka, dirbtinio intelekto taikymui ir tyrimui atlikti. Bendrajam projekte realizuoti keli pagrindinai dirbtinio intelekto metodai.

4 EKSPERIMENTINĖ DALIS

Sukurta žaidimo sistema yra skirta dirbtinio intelekto metodams testuoti. Norint patikrinti sukurtos sistemos ir metodų veikimo korektiškumą, atliekamas eksperimentinis tyrimas. Tyrimo metu tarpusavyje lyginami sukurti dirbtinio intelekto metodai, bei nustatytas jų tinkamumas tolimesniems pritaikymams.

Eksperimentui vykdyti buvo parinkta suprojektuota ėjimais paremto žaidimo sistemos dalis, kurios turėjo užtekti dirbtinio intelekto naudojamų metodų palyginimui. Patikrintas neuroninio tinklo, įtakos žemėlapių ir gamybos planuotojo dirbtinio intelekto metodų efektyvumas. Patikrinta neuroninio tinklo apmokymo priklausomybė nuo apmokymo laiko ir kokybės. Tarpusavyje palyginti dirbtinio intelekto metodai panaudojant juos konkrečiai vieną prieš kitą.

4.1 Eksperimentų planavimas

Šioje dalyje yra aprašyti suplanuoti eksperimentai. Eksperimentuose tiriami realizuoti dirbtinio intelekto metodai, jų veikimo efektyvumas ir tinkamumas konkrečiai ėjimais paremto žaidimo sistemai. Taip pat bus pateikiamas jų vykdymas ir gauti rezultatai.

Tiriant aplinkos dirbtinio intelekto metodus, atkreipiamas dėmesys į metodo:

- rezultatų tinkamumo koeficientą problemai spręsti
- laimėtų ir pralaimėtų agento žaidimų skaičių
- žaidimo trukmę

4.2 Metodų palyginimas

Realizuoti dirbtinio intelekto naudojami metodai remiasi skirtingomis žaidimo strategijomis. Visi trys metodai skiriasi problemos sprendimo principu. Todėl lyginsime ne tik kiekvieno metodo gerai išspręstų problemų skaičių, bet ir patikrinsime, kaip jie rungiasi vienas prieš kitą.

Iš Gamybos planuotojo tikimasi, kad jis užtikrins gerą gynybą, nes jo veikimo principas paremtas antikūnių ir antigenų įvertinimu žemėlapyje susidariusioms situacijoms.

Genetiniu algoritmu apmokytas neuronų tikslas arba duos labai gerus, arba labai prastus galutinius rezultatus, nes jį labai sunku apmokyti.

Įtakos žemėlapių struktūros panaudojimas, turėtų duoti gana aukštus rezultatus, nes jis „pasveria“ didžiąją dalį galimų sprendimo variantų, iš jų išsirinkdamas pačius optimaliausius.

4.3 Apmokymas ir parametrų parinkimas

Prieš nagrinėjant dirbtinio intelekto taikomus metodus, patartina naudoti tas jų parametrų reikšmes, prie kurių jie duoda geriausius problemų sprendimo rezultatus. Tam mes panaudosime genetinį algoritmą arba bandymų keliu ieškomą geresnių parametrų koeficientų reikšmę problemoms spręsti.

Pradžioje parenkamos pradinės parametrų svorių reikšmės, kurios pritaikius genetinį algoritmą arba bandymų keliu ieškomą parametrų paiešką, pasikeis atitinkamoms situacijoms. Parenkami pradiniai parametrai matomi (4.1 lentelė).

4.1 lentelė Skaičiavimo koeficientai

Koeficientai	W0	W1	W2	W3	W4	W5	W6	W7	W8
Įtakos žemėlapio Kovos	0,5	1	1	1	1	1	1	1	1
NT Judėjimo	1	1	1	1	1	1			

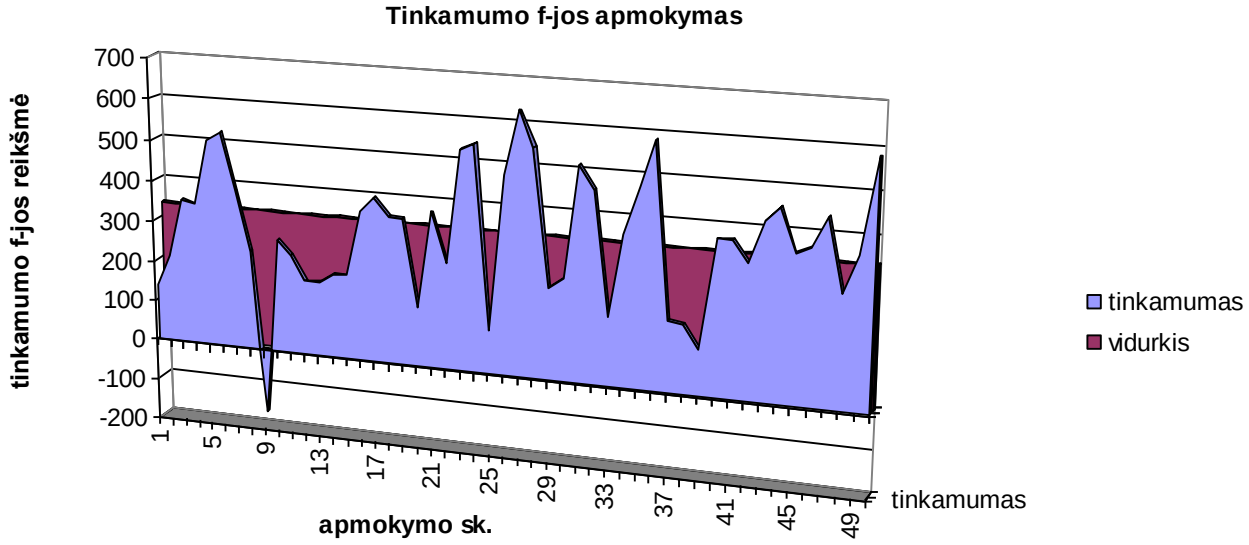
Tokios pradinių koeficientų reikšmės parinktos neatsitiktinai. Kadangi koeficientai naudojami funkcijose, kurios naudoja daugybą skaičiavimų rezultatams gauti, dauginimas iš vieneto – neturi jokios pradinės įtakos.

4.3.1 Įtakos žemėlapių

Įtakos žemėlapius paremtas metodas, naudoja įtakos žemėlapių struktūrą, kurią pritaiko ir trečiasis tiriamas metodas – gamybos planuotojas.

Parinkti kovos svorių koeficientai, kurie gali koreguoti įtaką tam tikrose žemėlapio zonose, buvo nustatyti lygūs vienetui (4.3 lentelė) ir kiekvienam koeficientui nustatytas kitimo žingsnis lygus 0,3.

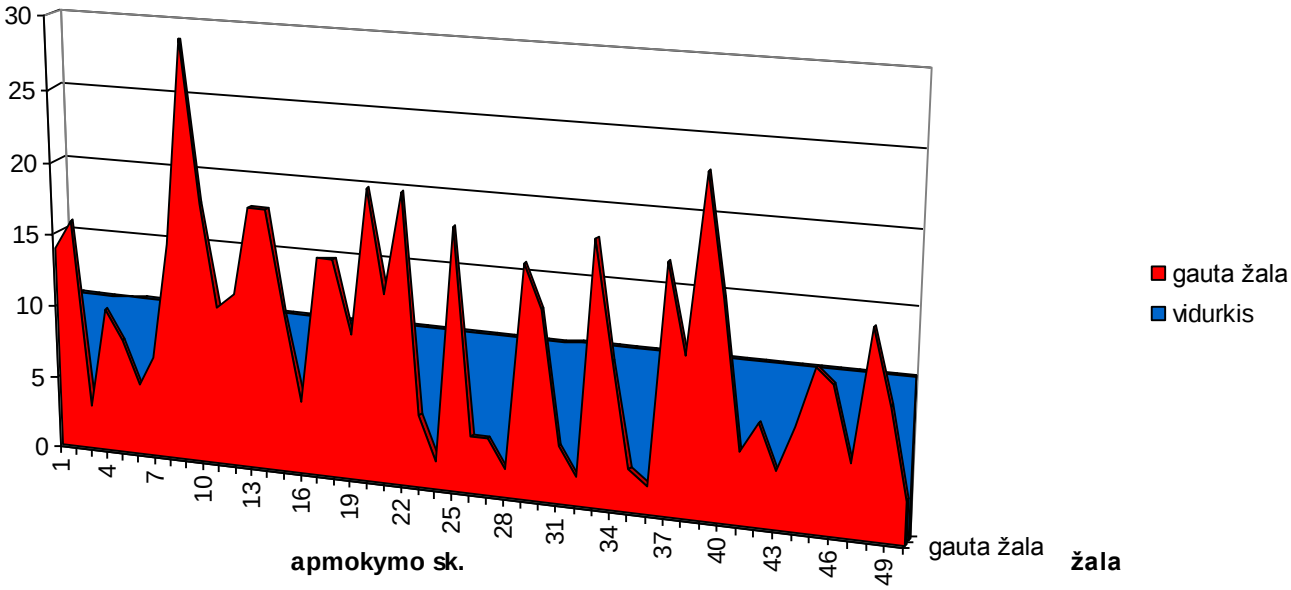
Svorių radimas bandymų keliu tyrė 500 atskirai sužaistų žaidimo partijų. Kiekvieną kartą, sužaidus 10 partijų, žaidimo agento panaudoti sprendimai būdavo įvertinami tinkamumo funkcija (žr. 4.1 pav.).



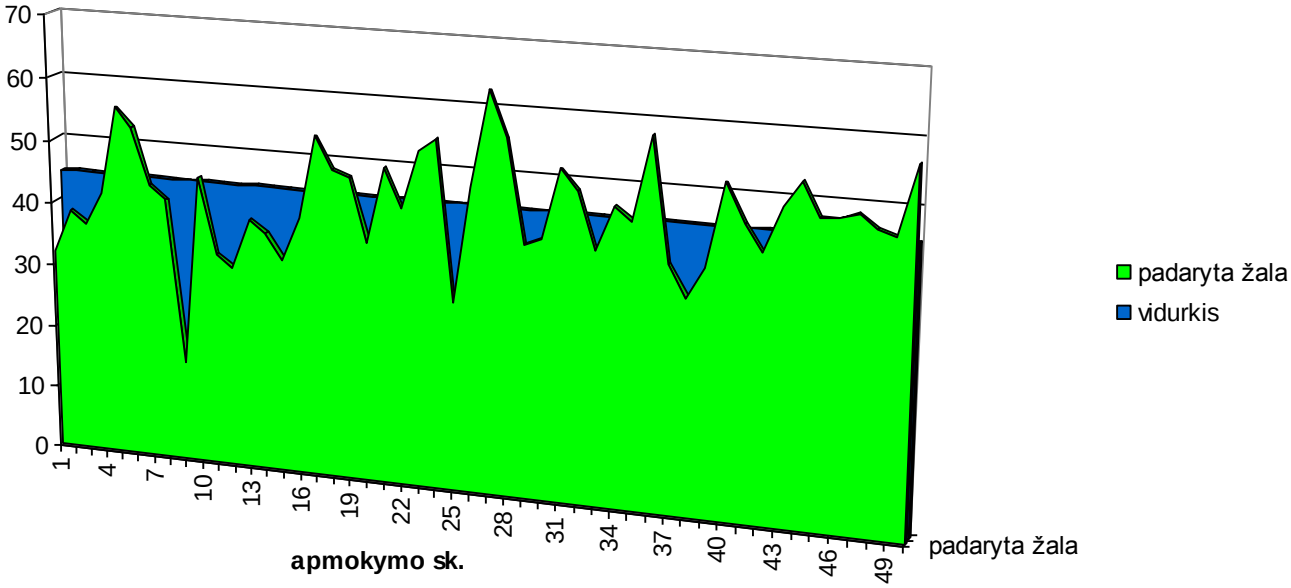
4.1 pav. Tinkamumo f-jos naudojimas įtakos žemėlapiuose

Iš paveikslėlio viršuje matyti, kad jau pradžioje turėti koeficientai davė gerą tinkamumo f-jos rezultatą lygų maždaug 110. Panaudojant bandymų paieškos keliu keičiamus kovos koeficientus – rezultatai staigiai didėjo, kol pasiekė 65 sužaistas žaidimo partijas. Nuo to laiko svorių paieška ilgą laiką nedavė gerų rezultatų. Taip dažnai nutinka, kai sprendimai gaunami naudojant bandymų paieškos būdą keičiamus koeficientus ir parametrus.

Iš tinkamumo f-jos gauto vidurkio matyti, kad bandymų keliu rasti koeficientai (w1-w6) duoda beveik triskart geresnius sprendimo įvertinimo reikšmę, lygią 340. Tinkamumo funkcijos gauti rezultatai, nebūtinai atspinti metodo gerumą, bet tai puikus būdas įvertinti problemos sprendime pasiektus rezultatus, kurie gerai atsispindi sekančiose diagramose (4.2 pav. ir 4.3 pav.).

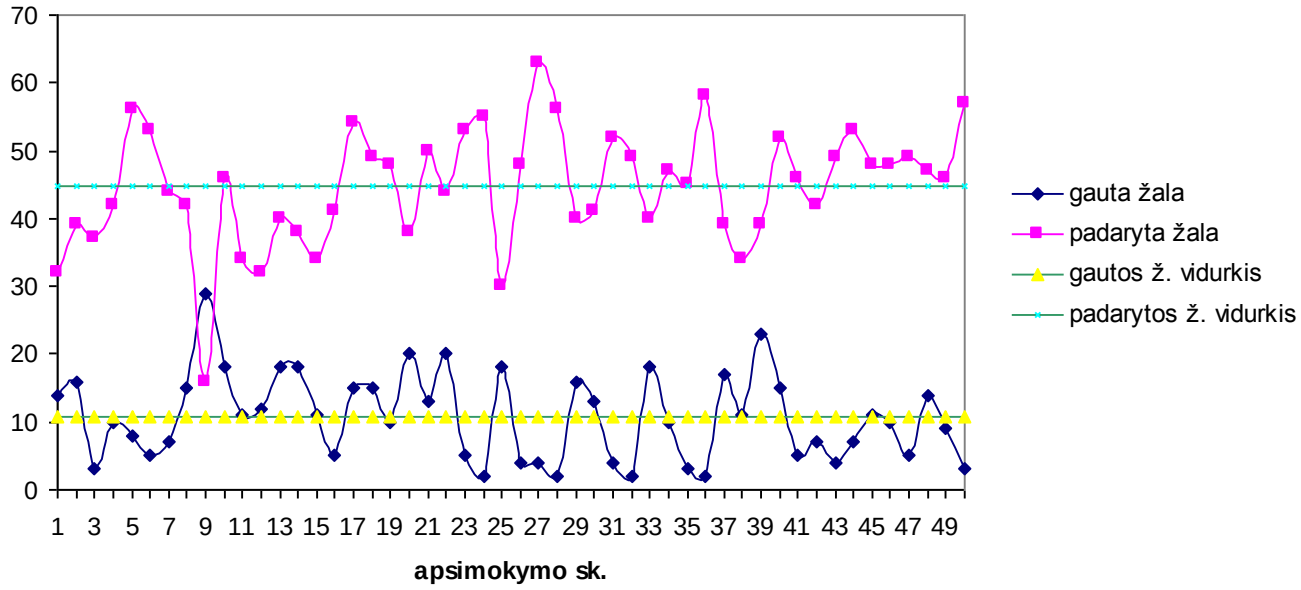


4.2 pav. Agento bazei padarytas žalos kiekis



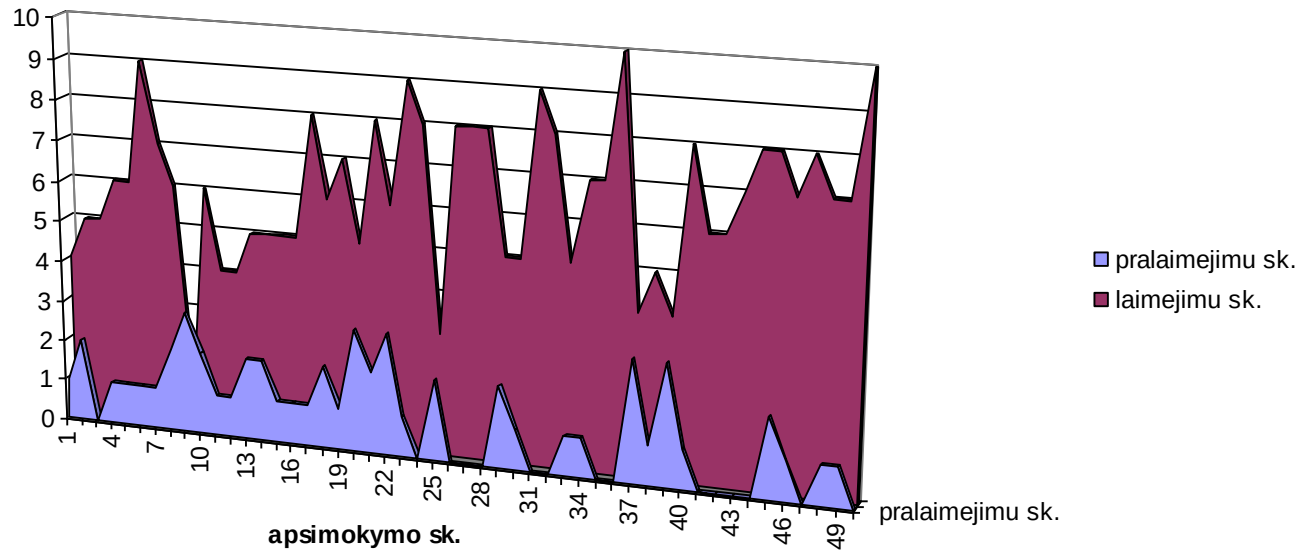
4.3 pav. Agento bazei padarytas žalos kiekis

Iš viršuje esančių paveikslėlių aiškiai matoma, kad padarytos priešininkams ir gautos žalos skaičius skiriasi beveik keturis kartus (4.4 pav.).



4.4 pav. Draugiškos ir oponento bazės sužalojimai

Mažas gautos žalos kiekis, leidžia daryti prielaidą, kad agentas gerai ginasi, o iš aukštų padarytos žalos rezultatų – kad agentas dažnai pasiekia žaidimo tikslą arba būna labai netoli jo, kas puikiai matoma metodo laimėjimų kiekyje (4.5 pav.).



4.5 pav. IŽ metodo pralaimėjimų ir laimėjimų kiekis

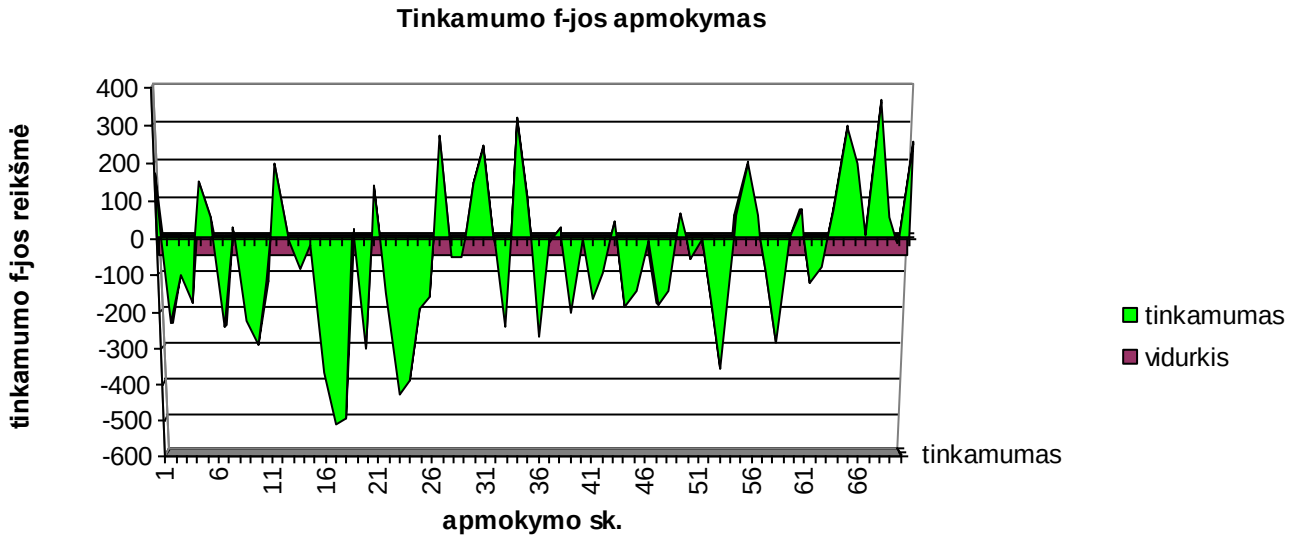
Pagal matomą žaidimo laimėjimų skaičiaus augimą ir išsilaikymą, galima spręsti, kad praktiniu keliu rasti sprendiniai ir jų koeficientai yra paruošti lyginimui su kitais metodais.

4.3.2 Neuroninio Tinklo apmokymas genetiniu algoritmu

Bendroje sistemoje pritaikyto neuroninio tinklo apmokymui panaudotas genetinis algoritmas.

GA realizuotas taip, kad panaudotų daugumą į evoliucijos procesą įeinančių veiksmų. NT apmokymo ciklas buvo vykdomas 700 kartų. Pradiniai svorių koeficientai, kurie yra taikomi NT, parinkti lygūs vienetui (4.2 lentelė) ir kiekvienam koeficientui nustatytas genetinio algoritmo kitimo žingsnis lygus 2,3.

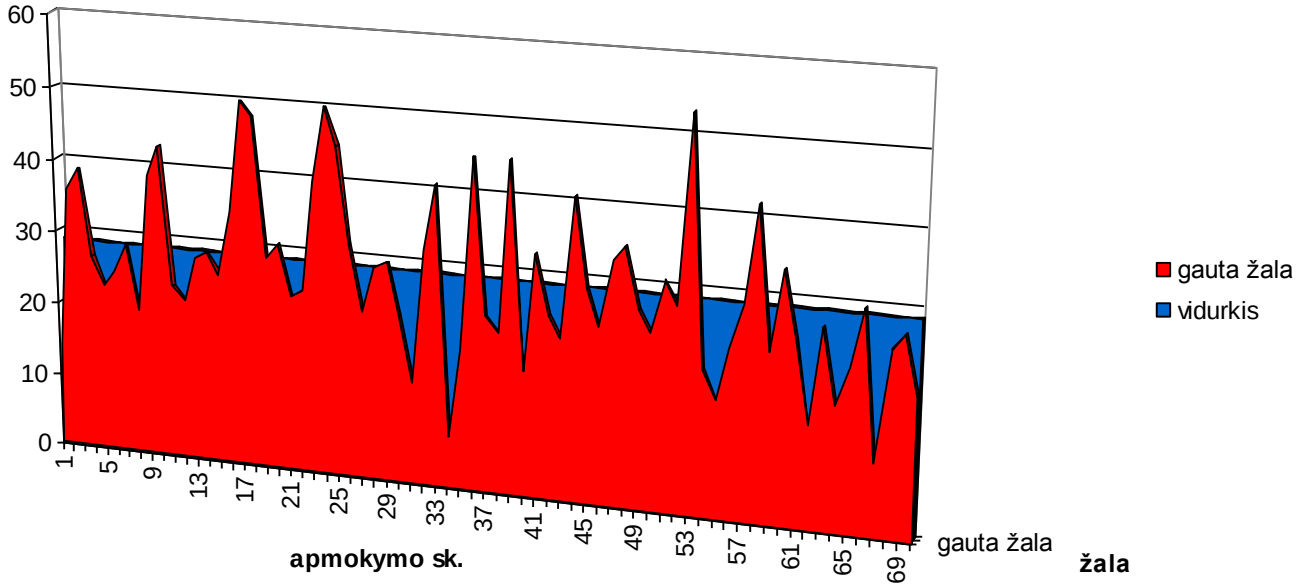
GA sužaidė ir atliko 700 žaidimo generacijų. Kiekvieną kartą sužaidus 10 partijų, žaidimo agento panaudoti sprendimai būdavo įvertinami tinkamumo funkcija (4.6 pav.).



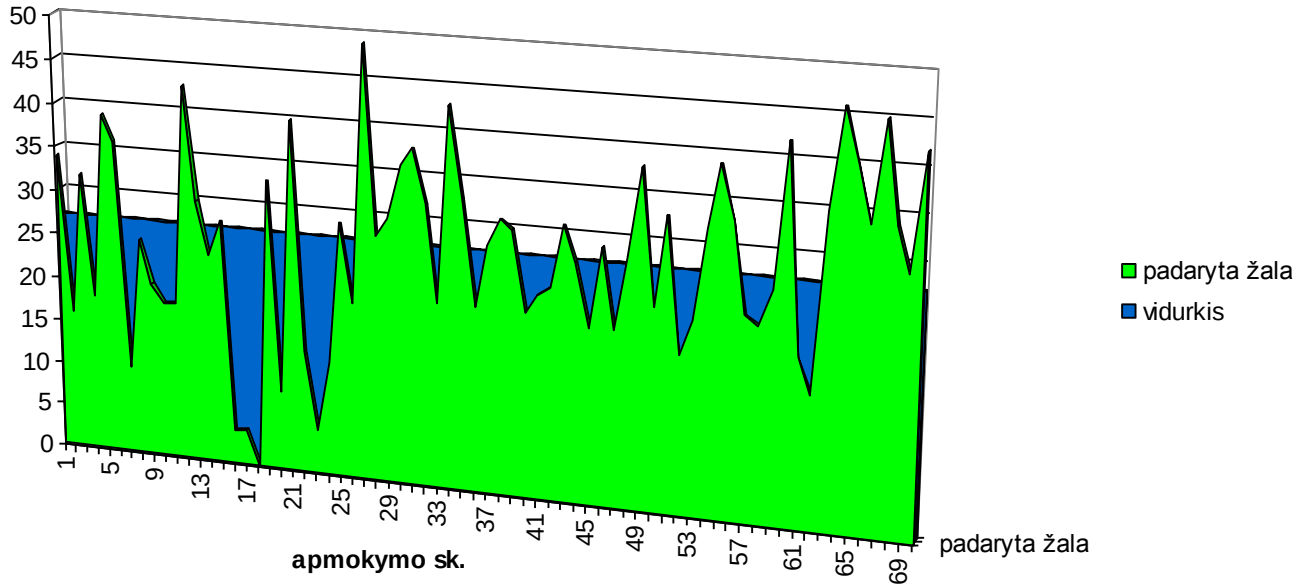
4.6 pav. Tinkamumo f-jos naudojimas NT apmokyme

Iš paveikslėlio viršuje matyti, kad apsimokymo procesui sekėsi sunkiai, nes tik po 280 sužaistų žaidimo partijų, tinkamumo funkcijos gaunami rezultatai nusistovėjo ties vidurkiu. Perkopus 600 sužaistų žaidimo partijų kiekį tinkamumo funkcija dažniau pradėjo duoti teigiamus rezultatus.

Iš tinkamumo f-jos gauto vidurkio matyti, kad bendras vidurkis nusistovi žemiau nulio. Tinkamumo funkcijos gauti rezultatai, nebūtinai atspinti tikrąjį metodo gerumą, bet tai puikus būdas įvertinti problemos sprendime pasiektus rezultatus, kurie gerai atsispindi sekančiose diagramose (4.6 pav. ir 4.7 pav.).

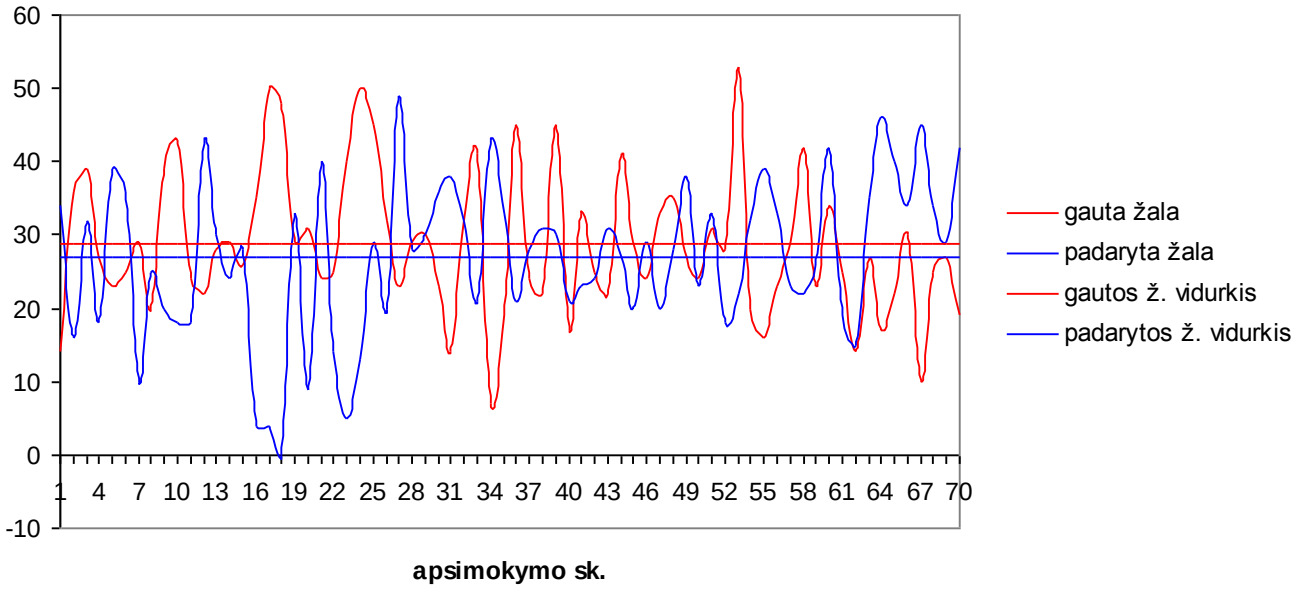


4.6 pav. Agento bazei padarytas žalos kiekis



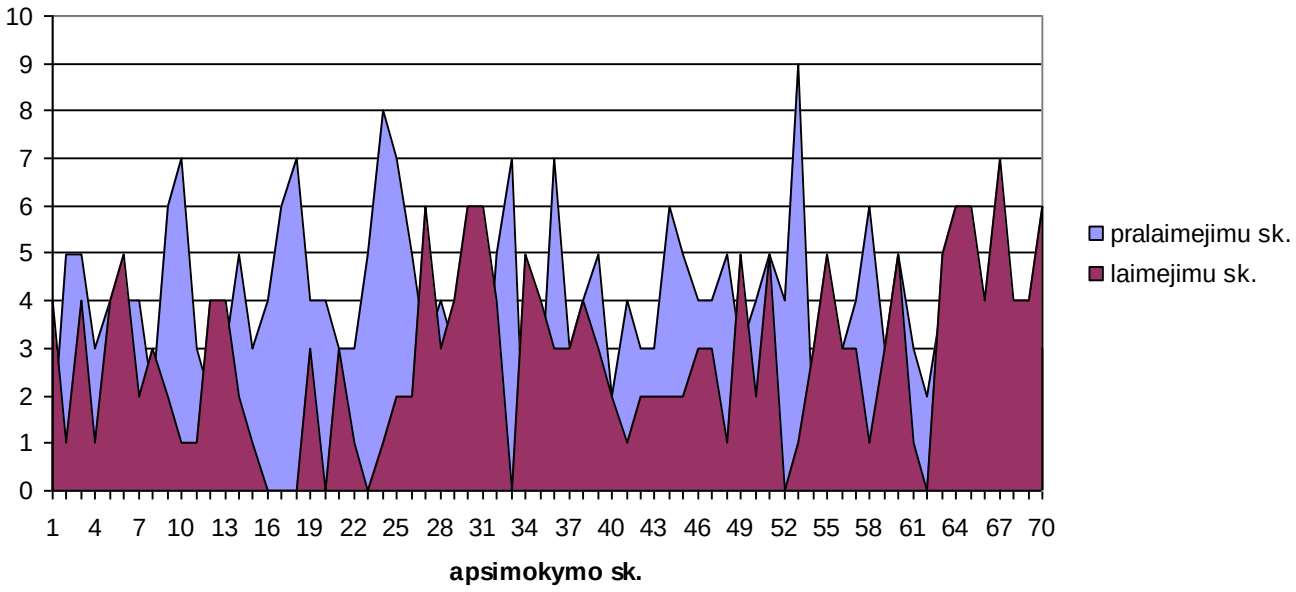
4.7 pav. Agento bazei padarytas žalos kiekis

Iš viršuje esančių paveikslėlių aiškiai matoma, kad padarytos priešininkams ir gautos žalos vidurkiai skiriasi labai nežymiai. Bet atkreipus dėmesį į pateiktus duomenis nuo 620 apsimokymo, matosi kad padaromos žalos kiekis ženkliai išaugo, o gaunamos sumažėjo (4.8 pav.).



4.8 pav. Žaidimo pralaimėjimų ir laimėjimų kiekis

Matoma į galą nusistovėjusi didesnė daroma žala, įtakojo išaugusį žaidimo pergalių skaičių (4.9 pav.). Tam prireikė virš 620 genetinio algoritmo iteracijų.



4.9 pav. Žaidimo pralaimėjimų ir laimėjimų kiekis

Pagal aukščiau matomą žaidimo laimėjimų skaičiaus nepastovumą galima spręsti, kad genetinis neuroninio tinklo apmokymas reikalauja didelio iteracijų skaičiaus.

4.4 Metodų palyginimas

Lyginant dirbtiniam intelektui taikomus metodus vienas prieš kitą (tiesioginėje kovoje), naudojamos, tos jų parametrų reikšmės, prie kurių jie duoda geriausius problemų sprendimo rezultatus.

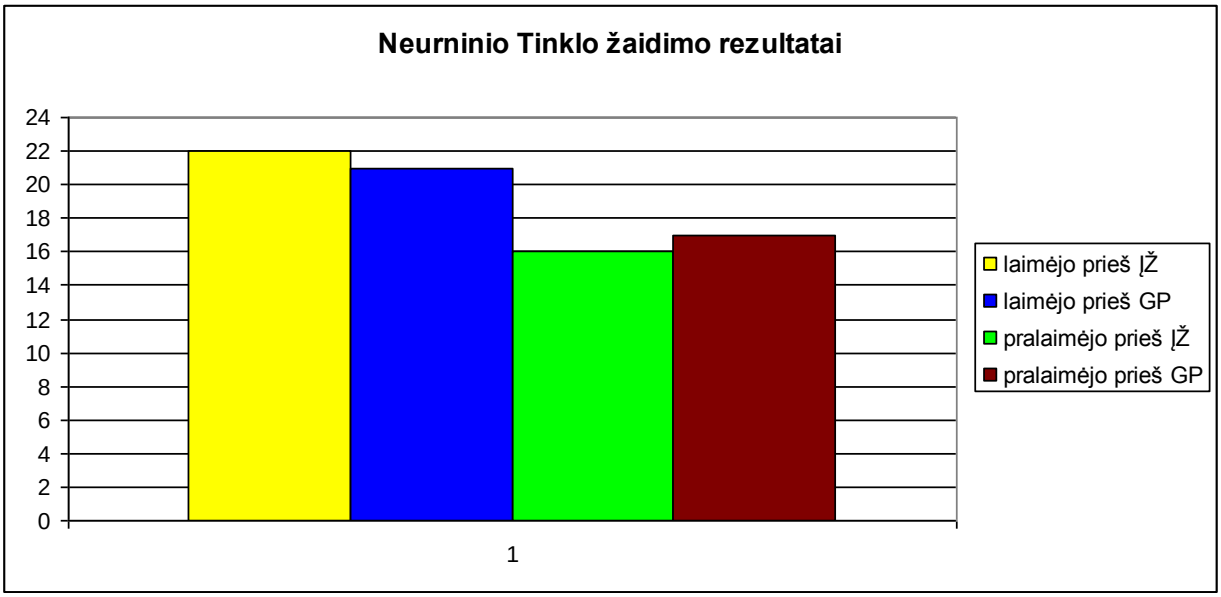
Metodų rezultatai yra lyginami realizuotoje ėjimais paremtoje žaidimo sistemoje.

Pradžioje parenkamos parametrų koeficientų reikšmės, kurios nebekis keičiantis varžyboms. Palyginimo duomenų tikslumui padidinti, tarpusavyje yra sužaidžiamos 50 žaidimo partijų. Kiekvienas metodas, paeiliui paleidžiamas žaisti prieš visus kitus metodus.

Pagrindiniai palyginimo kriterijai: laimėtų partijų ir panaudotas žaidimo ėjimų skaičius.

4.4.1 Laimėjimų skaičius

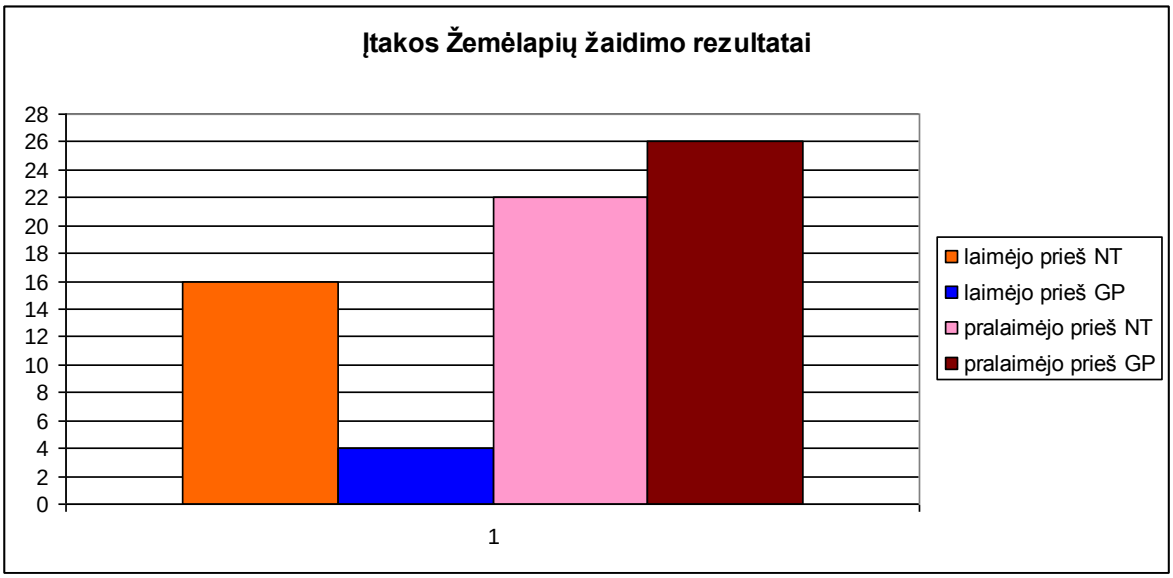
Apmokytas NT sužaidė 50 partijų prieš agentą naudojantį gamybos planuotojo metodą, ir kitą agentą naudojantį įtakos žemėlapiais paremtą metodą. Geriausiai metodo efektyvumą padedantis įvertinti kriterijus – patirtų žaidimo pergalių skaičius (4.10 pav.).



4.10 pav. Žaidimo pralaimėjimų ir laimėjimų kiekis

Iš viršuje esančio paveikslėlio matyti agento naudojančio NT pergalių skaičius prieš kitus agentus. Matomi agento naudojusio NT 20,2% didesnis laimėjimų skaičius, nei patirtų pralaimėjimų.

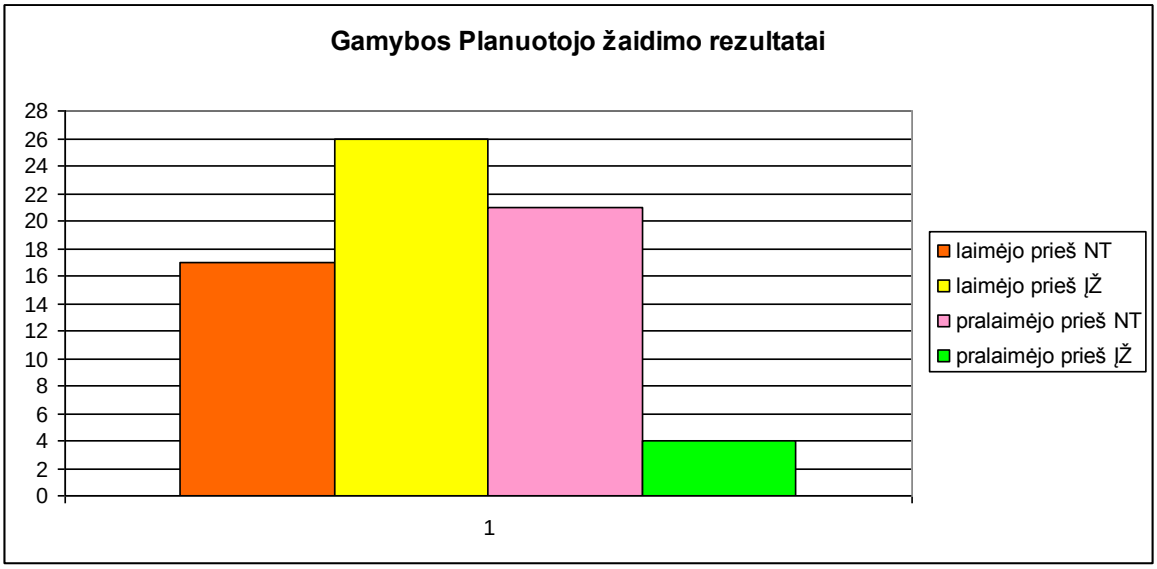
Agento naudojančio ĮŽ metodą sprendimams priimti, gauti rezultatai atrodo prasčiau (4.11 pav.).



4.11 pav. Žaidimo pralaimėjimų ir laimėjimų kiekis

Matomas nedaug atsiliekantis nuo NT naudojusio metodo pasiektų rezultatų. Gautas labai mažas laimėtų pergalių ir didelis pralaimėjimų skaičius prieš gamybos planuotojo metodą naudojusį agentą. Taip nutiko todėl, kad kartais žaidimo metu, susiklostydavo palankios situacijos gamybos planuotojui išnaudoti žaidimo kareivių balansavimo spragas.

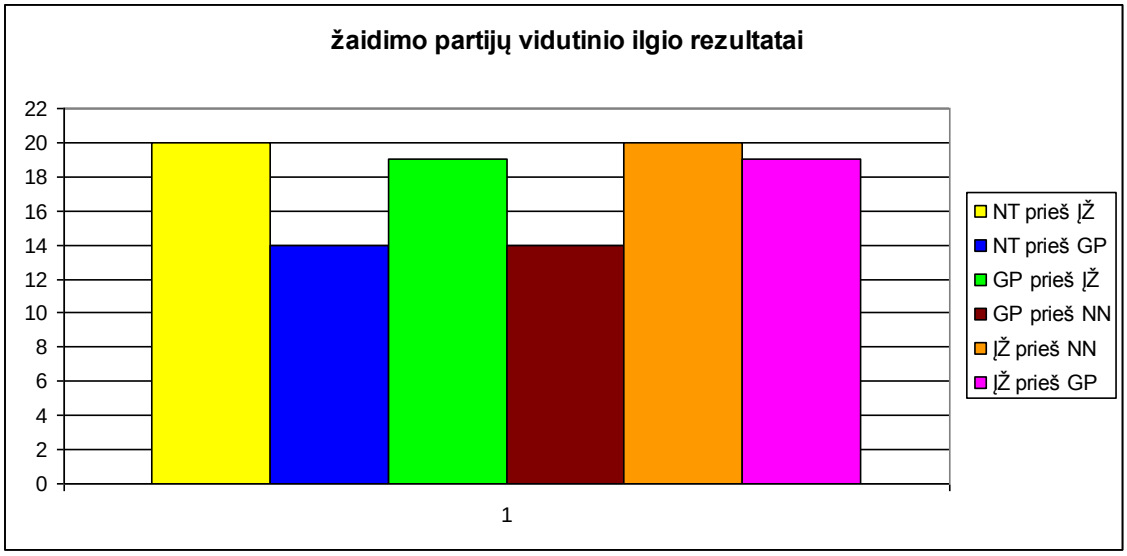
Atsitiktinai aptikta žaidimo balanso spraga, kai ĮŽ agentas nusprendžia arba neturi kito pasirinkimo, gaminti daugiau nei 2 karius, kurie yra labai silpni mūšiuose. Šių karių vienintelis tikslas - prasibrauti iki oponento bazės. Kaip matyti iš gautų rezultatų, gamybos planuotojas efektingai įvertina ir pasinaudoja tokia susidariusia situacija.



4.12 pav. Žaidimo pralaimėjimų ir laimėjimų kiekis

Nors gamybos planuotojo bendras pralaimėjimų vidurkis gana mažas, pastebėtas labai aukštas laimėjimų rodiklis, kurio bendra vidurkį padidino prieš ĮŽ metodą gautas aukštas laimėjimų skaičius.

Iš sekančios diagramos matyti, kad ĮŽ nėra lengvai įveikiamas (4.13 pav.).



4.13 pav. Žaidimo partijos trukmių ilgiai taikant NT, ĮŽ, GP

Iš grafiko viršuje matyti, kad žaidimo partijos vykstančios prieš ĮŽ metodą naudojantį agentą, yra 25.4% ilgesnės. Iš to galima daryti prielaidą, kad ĮŽ metodą naudojantis agentas, labai gerai ginasi ir prailgina žaidimo laiką.

4.5 Eksperimentų rezultatų analizė ir išvados

Geresnių parametrų ir svorių paieška

Iš ĮŽ koeficientų ieškojimuose panaudotos tinkamumo f-jos vidurkio matyti, kad tinkamumo funkcijos reikšmės vidurkis po paieškos tris su puse karto didesnis jei tinkamumo f-jos reikšmė su kuria pradėjome paiešką. Pagal matomą žaidimo laimėjimų skaičiaus augimą ir išsilaikymą, galima spręsti, kad praktiniu keliu rasti sprendiniai ir jų kovos parametrai gali pagerinti metodo duodamus problemos sprendimo rezultatus (tai yra laimėjimų skaičių (4.5 pav.)).

Genetiniu algoritmu apmokomo NT apsimokymo procesui sekėsi labai sunkiai, nes tik po 280 sužaistų žaidimo partijų, tinkamumo funkcijos gaunami rezultatai nusistovėjo ties vidurkiu, kuris davė vidutinius rezultatus. Tik perkopus 600 sužaistų žaidimo partijų kiekį, tinkamumo funkcija dažniau pradėjo duoti teigiamus rezultatus, nei neigiamus. Pagal matomą žaidimo laimėjimų skaičiaus nepastovumą galima spręsti, kad genetinis neuroninio tinklo apmokymas reikalauja didelio iteracijų skaičiaus, kad pasiektų gerus rezultatus (4.9 pav.).

Po optimalių ĮŽ kovos parametrų ir NT judėjimo koeficientų paieškos, įvykus 500 žaidimo iteracijų nusistovėjo šie parametrų ir svorių vidurkiai:

4.2 lentelė Nusistovėję parametrų ir koeficientų vidurkiai

Svoriai	W0	W1	W2	W3	W4	W5	W6	W7	W8
ĮŽ Kovos	0,5	1,2	1,3	0,7	0,6	1	0,7	1,6	0,7
NT Judėjimo	0,4	0,9	0,8	0,5	5,2	1,1			

Metodų lyginimas tarpusavyje

Geriausius rezultatus žaidime pasiekė agentas naudojantis GA apmokytą NT metodą. Jo rezultatai pateikia 20,2% didesnį laimėjimų skaičių, nei patiriamų pralaimėjimų.

Antras pagal laimėjimų skaičių yra agentas naudojantis gamybos planuotoją. Nors gamybos planuotojo bendras pralaimėjimų vidurkis gana mažas, jis vis tiek atsiliko nuo agento naudojusio NT metodą. Taip pat, gamybos planuotoją naudojantis agentas, atsitiktinai aptiko žaidimo kareivių balanso spragą. Iš to padarome išvadą, kad kai kurioms žaidimų balansavimo klaidoms rasti, galima pasitelkti agentą naudojantį gamybos planuotoją.

Prieš gamybos planuotoją labai atsiliko agentas naudojantis ĮŽ metodą. Šio agento gautas labai mažas laimėtų pergalių ir didelis pralaimėjimų skaičius prieš gamybos planuotojo metodą naudojusį agentą. Taip nutiko todėl, kad kartais žaidimo metu susiklostydavo palankios situacijos, kurias įvertinęs gamybos planuotojas galėdavo išnaudoti žaidimo balanso atrastas spragas. Pastebėta, kad žaidimo partijos vykstančios prieš ĮŽ metodą naudojantį agentą, yra 25.4% ilgesnės. Iš to galima daryti prielaidą, kad šis metodas yra tinkamas gynybiniam žaidimui stiliui.

1. Kaip ir tikėtasi palyginus gamybos planuotoją prieš kitus metodus pastebėta, kad jis turi didelį žaidimo laimėjimo skaičių ir labai mažą pralaimėjimų vidurkį.
2. Priešingai nei tikėtasi agentas naudojantis ĮŽ metodą gavo pačius prasčiausius rezultatus, bet pasižymėjo gaudamas geriausius žaidimo trukmės rezultatus.
3. Agentas naudojantis NT metodą, gavo pačius geriausius rezultatus, bet tam prireikė ilgo apmokymo proceso (620 žaidimų iteracijų).

5 IŠVADOS

1. Atlikta panašių realizuotų sprendimų analizė ir nustatyta, kad:
    - įtakos žemėlapiai padeda įvertinti žemėlapio situacijas
    - makro žaidimo strategijos taikyti reikalingas gamybos planuotojas
2. Atlikta žaidimuose taikomų dirbtinio intelekto metodų analizė ir nuspręsta, kad geriausiai taikymui tinka naudoti: įtakos žemėlapius, neuroninį tinklą apmoktą GA ir gamybos planuotoją.
3. Sukurta ėjimais paremto žaidimo sistemos dalis, kurioje realizuoti dirbtinio intelekto taikomi metodai. To užtenka tiriamajam darbui atlikti, tačiau žaidimas bus vystomas ir toliau.
4. Realizuoti šie dirbtinio intelekto naudojami metodai:
    - genetinis algoritmas
    - įtakos žemėlapiai
    - gamybos planuotojas
5. Palyginus ištirtų metodų gautus rezultatus paaiškėjo, kad:
    - geriausius laimėjimų rezultatus duoda agentas, naudojantis neuroninį tinklą. Gaunamas rezultatas yra 20,2% didesnis už vidutinį laimėjimų skaičių palyginus su kitais metodais.
    - gaunamais rezultatais atsilieka agentas, naudojantis gamybos planuotoją.
    - prasčiausius rezultatus duoda agentas, naudojantis įtakos žemėlapius, bet šio agento gauti rezultatai labiausiai prailgina žaidimo laiką, kuris gaunamas 25,4% ilgesnis.
6. Pagal gautus metodų palyginimo rezultatus nustatyta, kad duodantis geriausiu rezultatus „Warlike“ žaidimo dirbtinio intelekto realizacijoje yra NT metodas.

6 NAUDOTA LITERATŪRA

1. T. P. Runarsson and S. M. Lucas, (2005). ,,Co-evolution versus self-play temporal difference learning for acquiring position evaluation in small-board go,"IEEE Transactions on Evolutionary Computation, p. 628–640.
2. N. Richards, D. E. Moriarty, and R. Miikkulainen, (1997) ,,Evolving neural networks to play go," Applied Intelligence, vol. 8, p. 85–96. ISSN: 0924-669X
3. E. C. D. van der Werf, H. J. V. D. Herik, and J. W. H. M. Uiterwijk, (2003). ,,Solving go on small boards," International Computer Games Association Journal, vol. 26, p. 7-10.
4. Hirotaka Itoh, Naoki Ikeda, Kenji Funahashi, (2). ,,Heterogeneous Multi Agents Learning using Improved Genetic Network Programming", Sixth International Conference on Bio-Inspired Computing: Theories and Applications, p. 1-2.
5. Chang Kee Tong, Chin Kim On, Jason Teo, and Aroland MConie Jilui Kiring, (2011) ,,Evolving Neural Controllers using GA for Warcraft 3-Real Time Strategy Game", Sixth International Conference on Bio-Inspired Computing: Theories and Applications.
6. Chris Miles, Sushil J. Louis, (2006). ,,Towards the Co-Evolution of Influence Map Tree Based Strategy Game Players", Evolutionary Computing Systems Lab of Nevada.
7. SAI-Keung Wong, Shih-Wei Fang, (2012). ,,A Study on Genetic Algorithm and Neural Network for Mini-Games*" Journal of Information Science and Engineering 28, 145-159
8. Branquinho, A., and Lopes, (2010). ,,Planning for resource production in real-time strategy games based on partial order planning, search and learning", In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, 4205–4211.
9. libGDX biblioteka – Desktop/Android/iOS/HTML5 Java game development framework, [žiūrėta 2012-04-22]. Prieiga per internetą: http://libgdx.badlogicgames.com/documentation.html
10. Chong-U Lim, (2009). ,,An A.I. Player for DEFCON: An Evolutionary Approach Using Behavior Trees", Department of Computing Imperial College London, p. 30-46.
11. Ian Millington, (2006). ,,ARTIFICIAL INTELLIGENCE FOR GAMES". USA: Morgan Kaufmann Publishers is an imprint of Elsevier. p. 6-7. ISBN 13: 978-0-12-497782-2
12. Georgios N. Yannakakis, (2005). ,,AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation", Doctor of Philosophy Institute of Perception, p. 27.