KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

VYDŪNAS EIGIRDAS

VEIKĖJŲ VALDYMAS NAUDOJANT NEURONINĮ TINKLĄ IR GENETINĮ ALGORITMĄ

Magistro darbas

Darbo vadovas
lekt. dr. A. Noreika

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

VYDŪNAS EIGIRDAS

VEIKĖJŲ VALDYMAS NAUDOJANT NEURONINĮ TINKLĄ IR GENETINĮ ALGORITMĄ

Magistro darbas

Darbo vadovas:
lekt. dr. A. Noreika
2013-05-24

Recenzentas:
lekt. dr. L. Ablonskis
2013-05-24

Atliko:
IFM-1/1 gr. studentas
Vydūnas Eigirdas
2013-05-24

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

2013 - 05 - 24 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)

SANTRAUKA

Šiame darbe tiriama sritis yra kompiuteriniuose žaidimuose naudojamas dirbtinis intelektas. Konkrečiai gilinamasi į metodus, kurie valdo daugybę veikėjų žaidime, siekiančių tam tikro tikslo. Dėl konkretiems žaidimams unikalių mechanikų, šie metodai paprastai būna labai glaudžiai susiję su žaidimo aplinka ir taisyklėmis. Tyrimo tikslas yra sukurti ir ištirti metodą, skirtą daugybės veikėjų pajėgų valdymui ir jų veiksmų modeliavimui virtualioje aplinkoje.

Analizuojami metodai skirti pavienių veikėjų veiksmų įvertinimui ir modeliavimui, metodai skirti optimalių sprendimų žinių bazei sudaryti ir metodai toms žinioms pritaikyti paskirstant veikėjus aplinkoje. Pagal analizės rezultatus sukuriamas projektas daugelio veikėjų valdymui realiu laiku virtualioje aplinkoje. Lokalių veikėjų veiksmų modeliavimui naudojamas procedūrinis taktinių veiksmų parinkimo metodas. Veikėjų judėjimui aplinkoje modeliuoti naudojamas neuroninis tinklas. Jis apmokomas pagal genetiniu algoritmu sudarytus optimalius sprendimus.

Suprojektuota sistema realizuojama ir testuojama. Atliekamas eksperimentas su sistemos veikimo metu gautais rezultatais. Eksperimente nustatoma, kad šis sprendimo būdas gali tikslingai reaguoti į situacijas, susidarančias realaus laiko virtualioje aplinkoje, ir modeliuoti veikėjų veiksmus joje.

SUMMARY

The research area of this paper is artificial intelligence used in computer games. Specifically it is focused on methods for controlling a group of agents with a specific goal. Because of the uniqueness of individual game mechanics, those kinds of methods are usually closely related to that games environment and rules. The goal of this study is to design and test a method that could control a group of multiple agents in a virtual environment.

Methods for evaluating and selecting individual agent actions in a local environment, for gathering a database of optimal solutions and for applying that knowledge in distributing agents across the environment are analyzed. A design for controlling the actions of multiple agents in a real time virtual environment is designed, based on the results. Dynamic procedural combat tactics is used to model individual agent actions in a local environment. A neural network is used to model the movement of multiple agents in an environment. It is trained using optimal solutions, generated by a genetic algorithm.

Designed system is implemented and tested. Using data that the system generates, an experiment is conducted. It shows that this solution is capable of correctly reacting to situations, occurring in a real time virtual environment, and of modeling multiple agent actions in it.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Dirbtinis intelektas	(*angl. artificial intelligence*) tai mašinų arba programų parodomas intelektas.
Kelio taškų tinklas	(*angl. waypoint network*) duomenų struktūra, išreiškianti pozicijas, kurias gali užimti veikėjas, kaip tarpusavyje sujungtų taškų tinklą.
Sprendimų medis	(*angl. decision tree*) medžio tipo duomenų struktūra, skirta veikėjo veiksmų parinkimui.
Dalelių spiečiaus optimizacija	(*angl. particle swarm optimization*) iteratyvus stochastinis metodas spręsti optimizacijos problemas.
Dirbtinis neuroninis tinklas	(*angl. artificial neural network*) iš daugybės dirbtinų neuronų sudarytas matematinis modelis, paremtas biologiniais neuroniniais tinklais.
Aciklinis neuroninis tinklas	(*angl. feed-forward neural network*) tai neuroninis tinklas, kuriame neuronų jungtys tarpusavyje negali sudaryti ciklų.
Prižiūrimas mokymas	(*angl. supervised learning*) tai neuroninio tinklo apmokymo būdas, kurio metu kartu su užduotimi jis gauna ir jau paruoštą sprendimą. Tinklas turi išmokti išgauti tą sprendimą, kitą kartą gavęs tą pačią užduotį.
Atbulinė klaidos propagacija	(*angl. backpropagation*) metodas apmokyti neuroninį tinklą, kuris remiasi tinklo klaidos įvertinimu ir paskirstymu tarp jo briaunų.

ĮVADAS

Dirbtinis intelektas yra viena naujausių mokslo sričių, pradėta vystyti tik po antrojo pasaulinio karo. Terminas „Dirbtinis intelektas“ (*angl. Artificial intelligence*) pirmą kartą pavartotas 1956 metais. Šiuo metu tai viena perspektyviausių mokslo sričių, kadangi jos didieji atradimai dar laukia ateityje.Viena sparčiausiai besivystančių šios mokslo srities dalių yra kompiuterinių žaidimų dirbtinis intelektas. Pastaruosius trisdešimt metų kompiuterinių žaidimų progresas akivaizdžiai atspindėjo augančias technologines galimybes. Taip pat tobulėjo ir žaidimų veikėjų dirbtinis intelektas. Iš paprastų, viena kryptim judančių geometrinių figūrų išaugo veikėjai kurie slepiasi nuo kulkų už mūrinių sienų, išsigandę pabėga, įgavę pranašumą puola, planuoja savo veiksmus ir mokosi iš savo klaidų.

Darbo problematika ir aktualumas

Darbe tiriama sritis yra kompiuteriniuose žaidimuose naudojamas dirbtinis intelektas. Konkrečiai gilinamasi į metodus, kurie valdo didelius veikėjų būrius žaidime, siekiančius tam tikro tikslo. Žaidimo aplinkos ir galimų veikėjų veiksmų joje specifika paprastai yra unikali ir naudojama tik tame žaidime. Todėl tokio pobūdžio metodai taip pat kuriami ir derinami konkrečiam žaidimui. Dėl šios priežasties daugelis tokių metodų yra labai priklausomi nuo veiksmų parinkimo pagal iš anksto paruoštus scenarijus. Šis darbas orientuojasi į metodo sukūrimą, kuris veikimo metu procedūriškai kūrtų veikėjų valdymo realaus laiko virtualioje aplinkoje strategiją, ir papildytų ją, atsižvelgdamas į situacijos pokyčius.

Darbo tikslas ir uždaviniai

Šio darbo tikslas yra sukurti ir ištirti metodą, skirtą veikėjų pajėgų valdymui ir veikėjų veiksmų modeliavimui virtualioje aplinkoje. Darbui keliami tokie uždaviniai:

1) Analizės metu pasirinkti metodą lokaliam veikėjų veiksmų parinkimui atsižvelgiant į aplinką ir jį realizuoti.
2) Analizės metu pasirinkti metodą veikėjų išsidėstymui visoje aplinkoje valdyti ir jį realizuoti.
3) Atlikus eksperimentą įvertinti realizuoto metodo darbą.

Darbo rezultatai ir jų svarba

Darbe sukuriamas metodas realiu laiku valdantis daugelio veikėjų pajėgas virtualioje aplinkoje. Metodas naudoja neuroninį tinklą sprendimams priimti, ir genetinį algoritmą, žinių bazei skirtai neuroninio tinklo apmokymui sukaupti. Atliekamas eksperimentas, kurio metu ištiriami neuroninio tinklo pasiekti rezultatai žaidimo kontekste. Darbo rezultatai parodo, kad neuroninis tinklas gali būti pritaikytas daugelio veikėjų valdymui virtualioje aplinkoje.

Darbo struktūra

Šį dokumentą sudaro penkios dalys. Analizės dalyje analizuojami esami dirbtinio intelekto sprendimai ir parenkami darbe naudojami metodai. Sistemos projektą aprašančioje dalyje nustatomi kuriamos sistemos reikalavimai ir kokybės kriterijai. Aprašomas sistemos projektas. Realizaciją aprašančioje dalyje apžvelgiamas realizuotos sistemos veikimo eigos procesas. Eksperimentinį sistemos tyrimą aprašančioje dalyje ištiriamas realizuoto veikėjų valdymo metodo darbas ir įvertinami gauti rezultatai. Visas atliktas darbas apibendrinamas išvadų dalyje.

1. PROBLEMINĖS SRITIES ANALIZĖ

1.1. Analizės tikslas

Analizės metu bandoma išsiaiškinti kokiais būdais valdomi veikėjai realaus laiko virtualioje aplinkoje. Tiriama kokiais būdais veikėjams nurodomas bendras tikslas ir parenkami veiksmai jo siekti. Gilinamasi į duomenų struktūras reikalingas intelekto orientavimuisi virtualioje aplinkoje, bei galimybes realiu laiku modeliuoti valdomų veikėjų veiksmus.

Pagal atliktą analizę bus projektuojama sistema skirta veikėjų valdymui kompiuteriniame žaidime.

1.2. Tyrimo objektas, sritis ir problema

Tyrimo objektas yra daugybės veikėjų veiksmų parinkimas realaus laiko virtualioje aplinkoje, siekiant bendro tikslo.

Aplinkoje esantys veikėjai skirstomi į dvi priešiškas puses. Kiekviena pusė turi savo pradinę poziciją. Veikėjai, pradedami nuo šio taško, bando pasiekti priešininkų pusės pradžios poziciją. Tuo pačiu metu jie turi stengtis neleisti priešininkams padaryti to paties. Laimi pusė pirmoji pasiekusi priešininkų saugomą poziciją.

Planuojama sukurti metodą, kuris atsižvelgdamas į veikėjų būsenas virtualioje aplinkoje, modeliuotų jų veiksmus, siekdamas nuvesti juos iki priešiškų pajėgų pradžios taško, neleidžiant priešininkams pasiekti sąvojo.

1.3. Tyrimo planas

1) Analizės metodų pasirinkimas.
2) Bendrojo pobūdžio literatūros apie dirbtinį intelektą kompiuteriniuose žaidimuose analizė.
3) Informacijos apie jau realizuotus sprendimus rinkimas ir analizė.
4) Mokslinių straipsnių analizavimas siekiant surasti sprendimui tinkamus metodus ir algoritmus.
5) Mokslinės literatūros analizavimas siekiant įsigilinti į sprendimui tinkamų metodų bei algoritmų realizavimą.
6) Reikalavimų veikėjų veiksmus realaus laiko virtualioje aplinkoje modeliuojančiai sistemai sudarymas ir analizė.
7) Sistemos projekto sudarymas.
8) Sistemos projekto realizavimas.
9) Sistemos testavimas ir gautų duomenų analizė.
10) Rezultatų apibendrinimas.

1.4. Analizės tikslas

Analizės metu tiriami šiuolaikiniuose kompiuteriniuose žaidimuose naudojami metodai veikėjų dirbtiniam intelektui realizuoti ir su jais susijusios duomenų struktūros. Atsižvelgiama į šių metodų taikymo privalumus ir trūkūmus specifinėse aplinkose, modeliuojant konkrečius veikėjų veiksmus. Naudojantis analizės metu surinkta informacija bus modeliuojami įvairūs kuriamos sistemos aspektai.

1.5. Analizės metodai

Analizėje apžvelgiami metodai ir duomenų struktūros naudojami modeliuoti veikėjų veiksmus šiuolaikiniuose kompiuteriniuose žaidimuose. Vertinami galimi jų pritaikymo būdai kuriamoje sistemoje. Tyrime naudojamas teorinės analizės ir apibendrinimo metodas.

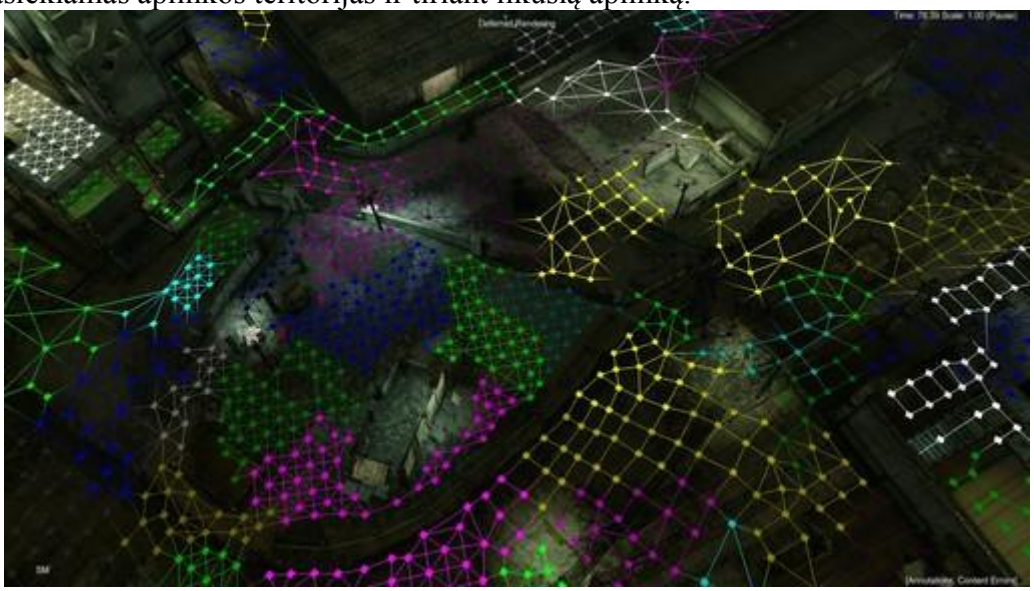
1.6. Esamų sprendimų analizė

1.6.1. Duomenų struktūrų, skirtų veikėjų dirbtiniam intelektui modeliuoti, analizė

Norint veikėjų dirbtiniam intelektui suvokti jį supančią aplinką ir realiu laiku priimti logiškus sprendimus reikalingos tą aplinką apibūdinančios duomenų struktūros. Šios duomenų struktūros gali būti sukuriamos kartu su aplinka, generuojamos prieš paleidžiant sistemą arba perskaičiuojamos realiu laiku. Paprastai kiekvienai kuriamai sistemai pritaikomos unikalios duomenų struktūros, skirtos konkrečios aplinkos atvaizdavimui. Toliau pateikiami keli pavyzdiniai tokių strukūrų tipai:

1.6.1.1. Kelio taškų tinklas

Kelio taškų tinklo (*angl. waypoint network*) duomenų struktūra atvaizduoja kiekvieną poziciją aplinkoje, kurią veikėjas gali potencialiai užimti [1]. Ši duomenų struktūra paprastai yra sudaroma rankiniu būdu, tačiau ją galima ir sugeneruoti prieš pradedant sistemos darbą. Tai atliekama atmetant uždaras, nepasiekiamas aplinkos teritorijas ir tiriant likusią aplinką.

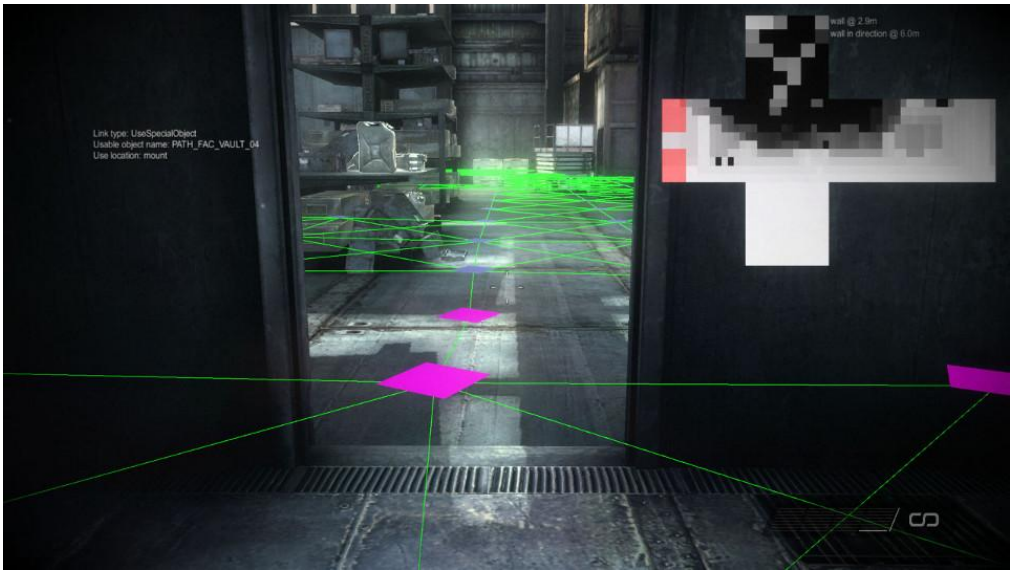


Pav. 1.1 Kelio taškų tinklas žaidime „Killzone 2“

Papildomai prie šių pozicijų gali būti prijungiamos kitos duomenų struktūros skirtos optimizuoti specifines dirbtinio intelekto funkcijas (pvz.: matomumo nustatymą) arba susijusios su logišku veikėjų sprendimų planavimu (pvz.: strateginės pozicijos vertės nustatymu, kelio parinkimu).

1.6.1.2. Specifinės duomenų struktūros matomumui sudėtingoje trimatėje aplinkoje nustatyti

Veikėjų matomumo nustatymas sudėtingoje trimatėje aplinkoje yra labai daug resursų reikalaujantis uždavinys. Todėl norint jį realizuoti realaus laiko sistemoje reikalingos tą procesą supaprastinančios duomenų struktūros. Tokios struktūros paprastai prijungiamos prie judėjimą trimatėje aplinkoje ribojančių duomenų struktūrų (pvz.: kelio taškų tinklo). Dėl aplinkų ir veikėjų skirtumų šių duomenų struktūrų realizacijos būna labai įvairios.



Pav. 1.2 Kubiniai žemėlapiai matomumui nustatyti žaidime „Killzone 2“

Pavyzdžiui žaidime „Killzone 2“ sudarant kelio taškų tinklą, prie kiekvienos viršūnės papildomai prijungiamas iš anksto sugeneruotas kubinis žemėlapis [2]. Šis žemėlapis nurodo matoma atstumą kiekviena kryptimi iš tos viršūnės. Paprastesnis metodas naudojamas žaidime „Dark Sector“ yra kiekvienoje kelio taškų tinklo viršūnėje prijungta dviejų aukščių cilindrinė lentelė, nurodanti apytikslį matomumą įvairiais kampais.

1.6.1.3. Specifinės duomenų struktūros veikėjų sprendimų priėmimui sudėtingoje trimatėje aplinkoje

Sudėtingos dirbtinio intelekto sistemos modeliuodamos veikėjo veiksmus bando atvaizduoti veikėjo mąstymą. Veikėjas nebūtinai renkasi trumpiausią kelią tarp dviejų pozicijų, o ieško optimalesnio kelio, kuriuo keliaudamas jis bus saugesnis ar užsiims geresnes strategines pozicijas kovos lauke. Kadangi veikėjų mąstymas yra labai specifinis konkretiems žaidimams, tokių sistemų realizacija beveik visada yra unikali, o joms įgyvendinti pasitelkiama daugybė įvairių rūšių duomenų struktūrų. Priklausomai nuo jos sudėtingumo tokia sistema gali atsižvelgti į priešiškų veikėjų pozicijas, matomumą iš galimų pozicijų, aplinkos geometrijos suteikiamą taktinį pranašumą ir t.t [1].

Pav. 1.3 Veikėjų sprendimų priėmimas kovos metu žaidime „Halo“

Žaidime „Halo“ yra sudaromas papildomas grafas kuris realiu laiku nurodo priešininkų pozicijų būsenas, atsižvelgdamas į žaidėjo buvimo vietą, aplinkos geometriją ir veikėjo galimybes.

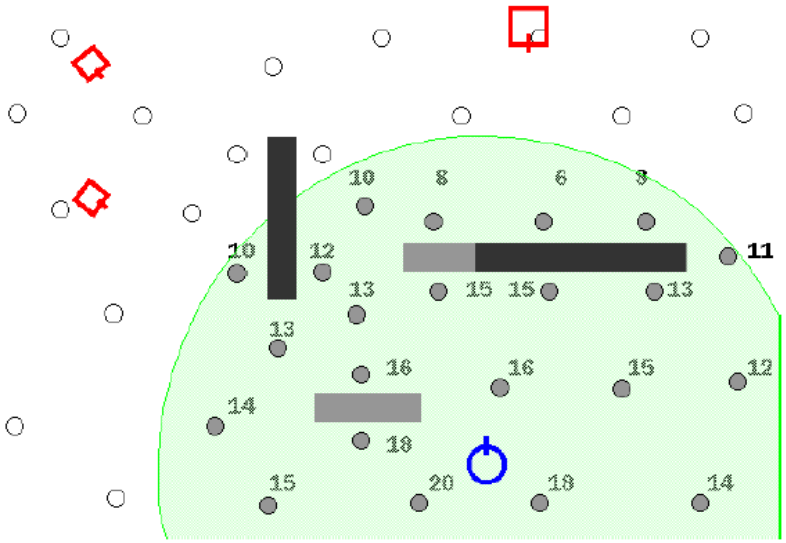
1.6.2. Veikėjo veiksmų parinkimas iš sprendimų medžio

Vienas būdų parinkti veikėjo veiksmus yra sprendimų medžio (*angl. decision tree*) sudarymas. Tai yra medžio tipo duomenų struktūra, kurioje kiekviena viršūnė išreiškia tam tikrą sąlygą, kurią reikia įvertinti [3]. Iš tos viršūnės išeinančios briaunos išreiškia galimus sąlygos rezultatus. Apatinės medžio viršūnės, vadinamos lapais, išreiškia galimus veikėjo veiksmus. Pradedant nuo pirmosios medžio viršūnės, dar vadinamos šaknimi, pagal viršūnėse esančias sąlygas vertinama veikėjo padetis ir leidžiamasi medžiu žemyn, iki pasiekiamas lapas. Tada įvykdomas tame lape nurodytas veiksmas.

Paprastą tokio pobūdžio duomenų struktūrą nesunku realizuoti, tačiau didėjant pasirenkamų veiksmų ir sąlygų kiekiui šie medžiai gali tapti labai sudėtingi. Taip pat jie gali būti apmokomi.

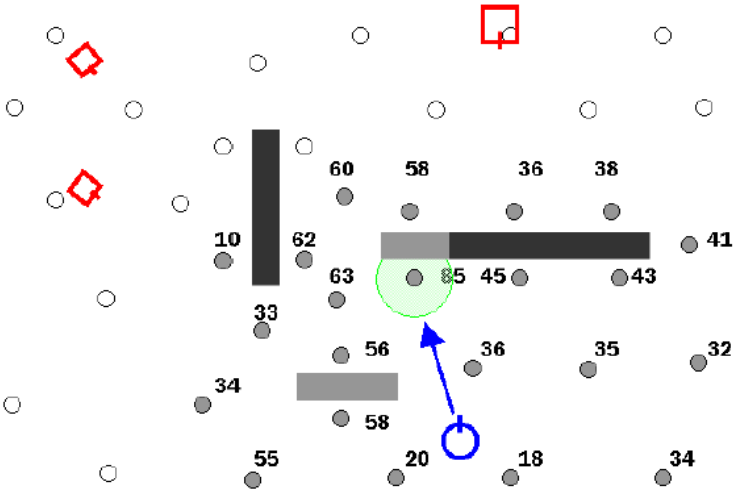
1.6.3. Procedūrinis taktinių veiksmų parinkimas

Šis metodas realiu laiku parenka veikėjo veiksmus, atsižvelgdamas į aplinką ir veikėjo modelį. Jis remiasi aplinkos sudalinimu į galimų pozicijų grafą ir tų pozicijų įvertinimu [1]. Atsižvelgdamas į veikėjo esamą poziciją, jo būseną, priešininkų pozicijas, sąjungininkų pozicijas, ugnies linijas, atstumus iki priešininkų ir panašius parametrus metodas priskiria tam tikrus koeficientus kiekvienam pozicijų grafo taškui, esančiam netoli veikėjo.



Pav. 1.4 Vertinamos ne toli veikėjo esančios pozicijos

Įvertinęs veikėjo užimamą poziciją ir pagal koeficientus atrinktą geriausią poziciją, metodas nustato pozicijos keitimo naudos koeficientą. Pagal veikėjo esamą būseną ir užimamą poziciją metodas nustato naudos koeficientus kitiems galimiems veikėjo veiksmams ir palyginęs juos pasirenka sekantį veikėjo veiksmą.



Pav. 1.5 Pagal suteiktus koeficientus nustatoma geriausia taktinė pozicija

Šis metodas patogus tuo, kad prie jo nesunkiai galima priderinti papildomas žaidimo mechanikas (pvz.: sprogimo zonų vengimą, specifinių veikėjo sugebėjimų susijusių su konkrečia pozicija naudojimą ir pan.). Reikia tiktai į pozicijų koeficientų skaičiavimą įtraukti šių mechanikų koeficientus.

1.6.4. Dalelių spiečiaus optimizacija

Dalelių spiečiaus optimizacija (*angl. particle swarm optimization*) yra stochastinis metodas spręsti optimizacijos problemas [4]. Jis veikia iteratyviai bandydamas gerinti turimą sprendimą, įvertintą tam tikra kokybės funkcija.

Sugeneruojami atsitiktiniai sprendimai, vadinami dalelėmis. Visų sugeneruotų dalelių visuma vadinama populiacija. Kiekvienos dalelės siūlomas sprendimas įvertinamas kokybės funkcija. Dalelės pradinė pozicija sprendimų erdvėje nustatoma kaip tos dalelės lokalus minimumas p_{id}. Jai judant, šis skaičius visada nurodys geriausiai įvertintą poziciją, kurioje ta dalelė yra buvus. Iš visų dalelių lokalių minimumų išrenkama geriausiai įvertinta pozicija ir nustatoma kaip globalus

minimumas p_{gd}. Algoritmui veikiant, kiekvienoje iteracijoje atsižvelgus į dalelės lokalų ir bendrą globalų minimumus, pagal paprastą matematinę formulę nustatomas jos greičio vektorius:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}); \quad (1.1)$$

Čia c_1 ir c_2 yra teigiamos konstantos vadinamos akseleracijos koeficientais, r_1 ir r_2 yra atsitiktiniai skaičiai intervale [0,1), i nurodo konkrečią dalelę, d nurodo sprendimų erdvės dimensiją, o x_{id} nurodo esamą dalelės poziciją. Greičio vektoriumi papildoma dalelės pozicija:

$$x_{id} = x_{id} + v_{id}; \quad (1.2)$$

Algoritmas veikia tol, kol surandamas iš anksto nustatytą kriterijų tenkinantis sprendimas.

1.6.5. Genetinis algoritmas

Ieškant optimalaus sprendimo konkrečioje situacijoje, gali būti naudojamas genetinis algoritmas. Tai yra euristinis paieškos algoritmas, kuris atkartoja natūralios evoliucijos procesus [5].

Sprendimas išreiškiamas skaičių eile, vadinama chromosoma. Atsitiktiniu būdu sudaromas tam tikras kiekis pradinių chromosomų, kurios vadinamos populiacija. Kiekvienos jų saugomas sprendimas įvertinamas naudojant tinkamumo funkcija ir jam suteikiamas tinkamumo koeficientas. Baigus vertinti visas pradines chromosomas, iš jų pasirenkamos dvi kryžminimui. Pasirinkimas yra atsitiktinis, tačiau galimybė išrinkti konkrečią chromosomą yra tiesiogiai proporcinga jai suteiktam tinkamumo koeficientui. Kryžminant chromosomas, jos skeliamos į dvi dalis atsitiktinai pasirinktoje vietoje. Vienos chromosomos pirma dalis sujungiama su kitos antraja dalimi, ir taip sudaroma nauja chromosoma. Gautoji chromosoma pridedama prie naujos populiacijos ir kryžminimo procesas kartojamas. Sudarius naują chromosomą jai papildomai gali būti suteikta galimybė mutuoti. Mutacijos metu , vienas chromosomą sudarančių skaitmenų gali būti atsitiktinai pakeičiamas.

Tokiu būdu sudarinėjamos naujos chromosomos, iki pasiekiamas tam tikras chromosomų skaičius, arba naujos chromosomos tinkamumo koeficientas pasiekia iš anksto nustatytą tinkamumo įvertį. Baigus generavimą, iš populiacijos išrenkama chromosoma su didžiausiu tinkamumo koeficientu ir naudojama sprendime. Žemiau pateikiamas kodas (žr. Genetinis-Algoritmas).

```
Function Genetinis-Algoritmas
// in: populiacija - aibė atsitiktinai sugeneruotų chromosomų
// in: tinkamumo-Fn – funkcija, kuri pamatuoja chromosomos tinkamumą
// out: optChr – geriausiai įvertinta chromosoma

repeat
   nauja_populiacija = tuščia populiacija;
   loop for i from 1 to size(populiacija) do
      x = Atsitiktinai-Parenkamas(populiacija, tinkamumo-Fn);
      y = Atsitiktinai-Parenkamas(populiacija, tinkamumo-Fn);
      rezultatas = Kryžminti(x, y);
      if (maža atsitiktinė tikimybė) then
         Mutuoti(rezultatas);
      endif;
      nauja_populiacija.Pridėti(rezultatas);
   endloop;
   populiacija = nauja_populiacija;
until rezultato tinkamumas pasiekia tam tikrą ribą, arba peržengiama iteracijų riba;
return optChr - geriausiai tinkamumo-Fn įvertintą chromosoma, esanti populiacijoje;
```

```
Function Kryžminti
// in: x - chromosoma
// in: y – chromosoma
// out: rezultatas – chromosoma, gauta sukryžminus x ir y

n = length(x);
c = atsitiktinis skaičius nuo 1 iki n;
return Substring(x,1,c) + Substring(y,c+1,n);
```

1.6.6. Dirbtinis neuroninis tinklas

Dirbtinis neuroninis tinklas (*angl. artificial neural network*), tai matematinis modelis paremtas biologiniais neuroniniais tinklais [6]. Jis susideda iš tarpusavyje sujungtų dirbtinių neuronų. Šis tinklas dažniausiai naudojamas kaip adaptyvi sistema, galinti išmokti sudėtingus ryšius tarp konkrečių įvedamų ir išvedamų duomenų.

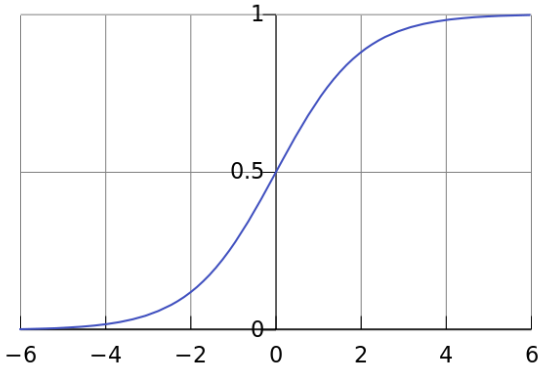
Dirbtinį neuroninį tinklą sudaro aibė mazgų, sujungtų orientuotomis briaunomis. Briauna iš mazgo *j* į mazgą *i* perduoda aktyvacijos funkcijos apskaičiuotą sumą a_j. Kiekvienai briaunai taip pat priskiriamas svoris $W_{j,i}$, kuris nurodo jungties įtaką. Kiekvienas mazgas *i* pirmiausia susumuoja visų į jį ateinančių briaunų ir papildomo šališkumo koeficiento $W_{0,i}$ įvestį in_i:

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j \; ; \tag{1.3}$$

Gautą sumą neuronas perleidžia per aktyvacijos funkciją ir gauna išvestį. Ši funkcija nustato ar neuronas yra aktyvus (grąžiną rezultatą artimą 1), ar neaktyvus (grąžiną rezultatą artimą 0). Dažniausiai naudojamos slenksčio ir sigmoido funkcijos. Slenksčio funkcija tiesiog grąžiną 1, kai įvesčių suma peržengia nulinę ribą, ir 0, kai suma yra mažesnė už 0. Sigmoido funkcija pranašesnė, nes yra diferencijuojama. Ji apskaičiuojama pagal šią formulę:

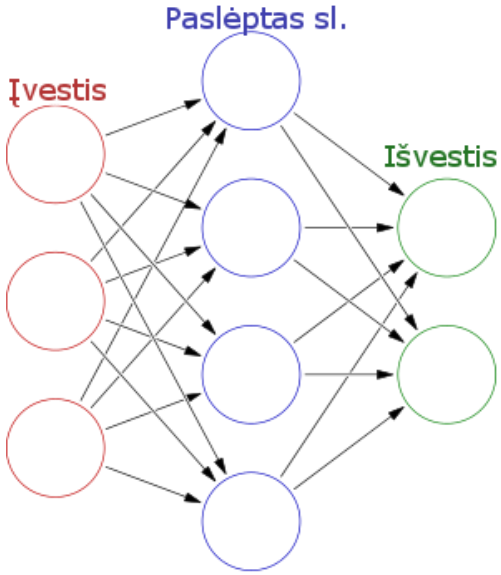
$$S(t) = \frac{1}{1+e^{-t}} \; ; \tag{1.4}$$

Šios funkcijos rezultatas atvaizduojamas kreive:



Pav. 1.6 Sigmoido kreivė

Dirbtiniai neuroniniai tinklai skirstomi į aciklinius (*angl. feed-forward*) ir ciklinius arba rekurentinius tinklus. Aciklinis tinklas nesaugo savo vidinės būsenos ir veikia kaip jam suteiktos įvesties funkcija. Tokie neuroniniai tinklai paprastai suskirstomi į sluoksnius, taip kad kiekvienas iš eilės einantis sluoksnis priimtu įvestį tik iš sluoksnio einančio prieš jį, o išvestį perduotų tik sluoksniui einančiam po jo.



Pav. 1.7 Vieno paslėpto sluoksnio aciklinis neuroninis tinklas

Pirmas sluoksnis gauna pirminius įvesties parametrus ir yra vadinamas įvesties sluoksniu. Mazguose, priklausančiuose šiam sluoksniui, įvestis paprastai neperskaičiuojama aktyvacijos funkcija, o perduodama nepakeista per mazgą jungiančias briaunas į sekantį sluoksnį. Paskutinis neuroninio tinklo sluoksnis yra vadinamas išvesties sluoksniu. Jis sudaro neuroninio tinklo išvestį, pagal į įvesties sluoksnį įvestus duomenis. Tarp pirmo ir paskutinio sluoksnių gali būti tam tikras skaičius paslėptų sluoksnių. Jei paslėptų sluoksnių nėra, toks tinklas vadinamas vieno sluoksnio neuroniniu tinklu arba perceptroniniu tinklu.

Tinklas su paslėptais sluoksniais vadinamas daugelio sluoksnių neuroniniu tinklu. Paslėptų sluoksnių suteikiamas privalumas yra tas, kad jie padidina hipotezių erdvę kurią tinklas gali atvaizduoti.

Kita neuroninių tinklų rūšis - rekurentinis tinklas. Jame jungtys tarp neuronų suformuoja vienkrypčius ciklus. Tai juose sudaro tam tikras vidines būsenas, kurios gali tinklams suteikti laikiną dinaminę elgseną. Dėl šios priežasties rekurentinių tinklų tyrimas yra daug sudėtingesnis procesas, nei aciklinių tinklų tyrimas.

Pati naudingiausia dirbtinių neuroninių tinklų sąvybė, yra jų gebėjimas mokytis. Procesas, apmokantis dirbtinį neuroninį tinklą iš gautų duomenų išgauti jau iš anksto žinomą rezultatą vadinasi prižiūrimu mokymusi (*angl. supervised learning*). Pirmiausia nustatoma įvesties būsena, ir naudojant papildomus algoritmus tai būsenai surandamas optimalus sprendimas. Sudaroma aibė įvesties būsenų ir jas atitinkančių sprendimų. Tada, naudojant atbulinės klaidos propagacijos metodą, dirbtinis neuroninis tinklas yra apmokomas aibėje esančioms įvesties būsenoms pateikti susietus sprendimus.

Gavęs nepažystamus įvesties duomenis, dirbtinis neuroninis tinklas pateiks tarpinį sprendimą, gautą pagal artimiausius įvesties duomenis kuriems buvo apmokytas. Dėl to jis gali būti naudojamas situacijose, kur įvesties duomenims būdingi triukšmai ir jie negali būti tiksliai apibrėžti.

1.6.7. Atbulinės klaidos propagacijos metodas

Dirbtiniam neuroniniam tinklui apmokinti dažniausiai naudojamas atbulinės klaidos propagacijos metodas (*angl. Backpropagation*) [5]. Jam atlikti reikalinga aibė įvesčių susietų su joms optimaliais sprendimais. Visų pirma dirbtinio neuroninio tinklo briaunų svoriams suteikiamos atsitiktinės reikšmės. Tada į jį įvedama viena turimų įvesčių, ir gaunamas išvesties sluoksnio rezultatų vektorius su atsitiktinėmis reikšmėmis. Kiekviena šio vektoriaus reikšmė, atitinkanti mazgą i, lyginama su atitinkama reikšme šiai įvesčiai priskirtame sprendimo vektoriuje ir gaunamas skirtumas tarp jų, kuris žymimas Err_i. Visų reikšmių skirtumai pakeliami kvadratu ir susumuojami. Gautas skaičius vadinamas bendra klaidų kvadratų suma, ir yra naudojamas neuroninio tinklo teisingumui vertinti. Kiekvienam išvesties mazgui papildomai apskaičiuojamas klaidos pokytis Δ_i.

$$\Delta_i = Err_i \times g'(in_i); \quad (1.5)$$

Čia $g'(in_i)$ yra sigmoido funkcijos išvestinė. Į mazgą įeinantys svoriai $W_{j,i}$ perskaičiuojami taip:

$$W_{j,i} = W_{j,i} + \alpha \times a_j \times \Delta_i; \quad (1.6)$$

Čia a_j yra prieš šį einančio sluoksnio mazgo j išvestis, o α yra pasirenkamas mokymo greitis. Norint papildyti briaunų svorius tarp prieš tai einančių neuroninio tinklo sluoksnių, reikia paslėpto sluoksnio mazgams nustatyti kiekius analogiškus išvesties sluoksnio mazgų klaidų skirtumams Err_i. Žinant, kad mazgas j įtakoja tam tikrą dalį klaidos Δ_i, kiekviename sekančio sluoksnio mazge su kuriuo yra sujungtas, Δ_i vertės išdalinamos pagal briaunų svorius ir perduodamos atgal į prieš tai einantį sluoksnį, Δ_j verčių sudarymui. Δ reikšmės apskaičiuojamos šia formule:

$$\Delta_j = g'(in_j)\sum_i W_{j,i}\Delta_i; \quad (1.7)$$

Briaunų svoriai perskaičiuojami pagal šią formulę:

$$W_{k,j} = W_{k,j} + \alpha \times a_k \times \Delta_j; \quad (1.8)$$

Procesas kartojamas iki bendra klaidų kvadratų suma yra mažesnė negu iš anksto nustatyta riba. Po to pateikiamas kodas (žr. Atbulinė-Klaidos-Propagacija).

```
Function Atbulinė-Klaidos-Propagacija
// in: pavyzdžiai - aibė įvesčių vektorių x, su jiems atitinkančių optimalių sprendimų vektoriais y
// out: tinklas – daugelio sluoksnių tinklas turintis L sluoksnių, su svoriais Wj,i ir aktyvacijos funkcija g

repeat
   for each e in pavyzdžiai do
      for each mazge j įvesties sluoksnyje do
         aj = xj[e];
      endfor;
      for l = 2 to L  do
         ini = ∑j Wj,i aj;
         ai = g(ini);
      endfor;
      for each mazge i išvesties sluoksnyje  do
         Δi = g'(ini)×( yi[e]-ai);
      endfor;
      for l = L-1 to 1 do
         for each mazge j sluoksnyje l do
            Δj = g'(inj)∑i Wj,i Δi;
            for each mazge i sluoksnyje l+1 do
               Wj,i = Wj,i + α × aj × Δi;
            endfor;
         endfor;
      endfor;
   endfor;
until pasiekiama pakankamai maža tinklo paklaida;
return tinklas;
```

1.7. Siekiamas sprendimas

Daugybės veikėjų valdymui virtualioje aplinkoje realiu laiku nėra vieno optimalaus sprendimo. Kiekvieno kompiuterinio žaidimo specifika reikalauja konkrečiai tam žaidimui pritaikytų, unikalių sprendimų priėmimo būdų.

Siekamas sprendimas yra suprojektuoti ir sukurti sistemą, kuri valdytų daugelio veikėjų pajėgas kompiuteriniame žaidime. Žaidimo esmė yra veikėjams pasiekti priešininkų saugomą tašką, tuo pačiu metu apginant sąvajį. Siekdama šių tikslų, sistema turėtų valdyti veikėjų išsidėstymą virtualioje aplinkoje.

1.8. Analizės išvados

1) Analizės metu apžvelgti duomenų struktūrų panaudojimo būdai realizuojant sistemą skirtą veikėjų veiksmų modeliavimui realaus laiko virtualioje aplinkoje.
2) Ištirtos šiuolaikiniuose kompiuteriniuose žaidimuose esančios veikėjų veiksmų modeliavimo sistemos ir jų galimybės.
3) Apžvelgti būdai, skirti nustatyti optimalius sprendimus konkrečiose situacijose ir išmokyti dirbtinį intelektą priimti adekvačius toms situacijoms sprendimus, realiu laiku.

2. SISTEMOS PROJEKTAS

2.1. Funkciniai reikalavimai

Modeliuojant veikėjų veiksmus realiu laiku sistemai išskiriami tokie funkciniai reikalavimai:

- Sistema turi atvaizduoti aplinką tarpusavyje sujungtų zonų grafu. Kiekviena zona turi turėti duomenų struktūras apibrėžiančias jos ryšius su aplinkinėmis zonomis.
- Sistema turi modeliuoti neuroninį tinklą, kuris, atsižvelgdamas į aplinkos sąvybes, bei žemėlapyje esančių veikėjų būklę, valdytų jų pasiskirstymą aplinkoje.
- Duomenys skirti neuroninio tinklo apmokymui turi būti paruošti naudojant genetinį algoritmą.

2.2. Nefunkciniai reikalavimai

- Sistemos dalis, modeliuojanti veikėjų veiksmus turi veikti realiu laiku.
- Sistema turi veikti Windows operacinės sistemos aplinkoje.
- Sistema neturi naudoti komercinės programinės įrangos.

2.3. Siekiami kokybės kriterijai

Sistemos darbo kokybę galima įvertinti šiais kriterijais:

- Papildomų apmokymo iteracijų žaidime suteikiamas pranašumas.
- Veikėjų logiškas paskirstymas žemėlapyje.
- Greitas situacijos įvertinimas ir sprendimų priėmimas.

2.4. Sistemos pagrindimas ir esmės išdėstymas

Kuriama sistema skirta valdyti veikėjų išsidėstymą ir judėjimą kompiuteriniame žaidime. Žaidimo aplinka yra supaprastinta ir išreikšta grafu. Grafo viršūnės nurodo aplinkoje išskirtas zonas. Tai nedideli plotai , kurie jungiasi tarpusavyje. Grafe gretimas zonas nurodančios viršūnės sujungiamos briauna. Taip pat zonoms suteikiamos duomenų struktūros, nurodančios jų ryšius su aplinkinėmis zonomis. Veikėjus sistema atvaizduoja kaip figūras judančias tarp grafo viršūnių. Jų tarpusavio sąveika yra aproksimuojama pagal veikėjų būklę ir aplinkos parametrus.

Yra keturi veikėjų tipai. Nuo tipo priklauso veikėjo žalos potencialas vertinant jo būseną aplinkoje. Jis pasako, kokiu atstumu nuo taikinio veikėjas turi galimybę padaryti priešininkui optimalią žalą. Iš veikėjų sudaromi būriai. Sistemos veikimo metu pajėgos papildomos vis naujais būriais. Vienos pusės pajėgų sudėtis nurodoma faile. Tame pačiame faile nurodomos taisyklės kas kiek laiko ir kokios sudėties būriai papildo pajėgas.

Sistemai nurodomi keliai, einantys per žaidimo aplinką išreiškiantį grafą. Veikėjų judėjimui šiais keliais valdyti sukuriamas neuroninis tinklas, kuris įvertina susidariusią situaciją kiekviename iš kelių, bei bendrus žemėlapio parametrus ir suteikia kiekvienam keliui tam tikrą koeficientą. Pagal šį koeficientą pasirenkami veikėjų veiksmai tame kelyje.

Neuroniniam tinklui apmokyti naudojamas genetinis algoritmas. Jis sudaro chromosomas, kurios kiekvienam keliui nurodo veikėjų valdymui skirtą koeficientą. Pagal šiuos koeficientus modeliuojami veikėjų veiksmai. Juos atlikus specifinė funkcija įvertina pokyčius žemėlapyje ir suteikia jiems tam tikrą skaitinę vertę. Genetinis algoritmas generuoja naujas chromosomas iki pasiekia tam tikrą ribą. Tada geriausiai įvertinta chromosoma suporuojama su situaciją apibrėžiančiais parametrais ir išsaugoma dirbtinio neuroninio tinklo apmokymui.

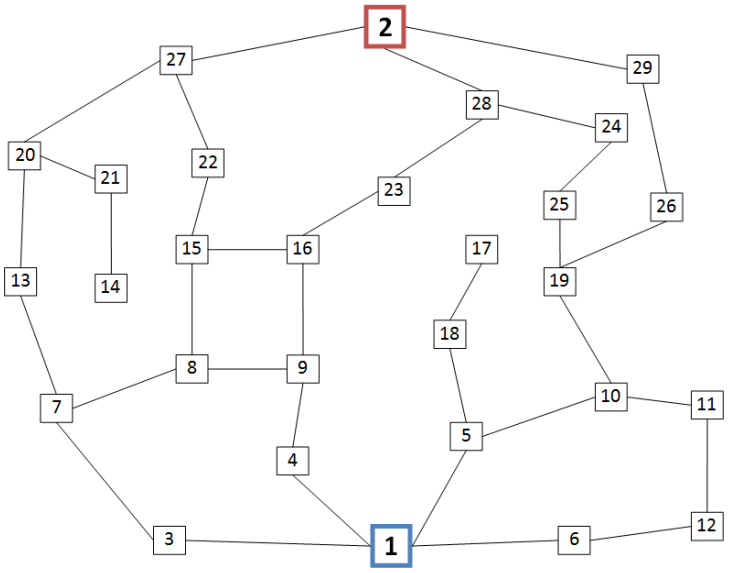
Neuroninis tinklas naudojamas sprendimų priėmimui realiu laiku. Jam apmokyti naudojamos genetinio algoritmo sugeneruotų sprendimų ir juos atitinkančių būsenų poros. Neuroninio tinklo sprendimų kokybė priklauso nuo jo apmokymui panaudotų porų kiekio.

2.5. Sistemos architektūra

2.5.1. Žemėlapio duomenys

Žemėlapio duomenys yra skirti aproksimuoti detalų trimatį pasaulį į zonas ir atvaizduoti jas grafu. Gretimas zonas grafe jungia briaunos. Kiekvienai zonai priskiriami parametrai skirti aproksimuoti jos sąvybes:

- Zonos talpa nurodo kiek veikėjų gali tilpti zonoje vienu metu.
- Zonos jungčių sąrašas nurodo visas zonas pasiekiamas iš šios zonos, atstumus iki jų ir pralaidumo koeficientus. Pralaidumo koeficientai nurodo jungčių pralaidumą, ir yra naudojami skaičiuojant per kokį laiką konkretus veikėjų būrys pereis iš vienos zonos į kitą.
- Zonos ryšių sąrašas nurodo visas aplinkines zonas su kuriomis ši zona turi ryšį. Ryšiai nurodo kiekvienos rūšies veikėjo žalos potencialą tarp dviejų zonų ir yra naudojami žalos apskaičiavimui, vykstant susirėmimui tarp dviejų būrių.
- Potenciali žala, nurodo kokią potencialią žalą gali padaryti priešininkų pajėgos šioje zonoje duotuoju momentu. Šis parametras naudojamas įvertinti zonos perspektyvas keičiant būrių pozicijas ir yra perskaičiuojamas realiu laiku.



Pav. 2.1 Grafo pavyzdys

Kartu su žemėlapiu taipogi pateikiami keliai per zonas. Jie nurodo kokiomis kryptimis grafe gali judėti veikėjų būriai. Keliai paprastai veda nuo pradžios taško iki tikslo, tačiau gali nuvesti ir į akligatvius. Visi keliai gali būti atvaizduojami lentele (žr. Lent. 2.1).

Lentelė 2.1 Kelių lentelės pavyzdys

Tikslas	T (2)												
Kelio viršūnės: T-tikslas A-aklig-atvis			27						28			T	
			20		T				24			28	T
		A	21	T	28	T			25	T		24	29
	T	14	14	27	23	27	T		19	28	T	25	26
	27	21	A	22	16	22	28		10	24	29	19	19
	20	20		15	9	15	23	A	5	25	26	10	10
	13	13		8	8	16	16	17	18	19	19	11	11
	7	7		7	7	9	9	18	17	10	10	12	12
	3	3		3	3	4	4	5	A	5	5	6	6
Pradžia	P (1)												

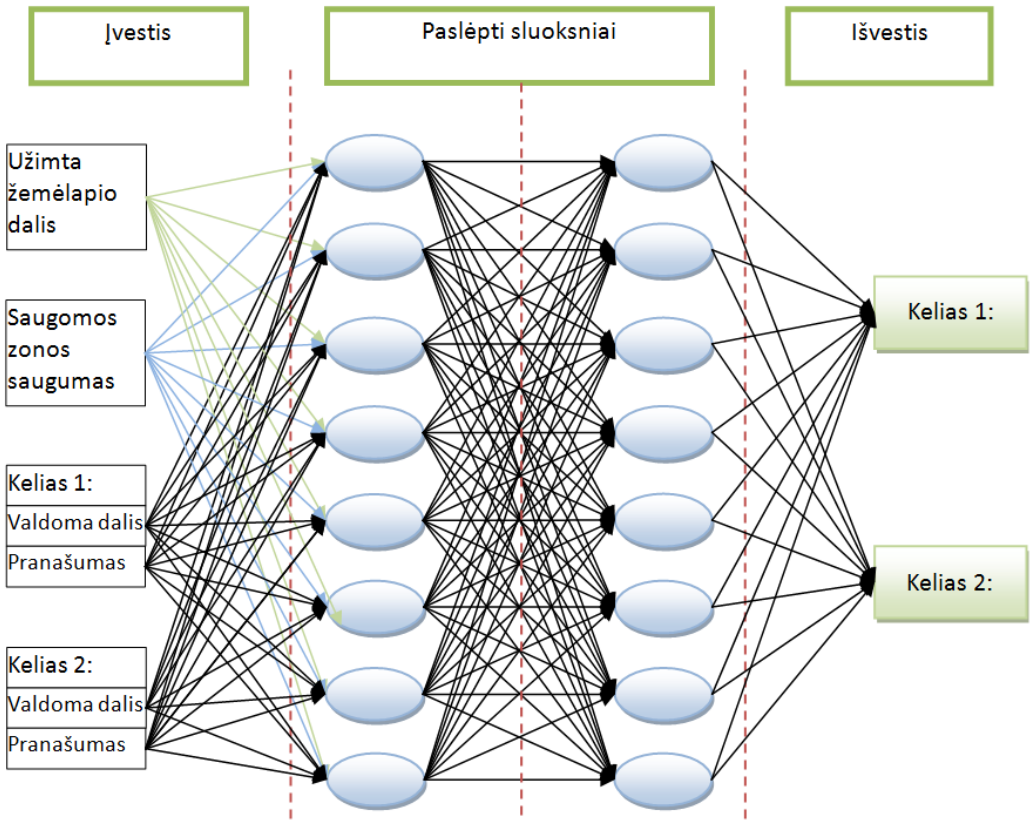
2.5.2. Neuroninis tinklas

Neuroninis tinklas skirtas modeliuoti žaidimo veikėjų veiksmus žaidimo eigoje. Veikėjų būriai atvaizduojami figūromis. Kiekvienai jų aproksimuojamos veikėjų būrio sąvybės:

- Būrio gyvybingumas priklauso nuo jį sudarančių karių skaičiaus ir būklės.
- Puolamosios jėgos parametrai priklauso nuo būrį sudarančių karių skaičiaus ir jiems priskirtos ginkluotės.

Veikėjai pasirenka veiksmus pagal kelių lentelėje išskirtiems keliams suteiktus koeficientus. Koeficientas nurodo veikėjui ar jis nori konkrečiu keliu judėti į priekį, ar sustoti ir gintis. Veikėjų būriai, būdami vienoje iš zonų grafo viršūnių, gali rinktis judėti bet kuriuo keliu einančiu per šią viršūnę. Pasirinkimas priklauso nuo keliams suteiktų koeficientų.

Veikėjų veiksmams parinkti naudojamas neuroninis tinklas. Jis įvertina valdomą žemėlapio dalį, priešiškų pajėgų atstumą nuo saugomo taško, bei padėtį kiekviename kelyje atskirai. Paslėptų neuroninio tinklo sluoksnių dydžiai priklauso nuo kelių skaičiaus ir juos sudarant nurodyto daugiklio. Išvesties sluoksnyje neuroninis tinklas kiekvienam keliui išveda veikėjų veiksmų parinkimui skirtą koeficientą.



Pav. 2.2 Neuroninio tinklo struktūra

Toks neuroninis tinklas turi būti sudaromas ir apmokomas kiekvienam žemėlapiui atskirai.

2.5.3. Genetinis algoritmas

Neuroninis tinklas apmokomas naudojant genetinį algoritmą.Sudaromos chromosomos, kurių ilgis yra lygus kelių skaičiui. Kiekvienas chromosomą sudarantis skaitmuo nurodo atitinkamo kelio koeficientą. Nurodžius šiuos koeficientus atliekamas veikėjų veiksmų modeliavimas. Tada tinkamumo funkcija, atsižvelgdama į valdomo žemėlapio plotą, kraštinių valdomo ploto zonų būklę ir situacijas visuose keliuose suteikia chromosomai tinkamumo koeficientą. Atlikus veikėjų veiksmų modeliavimą, sistemos parametrai atstatomi, ir viskas kartojama su kita chromosoma. Chromosomos kryžminimo ir mutacijų būdais keičiamos ir pritaikomos skaičiavimuose, iki pasiekiama chromosomų generavimo riba. Tada atrenkama geriausiai įvertinta chromosoma. Ji išsaugoma kartu su buvusiais žemėlapio parametrais kaip įvesties ir jai optimalios išvesties pora. Pagal šiuos duomenis vėliau bus apmokomas neuroninis tinklas.

2.5.4. Veikėjų veiksmų parinkimas naudojant pozicijų įvertinimą

Veikėjų būriai lokalius veiksmus pasirenka pagal pozicijų įvertinimo metodą. Konkretus būrys įvertina kiekvienam keliui, einančiam per to būrio užimamą zoną, suteiktą koeficientą. Jei koeficientas artimas vienetui, būrys linkęs judėti tuo keliu į priekį. Jei koeficientas artimas nuliui, būrys linkęs trauktis tuo keliu atgal. Papildomai įvertinami potencialios būriui tenkančios žalos ir potencialios būrio keliamos žalos priešininkams skirtumai tarp zonos kurioje jis yra ir zonos į kurią jį nukreipia kelio koeficientas. Pagal keliui suteiktą koeficientą ir gautus potencialų koeficientus, suskaičiuojami koeficientai visiems galimiems būrio veiksmams, susijusiems su tuo keliu. Šie skaičiavimai kartojami visiems keliams einantiems per būrio užimamą zoną, ir išrenkamas veiksmas su didžiausiu koeficientu.

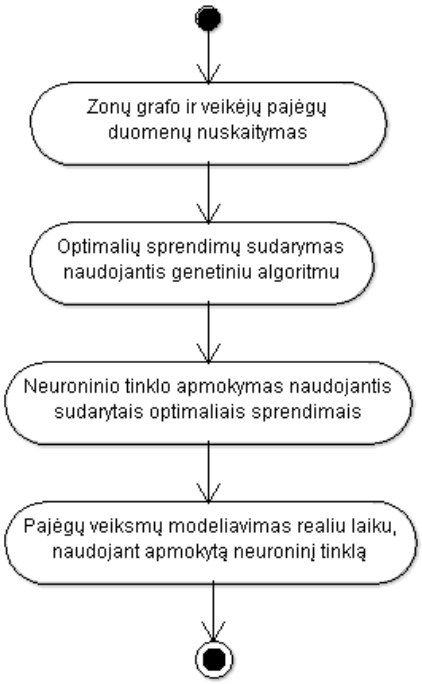
2.5.5. Sistemos loginė architektūra

Sistema susideda iš keturių pagrindinių komponentų:

- Navigacijos grafo posistemė enkapsuliuoja visas su grafu susijusias duomenų struktūras ir procesus, manipuliuojančius grafo duomenimis.
- Veikėjų pajėgų posistemė enkapsuliuoja būrių duomenis, procesus nustatančius lokalius būrių veiksmus ir apskaičiuojančius su tais veiksmais susijusius kriterijus.
- Genetinio algoritmo posistemė enkapsuliuoja optimalių sprendimų generavimą ir jų išsaugojimą.
- Neuroninio tinklo posistemė enkapsuliuoja neuroninio tinklo duomenų išvedimo ir jo apmokymo procesus.

2.5.6. Sistemos elgsenos modelis

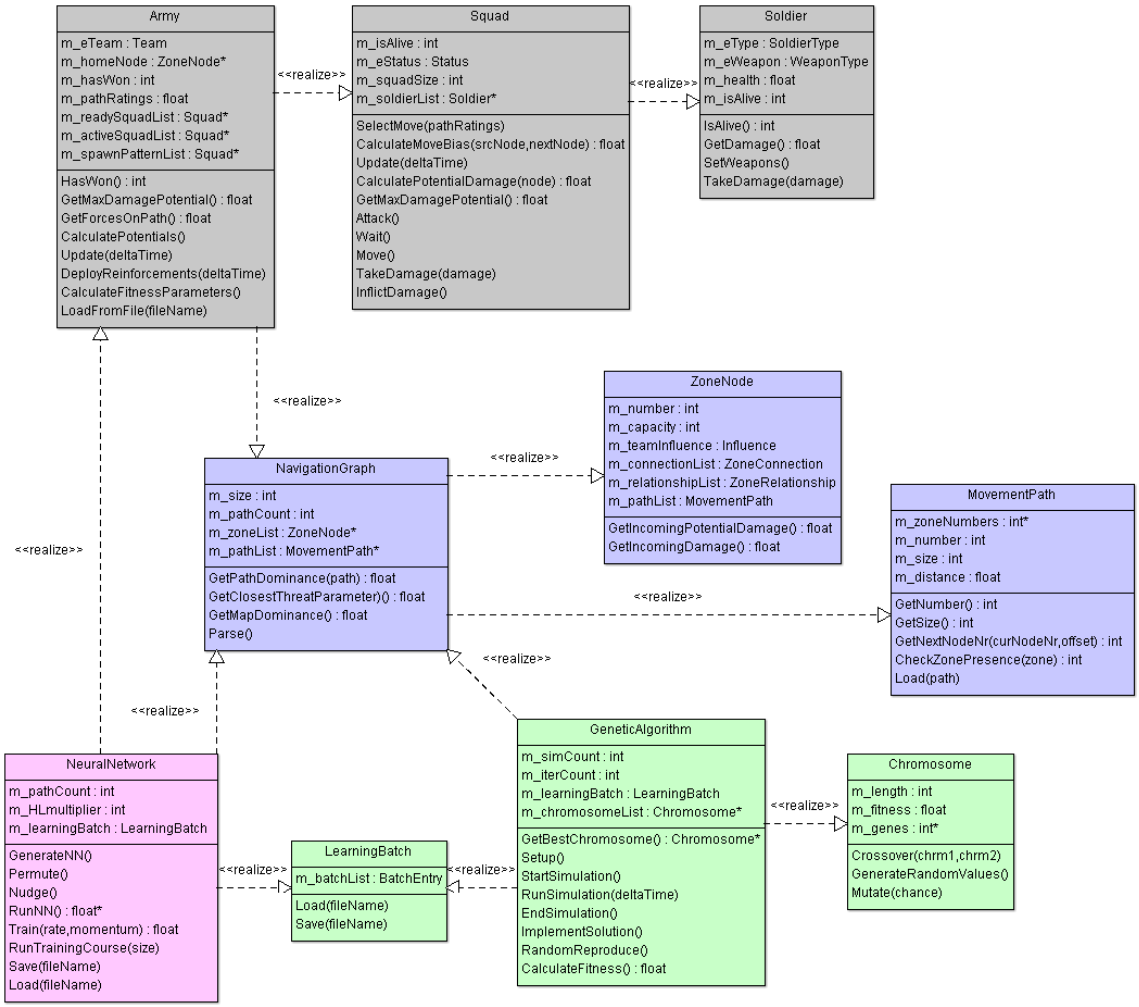
Pagrindiniai sistemos procesai ir jų eiliškumas pavaizduoti sistemos elgsenos modeliu.



Pav. 2.3 Sistemos būsenos diagrama

2.5.7. Detalus projektas

Sistemos detalaus projekto klasių diagrama vaizduojama 2.4 paveiksle.



Pav. 2.4 Sistemos klasių diagrama

Pagrindiniai sistemos klasių metodai aprašyti 2.2 lentelėje.

Lentelė 2.2 Projekto klasių metodai

Metodas	Aprašymas
Klasė NavigationGraph (skirta realizuoti zonų grafo funkcionalumą)	
GetPathDominance(int path)	Metodas apskaičiuojantis pajėgų sąntykį nurodytame kelyje.
GetClosestThreatParameter()	Metodas randantis kokiu atstumu nuo saugomo taško yra priešininkų pajėgos.
GetMapDominance()	Metodas apskaičiuojantis kokią dalį grafo valdo pajėgos.
Parse()	Metodas, iš failo nuskaitantis grafo duomenis.
Klasė ZoneNode (skirta realizuoti zonos funkcionalumą)	
GetIncomingPotentialDamage()	Metodas skirtas apskaičiuoti kokią potencialią žalą zonoje gali padaryti priešininkų pajėgos.
GetIncomingDamage()	Metodas skirtas apskaičiuoti priešininkų padarytą žalą zonoje.
Klasė Army (skirta vienoje pusėje esančių veikėjų pajėgų valdymui)	

CalculatePotentials()	Metodas apskaičiuojantis pajėgų potencialią žalą visose grafo zonose.
Update(float time)	Metodas, kintant laikui vykdantis pajėgų judėjimą.
DeployReinforcements(float time)	Metodas valdantis papildomų pajėgų sukūrimą aplinkoje.
LoadFromFile(String filename)	Metodas, iš failo nuskaitantis pajėgų sudėtį.
Klasė Squad (skirta būrio valdymui)	
SelectMove()	Metodas, pagal pasiekiamų kelių koeficientus parenkantis sekantį būrio veiksmą.
Update(float time)	Metodas, kintant laikui vykdantis būrio pasirinktus veiksmus.
Attack()	Metodas, nurodantis būriui kovoti su aplinkiniais priešininkais 1 sekundę.
Wait()	Metodas, nurodantis būriui palaukti vietoje 1 sekundę.
Move()	Metodas, nurodantis būriui pereiti į kitą zoną.
TakeDamage(float damage)	Metodas, atsitiktinai paskirstantis priešininkų padarytą žalą būrio nariams.
InflictDamage()	Metodas, apskaičiuojantis žalą, kurią būrys kovodamas sukelia aplinkiniams priešininkų būriams.
Klasė Soldier (skirta veikėjo valdymui)	
GetDamage()	Metodas, kuris nustato veikėjo sukeliamą žalą konkrečioje situacijoje.
SetWeapons()	Metodas, pagal situaciją parenkantis veikėjui optimalų ginklą.
Klasė GeneticAlgorithm (skirta realizuoti genetinio algoritmo veikimą)	
Setup()	Metodas, paruošiantis pradinę chromosomų populiaciją.
StartSimulation()	Metodas, pradedantis pajėgų valdymo simuliaciją, pagal chromosomoje esančias reikšmes.
EndSimulation()	Metodas, įvertinantis atliktos simuliacijos tinkamumą.
ImplementSolution()	Metodas, atrenkantis ir išsaugantis geriausią chromosomą.
RandomReproduce()	Metodas, vykdantis genetinio algoritmo chromosomų atrinkimą ir kryžminimą.
CalculateFitness()	Metodas apskaičiuojantis chromosomų tinkamumo koeficientą.
Klasė Chromosome (skirta chromosomų generavimui ir kryžminimui)	
Crossover()	Metodas, gaunantis naują chromosomą sukryžminant dvi turimas chromosomas.
GenerateRandomValues()	Metodas, generuojantis atsitiktines chromosomos reikšmes.
Mutate(int chance)	Metodas, atsitiktinai pakeičiantis vieną chromosomos reikšmę.
Klasė NeuralNetwork (skirta realizuoti dirbtinio neuroninio tinklo apmokymą ir veikimą)	
GenerateNN()	Metodas, sugeneruojantis neuroninį tinklą su atsitiktiniais briaunų svoriais.

Permute()	Metodas, atsitiktine tvarka sumaišantis apmokymui skirtus duomenis.
Nudge()	Metodas, pridedantis santykinai nedideles sumas prie neuroninio tinklo briaunų svorių.
Run()	Metodas, įvykdantis neuroninio tinklo skaičiavimus ir gaunantis jo išvestį.
Train(float rate)	Metodas, įvykdantis vieną atbulinės klaidos propagacijos metodo iteraciją ir nustatantis neuroninio tinklo klaidos dydį.
RunTrainingCourse()	Metodas, vykdantis atbulinės klaidos propagacijos algoritmą tol, kol neuroninio tinklo klaidos dydis netampa mažesnis už iš anksto nustatytą ribą.

3. SISTEMOS REALIZACIJA IR TESTAVIMAS

3.1. Sistemos veikimo aprašymas

Sistema veikia trimis etapais:

- Sprendimų generavimo etapo metu, genetinis algoritmas ieško optimalių sprendimų susidariusioms situacijoms. Atlikęs tam tikrą kiekį paieškos iteracijų, algoritmas išsaugo visus rastus sprendimus faile.
- Neuroninio tinklo apsimokymo etapo metu sukuriamas ir apmokomas naujas neuroninis tinklas. Tam naudojami atbulinės klaidos propagacijos metodas ir sprendimų generavimo etape sugeneruoti sprendimai.
- Sistemos veikimo metu, veikėjų veiksmai modeliuojami realiu laiku, naudojant apmokytą neuroninį tinklą.

Sistema valdoma nurodant veikimo parametrus setup.xml faile. Šis failas suskirstytas į dvi dalis: nustatymai skirti apmokymams ir nustatymai skirti veikėjų valdymui realiu laiku. Visi nustatymai aprašyti 3.1 lentelėje.

Lentelė 3.1 setup.xml faile saugomi programos nustatymai

Pavadinimas	Reikšmė
Skiltis Learning (skirta saugoti nustatymus sprendimų generavimo ir neuroninio tinklo apmokymo etapams)	
RunGA	Pasirinkimas, leidžiantis nustatyti genetinio algoritmo sprendimų generavimo etapo vykdymą. Galimi parametrai: (true, false).
Iterations	Nurodomas sprendimų generavimo etape generuojamų sprendimų skaičius.
BlueBatchSaveFile	Nurodomas failas mėlynos spalvos pajėgų sprendimams saugoti.
RedBatchSaveFile	Nurodomas failas raudonos spalvos pajėgų sprendimams saugoti.
TrainNN	Pasirinkimas, leidžiantis nustatyti neuroninio tinklo apmokymo etapo vykdymą. Galimi parametrai: (true, false).
TrainTeam	Pasirenkamos pajėgos kurioms bus apmokomas neuroninis tinklas. Galimi parametrai: (blue, red).
BatchSize	Nurodomas sprendimų, naudojamų apmokyti neuroninį tinklą kiekis.
BlueBatchLoadFile	Nurodomas failas, kuriame saugomi sprendimų generavimo etape mėlynos spalvos pajėgoms sugeneruoti sprendimai.
RedBatchLoadFile	Nurodomas failas, kuriame saugomi sprendimų generavimo etape raudonos spalvos pajėgoms sugeneruoti sprendimai.
BlueNNSaveFile	Nurodomas failas mėlynos spalvos pajėgų neuroniniam tinklui saugoti.
RedNNSaveFile	Nurodomas failas raudonos spalvos pajėgų neuroniniam tinklui saugoti.
Skiltis Running (skirta saugoti nustatymus veikėjų veiksmų modeliavimo realiu laiku etapui)	
RunMatches	Nurodomas kiekis kovų tarp pajėgų, vykdomų veikėjų veiksmų modeliavimo realiu laiku etape.
BlueNN	Nurodomas failas, kuriame saugomas mėlynos spalvos pajėgoms skirtas neuroninis tinklas.

RedNN	Nurodomas failas, kuriame saugomas raudonos spalvos pajėgoms skirtas neuroninis tinklas.

Sprendimų generavimo genetiniu algoritmu etapas atliekamas nustatymuose pasirinkus sprendimų generavimo etapo vykdymą ir paleidus sistemą. Ji sugeneruos nustatymuose nurodytą kiekį sprendimų abiejų spalvų pajėgoms ir išsaugos juos sprendimų saugojimui nurodytuose failuose. Generavimo metu konsolėje sistema spausdins iteracijos numerį, įvesties parametrus ir visas tikrinamas chromosomas bei joms suteiktus tinkamumo koeficientus. Pasibaigus sprendimų generavimui sistemą galimą išjungti.

Neuroninio tinklo apmokymo etapas atliekamas nustatymuose pasirenkant neuroninio tinklo apmokymo galimybę, bei pajėgas, kurioms jis bus apmokomas. Taip pat nurodoma kiek sprendimų bus naudojama apmokymo procese. Sprendimai nuskaitomi iš juos saugančio failo, atitinkančio nurodytas pajėgas. Neuroninio tinklo apmokymo procesas gali ilgai užsitęsti arba išvis nepasiekti tikslo. Jo metu sistema kas 5000 iteracijų išspausdina neuroninį tinklą įvertinantį klaidos koeficientą. Neuroninis tinklas yra apmokomas ir išsaugomas nustatymuose nurodytame faile, jei šis koeficientas tampa mažesnis už 0,03. Jam apsimokius sistema pereina į veikėjų veiksmų modeliavimo etapą. Priešininkų pajėgas valdantis neuroninis tinklas nuskaitomas iš nustatymuose nurodyto failo.

Jei nustatymuose nepasirinkti nei sprendimų generavimo etapas, nei neuroninio tinklo apmokymų etapas, sistema modeliuoja veikėjų veiksmus. Neuroniniai tinklai nuskaitomi iš failų nurodytų nustatymų skiltyje „Running“. Veikėjų veiksmų modeliavimas vyks nurodytą kiekį kovų. Kiekvienai kovai pasibaigus sistema konsolėje atspausdina abiejų pajėgų laimėjimų skaičių.

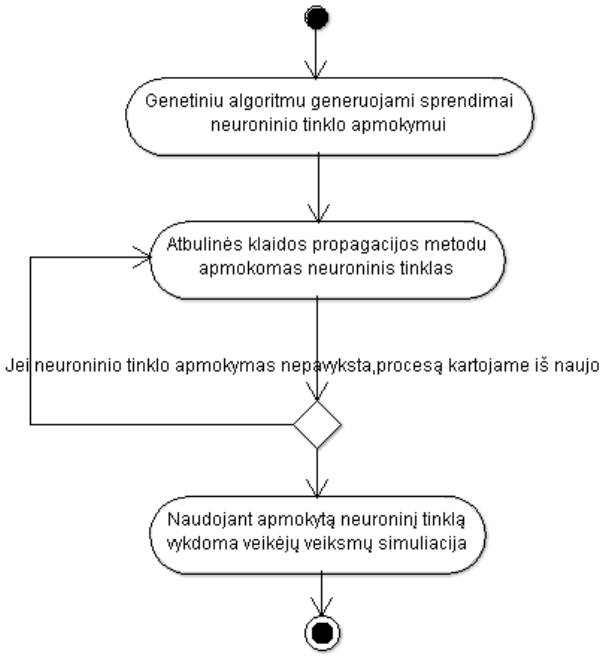
Aplinką aprašantis grafas ir veikėjų pajėgos nurodomos atskiruose failuose. Veikėjų pajėgas aprašantys failai taip pat nurodo naujų būrių sukūrimo aplinkoje taisykles.

3.2. Testavimo modelis

Testuojant sistemą patikrinami visi jos darbo aspektai. Testavimas laikomas sėkmingu jei:

- Genetinis algoritmas sėkmingai sugeneruoja nurodytą kiekį sprendimų mėlynąsias ir raudonąsias pajėgas valdantiems neuroniniams tinklams apmokinti ir išsaugo juos, kartu su atitinkančiomis įvesties būsenomis, atskiruose failuose.
- Naudojantis atbulinės klaidos propagacijos metodu, pagal nurodytą kiekį genetinio algoritmo sugeneruotų sprendimų ir jiems priskirtų įvesčių apmokomas neuroninis tinklas.
- Nurodytą kiekį kartų įvykdoma veikėjų veiksmų simuliacija, veikėjų pasiskirstymui virtualioje aplinkoje valdyti naudojant du neuroninius tinklus.

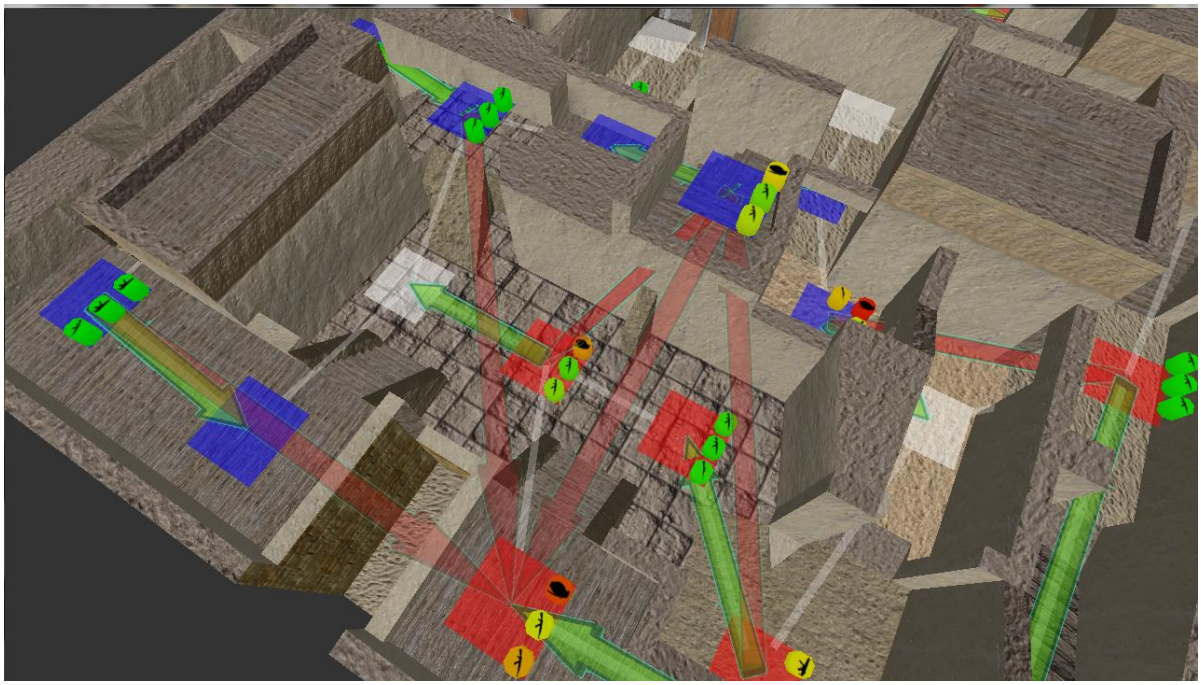
Pateikiamas sistemos testavimo modelis:



Pav. 3.1 Sistemos testavimo modelis

3.3. Testavimo duomenys ir rezultatai

Pirmiausia genetiniu algoritmu sudarinėjami sprendimai neuroninio tinklo apmokymui. Nustatymų faile setup.xml nurodomi sudarinėjamų sprendimų kiekis ir du failai, abiejų pusių pajėgų sprendimų išsaugojimui. Genetinis algoritmas generuoja sprendimus iškart dviems neuroniniams tinklams. Kiekvienos iteracijos metu, nuskaitomi situaciją nusakantys duomenys ir sugeneruojamos pradinės chromosomos. Algoritmas pagal kiekvienoje chromosomoje saugomą sprendimą nustato kelių koeficientus. Pagal šiuos koeficientus procedūrinis pozicijų parinkimo metodas parenka veiksmus veikėjų būriams (žr. Pav 3.2). Veiksmų modeliavimas trunka 10 sekundžių. Po šio termino judėjimas stabdomas ir tinkamumo funkcija įvertinamas tinkamumo koeficientas. Veikėjų būsenos atstatomos į pradinę padėtį ir vertinama sekanti chromosoma. Įvertinus visas pradines chromosomas, pagal genetinio algoritmo specifika iš jų sudarinėjamos ir vertinamos naujos. Kiekviena įvertinta chromosoma išspausdinama konsolėje kartu su jai priskirta tinkamumo funkcija. Įvertinus visas chromosomas, aukščiausią tinkamumo koeficientą turinti chromosoma išsaugoma kartu su pradine situacija. Veikėjų veiksmai 10 sekundžių modeliuojami pagal šios chromosomos sprendimą. Tada pradedama kita iteracija. Iteracijos vykdomos iki pasiekiamas iš anksto nurodytas iteracijų limitas. Tuo atveju visi atrinkti sprendimai kartu su juos atitinkančiomis įvestimis išsaugomi faile ir programą galima išjungti.



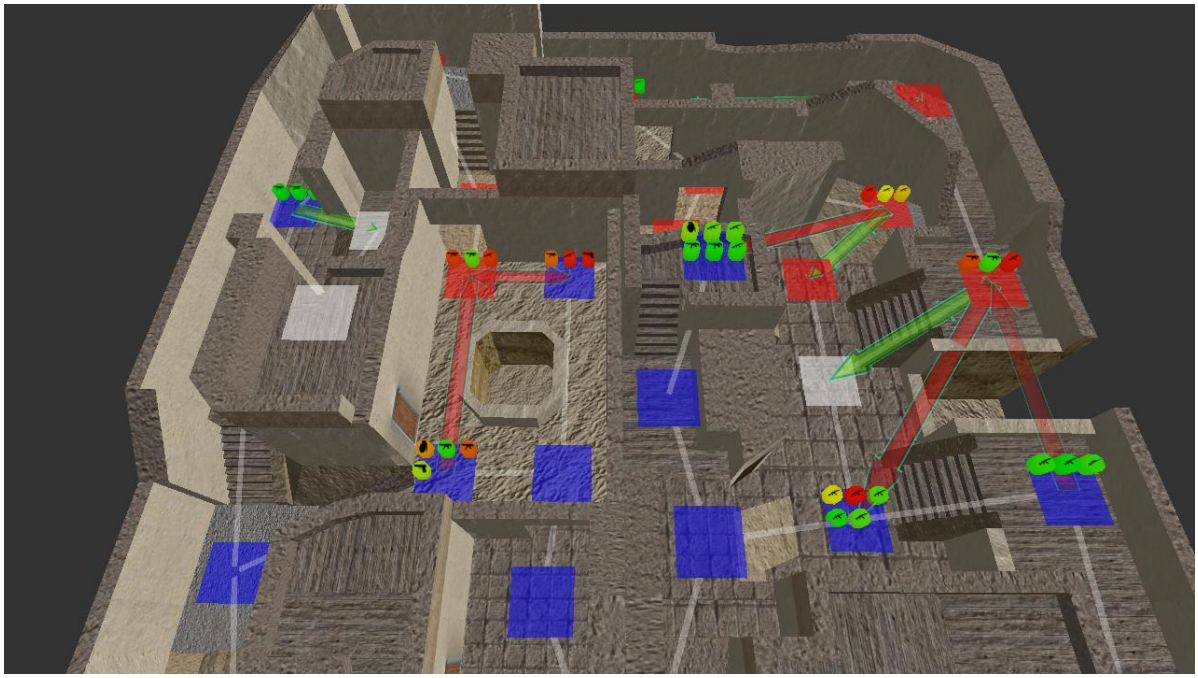
Pav. 3.2 Genetinio algoritmo metu modeliuojamas veiksmas

Sugeneravus sprendimų failus galima apmokyti neuroninį tinklą. Nustatymų faile setup.xml nurodomas sprendimus saugantis failas, pagal kurį neuroninis tinklas bus apmokomas, failas į kurį neuroninis tinklas bus išsaugomas ir sprendimų, naudojimų apmokyme skaičius. Taip pat pasirenkama kurios spalvos pajėgų neuroninis tinklas bus apmokomas. Paleidus programą pradedamas vykdyti atbulinės klaidos propagacijos algoritmas. Kas 5000 iteracijų, algoritmas išspausdina neuroniniam tinklui suskaičiuotą klaidos kriterijų. Kai šis kriterijus nusileidžia žemiau 0.03 ribos, tinklas laikomas apmokytu. Konsolėje išspausdinami visi neuroninio tinklo apmokyme naudoti sprendimai, ir tinklo suskaičiuoti jų atitikmenys. Taip pat išspausdinama apmokymų trukmė sekundėmis. Baigus neuroninio tinklo apmokymą, programa iškart paleidžia to tinklo veikėjų veiksmų modeliavimo simuliacija.

```
Iteration: 25000
14.2642
Iteration: 30000
14.152
Iteration: 35000
13.9167
Iteration: 40000
13.6919
Iteration: 45000
13.5469
Iteration: 50000
13.3984
```

Pav. 3.3 Neuroninio tinklo apmokymas atbulinės klaidos propagacijos metodu

Paruošus neuroninius tinklus abiems veikėjų pajėgoms valdyti, galima modeliuoti veikėjų tarpusavio kovas. Nustatymų faile setup.xml nurodomi neuroninių tinklų failai ir vykdomų kovų skaičius. Paleidus programą bus vykdoma tiek kovų, kiek nurodys šis skaičius. Kaskart pasibaigus kovai, konsolėje bus išspausdinama bendra kovų statistika.



Pav. 3.4 Veikėjų veiksmų modeliavimas neuroniniais tinklais

4. EKSPERIMENTINIS SISTEMOS TYRIMAS

4.1. Eksperimento planas

Eksperimentas vyksta virtualioje aplinkoje, kurios sąvybes aproksimuoja zonų grafas. Šioje aplinkoje kovoja dvi veikėjų pusės: melynųjų pajėgos ir raudonųjų pajėgos. Abi jos turi savo pradinę zoną skirtingose aplinkos pusėse. Jų tikslas yra pasiekti priešininkų pradinę zoną, tuo pačiu metu apginant sąvają. Veikėjų būrių judėjimas grafu vyksta tarpusavyje sujungtų zonų eilėmis, vadinamomis keliais. Šiame eksperimente kelių skaičius yra 13. Kiekvienam keliui suteikiamas koeficientas, pagal kurį veikėjų būrys renkasi sekantį savo ėjimą. Abiejų veikėjų pajėgų sudėtys yra identiškos. Kas 20 sekundžių abi pajėgos papildomos veikėjų būriu. Naujo būrio sudėtis parenkama pagal tam tikrą seką, kuri abiems pajėgoms taipogi yra identiška.

Genetinis algoritmas ruošia optimalių sprendimų aibę, skirtą neuroninio tinklo apmokymui. Šis algoritmas sudaro dešimt pirminių chromosomų atsitiktinai, jas įvertina ir kryžminimo būdu iš turimos populiacijos generuoja naują populiaciją. Šis procesas kartojasi, kol bendrai paėmus įvertinama penkiasdešimt chromosomų. Tada išrenkama geriausiai įvertinta chromosoma. Norint įvertinti kiekvieną chromosomą, reikia tam tikrą laiko tarpą vykdyti veikėjų veiksmų simuliaciją. Kadangi ši simuliacija užima tam tikrą laiko tarpą, didelio chromosomų kiekio vertinimas gali labai užtrukti. Dėl šios priežasties vertinimas apsiriboja penkiasdešimčia chromosomų.

Jeigu veiksmų simuliacijai suteikiama per mažai laiko, veikėjų būriai, pradedami judėti nuo vienos zonos prie kitos, per duotą laiką nepabaigs ėjimo. Tokiu atveju šio ėjimo potencialas liks neįvertintas. Jei simuliacijai suteikiama per daug laiko, veikėjų būriai praras galimybę greitai reaguoti į kintančią situaciją. Pasirinkta simuliacijos trukmė yra 10 sekundžių, kadangi tai mažiausias laiko tarpas per kurį veikėjų būrys gali pereiti ilgiausią, tarp dviejų zonų esantį atstumą grafe.

Vykdant simuliaciją, chromosomos saugomos reikšmės nustatomos kaip kelių koeficientai. Veikėjų būriai, pagal tuos koeficientus pasirenka veiksmus. Prieš atliekant simuliaciją nuskaitomi aplinkoje susidariusią situaciją nusakantys parametrai. Po simuliacijos nustatomas šių parametrų pokytis. Pagal šį pokytį tinkamumo funkcija nustato tikrinamos chromosomos tinkamumą. Aplinkoje susidariusią situaciją nusakyti išgaunami šie parametrai:

1) Pajėgų valdoma aplinkos dalis, kuri nusako, kokią grafo dalį yra užsiėmusios veikėjų pajėgos.
2) Visų veikėjų esančių pajėgose bendras gyvybingumas.
3) Visų veikėjų esančių pajėgose bendras galimas potencialas padaryti žalą priešininkams.
4) Visų veikėjų esančių pajėgose bendras galimas potencialas susižeisti.
5) Priešininkams likęs atstumas iki saugomo taško, parodantis pralaimėjimo grėsmę.

Prieš pradedant genetiniu algoritmu generuoti chromosomas, apskaičiuojami neuroninio tinklo įvesties parametrai. Jie susiejami su geriausia išrinkta chromosoma. Šie parametrai nurodo kaip aplinką suvokia neuroninis tinklas. Juos apskaičiavus, patikrinama, ar optimalių sprendimų aibėje jau nėra sudarytų sprendimų panašiai situacijai spresti. Jeigu jų yra, sugeneruotas sprendimas neįrašomas į optimalių sprendimų aibę. Taip apsisaugojama nuo sprendimų daugiareikšmiškumo, kada vienodiems duomenims egzistuoja du skirtingi sprendimai. Neuroniniam tinklui situacija aplinkoje nurodoma šiais parametrais:

1) Pajėgų valdoma aplinkos dalis.
2) Priešininkams likęs atstumas iki saugomo taško.
3) Pajėgų valdoma dalis kiekviename kelyje.
4) Pajėgų veikėjų sąntykis su priešininkų veikėjais kiekviename kelyje.

Iš paruoštos optimalių sprendimų aibės apmokomas neuroninis tinklas. Aciklinio neuroninio tinklo struktūra pritaikoma žemėlapio kelių skaičiui. Įvesties sluoksnis turi po įvedimo tašką,

kiekvienam tinklo įvesties parametrui. Išvesties sluoksnis turi tiek pat neuronų, kiek yra kelių. Tarp šių sluoksnių yra du paslėpti sluoksniai. Juos sudarančių paslėptų neuronų skaičius priklauso nuo žemėlapyje esančių kelių ir pasirinkto daugiklio. Sudauginus kelių skaičių ir nurodytą daugiklį gaunamas paslėptų sluoksnių neuronų skaičius. Kiekvieno sluoksnio kiekvienas mazgas turi jungiančią briauną su visais mazgais sekančiame sluoksnyje.

Eksperimente palyginama kaip neuroninių tinklų apmokymų rezultatai priklauso nuo paslėptų sluoksnių dydžio. Apmokymuose naudojami skirtingi daugikliai ir apmokymui naudojamų duomenų kiekiai. Kiekvienos naudojamos konfigūracijos neuroninis tinklas apmokomas tris kartus. Suskaičiuojamas apmokymų trukmės vidurkis kiekvienai konfigūracijai.

Neuroninio tinklo apmokymas vyksta atbulinės klaidos propagacijos metodu. Jis tesiasi tol, kol tinklo išvesties sluoksnio neuronų paklaidų kvadratų suma netampa mažesnė už 0.03. Šis dydis yra pakankamai mažas, kad tinklas veiktų korektiškai. Jei tinklas neapsimoko per 12000000 iteracijų, apmokymas laikomas nepavykusiu.

Eksperimente lyginami tarpusavyje besivaržantys, priešiškas pajėgas valdantys neuroniniai tinklai. Bandoma nustatyti kokį poveikį neuroninio tinklo priimtų sprendimų kokybei turi apmokymams naudotų sprendimų imties dydis. Sudaromos neuroninių tinklų poros, kurios varžosi tarpusavyje 100 kartų. Gautas rezultatas parodo apytikrį vieno neuroninio tinklo pranašumą prieš kitą.

Eksperimente kovojančioms pajėgoms aplinkos suteikiamos sąlygos labai skiriasi. Dėl šių skirtumų priešiškas pajėgas valdantys neuroniniai tinklai negali būti tiesiogiai lyginami tarpusavyje. Todėl vertinamas neuroninio tinklo rezultatų pokytis keičiant vieno iš šių tinklų apmokymų imties dydį.

4.2. Eksperimento rezultatas

Eksperimentas buvo atliktas kompiuteryje su šiais parametrais:

- Procesorius: Intel Core 2 Quad CPU Q9400 @ 2,66 GHz
- Operatyvioji atmintis: 4 GB
- Operacinė sistema: Windows 7 Ultimate

Eksperimento metu vykusių neuroninių tinklų apmokymų rezultatai pateikiami 4.1 lentelėje.

Lentelė 4.1 Neuroninių tinklų apmokymų rezultatai

Apmokymuose naudojamų pavyzdžių kiekis	Paslėptųjų sluoksnių dydžio daugiklis	Tinklas 1		Tinklas 2		Tinklas 3		Vidutinė apmokymų trukmė
		Apmokymo trukmė	Iteracijų skaičius	Apmokymo trukmė	Iteracijų skaičius	Apmokymo trukmė	Iteracijų skaičius	
4	1	1min 8s	1396968	1min 12s	1482070	0min 55s	1112901	1min 5s
6	1	2min 35s	2140481	3min 35s	2968218	1min 58s	1637389	2min 43s
8	1	9min 19s	5835271	6min 21s	3966860	6min 13s	3435813	7min 18s
10	1	24min 19s	10865654	24min 35s	10986224	24min 47s	11080358	24min 34s
12	1	Apmokymas nepavyko		Apmokymas nepavyko		Apmokymas nepavyko		-
14	1	Apmokymas nepavyko		Apmokymas nepavyko		Apmokymas nepavyko		-
4	2	1min 55s	923850	1min 57s	927474	3min 54s	1862506	2min 35s
6	2	4min 27s	1652663	4min 31s	1673079	6min 12s	2300957	5min 3s
8	2	7min 42s	1836525	8min 34s	2059808	16min 52s	3899311	11min 3s
10	2	12min 57s	2520276	17min 44s	3350871	20min 6s	3897073	16min 56s
12	2	24min 7s	3925638	26min 9s	4228622	43min 6s	6911884	31min 7s
14	2	Apmokymas nepavyko		Apmokymas nepavyko		Apmokymas nepavyko		-
4	3	1min 29s	474484	1min 42s	541597	2min 32s	717153	1min 54s
6	3	5min 35s	1186583	5min 53s	1250340	7min 37s	1610848	6min 22s
8	3	33min 3s	5081130	15min 36s	2398355	37min 35s	5920476	28min 45s
10	3	20min 46s	2646273	29min 55s	3811575	30min 15s	3856811	26min 59s
12	3	36min 27s	3872087	44min 52s	4766379	40min 26s	4296448	40min 35s
14	3	92min 6s	7203771	99min 5s	7750576	109min 2s	8529299	100min 4s

4	4	4min 45s	988550	1min 59s	409997	4min 8s	854693	3min 37s
6	4	10min 3s	1401952	8min 16s	1144137	20min 0s	2718657	12min 46s
8	4	16min 39s	1748735	15min 57s	1676347	18min 44s	1673041	17min 7s
10	4	31min 44s	2269763	33min 53s	2840926	38min 25s	2747346	34min 41s
12	4	40min 19s	2825354	61min 21s	4299560	48min 23s	3391138	50min 1s
14	4	95min 2s	4409248	159min 54s	7828542	143min 58s	6679888	132min 58s

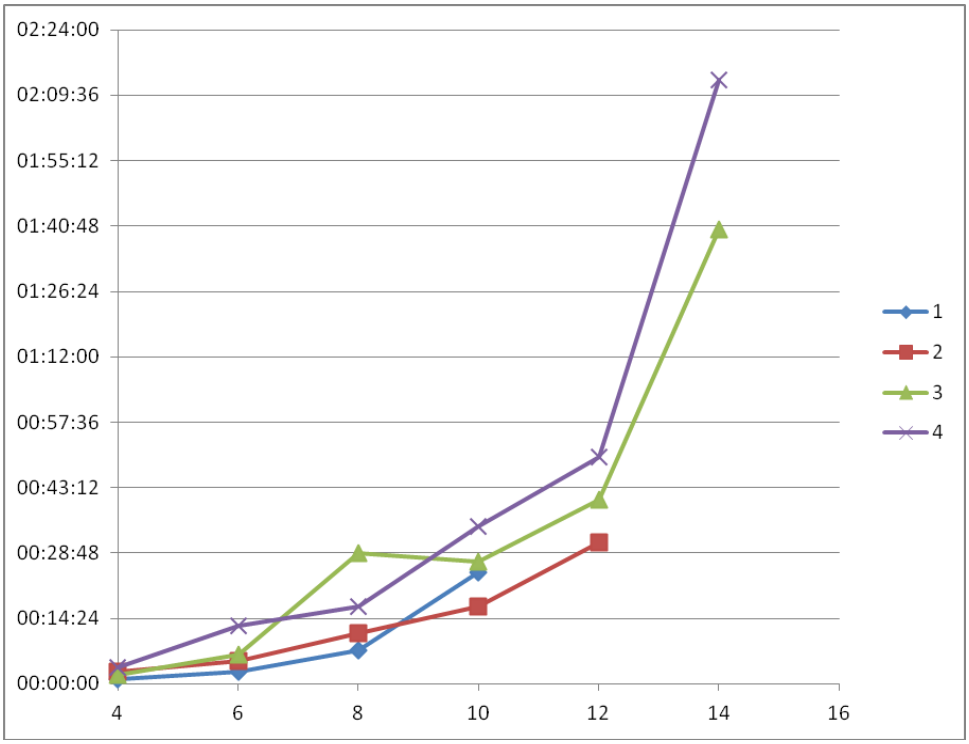
Eksperimentas parodo mėlynos spalvos pajėgas valdančio neuroninio tinklo kovoje pasiekto rezultato kitimą, kintant jo apmokymams naudotų sprendimų imties dydžiui. Rezultatų lentelėje 4.2 parodyta kiek iš šimto galimų laimėjimų, laimėjo mėlynos spalvos pajėgos.

Lentelė 4.2 Tarpusavyje kovojančių neuroninių tinklų rezultatai

Raudonųjų neuroninio tinklo apmokymų dydis	Mėlynųjų neuroninio tinklo apmokymų dydis						
	3	4	5	6	7	8	9
4	49	33	16	49	64	69	53
5	12	24	3	0	22	50	52
6	50	26	28	15	81	45	64
7	15	10	44	43	77	32	71
8	27	50	67	88	86	50	85
9	17	55	78	89	89	36	87

4.3. Eksperimento rezultatų įvertinimas

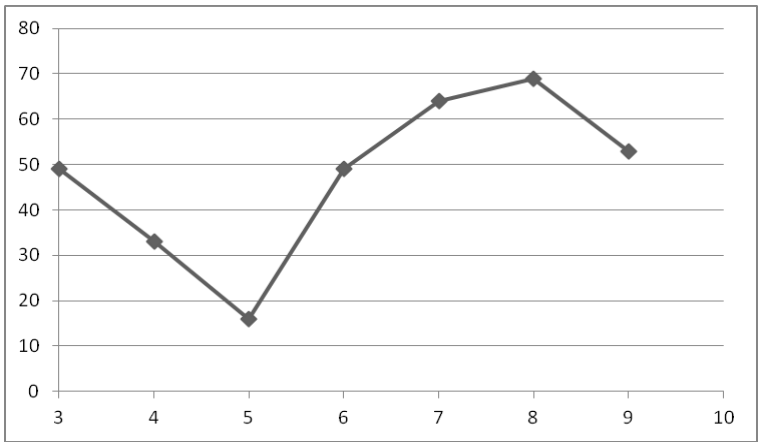
Eksperimente tiriama, kaip skirtingi apmokymui naudojamų pavyzdžių kiekiai ir paslėptųjų sluoksnių dydis veikia neuroninio tinklo apmokymo trukmę. Rezultatai, parodantys neuroninio tinklo apmokymo trukmės priklausomybę nuo pasirinkto paslėptų sluoksnių daugiklio ir apmokymui naudojamų pavyzdžių skaičiaus, parodomi 4.1 paveiksle.



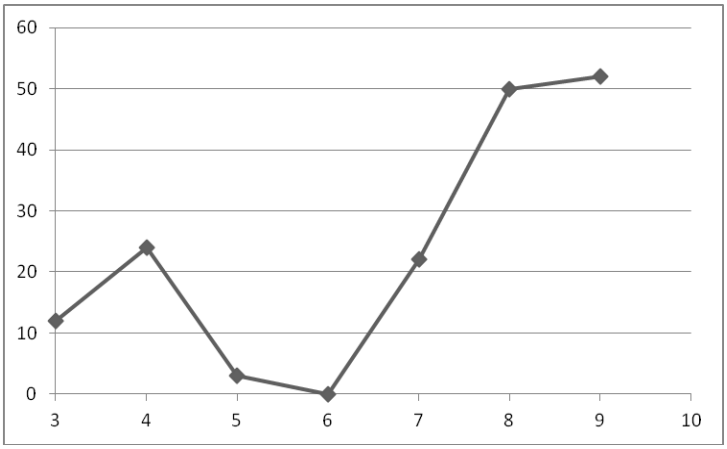
Pav. 4.1 Neuroninių tinklų apmokymo trukmės kitimas

Rezultatuose matosi kiek laiko trunka neuroninį tinklą apmokyti skirtingais kiekiais sprendimų pavyzdžių, priklausomai nuo neuroninio tinklo paslėptųjų sluoksnių dydžio. Šiuose sluoksniuose esančių neuronų skaičius nustatomas padauginus kelių skaičių iš pasirinkto daugiklio. Iš gautų rezultatų matosi, kad mažesni neuroniniai tinklai yra greičiau apmokomi, tačiau jų išmokstamų pavyzdžių kiekis yra labai ribotas. Didinant neuronino tinklo paslėptuosius sluoksnius, didėja sprendimų erdvė, kurią neuroninis tinklas gali atvaizduoti, tačiau tuo pačiu didėja jo apmokymo trukmė.

Pajėgų tarpusavio kovų duomenys atvaizduojami grafikais. Jie parodo, kiek iš šimto kovų prieš konkretų, raudonuosius valdantį neuroninį tinklą, laimėjo mėlynuosius valdantis neuroninis tinklas, su skirtingomis apmokymams naudotų pavyzdžių imtimis. Šiuose grafikuose galima įžvelgti tendenciją rezultatams gerėti, didinant apmokymams naudotų pavyzdžių imties skaičių. Tačiau ši priklausomybė nėra vienareikšmė.

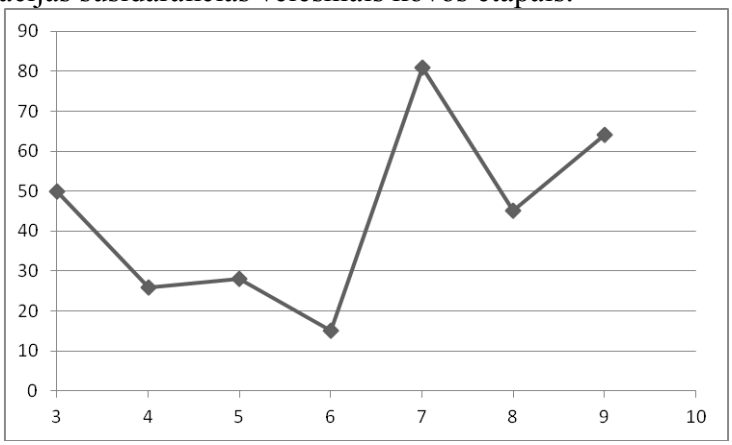


Pav. 4.2 Rezultatų kitimas, lyginant su keturiems sprendimams apmokytu tinklu

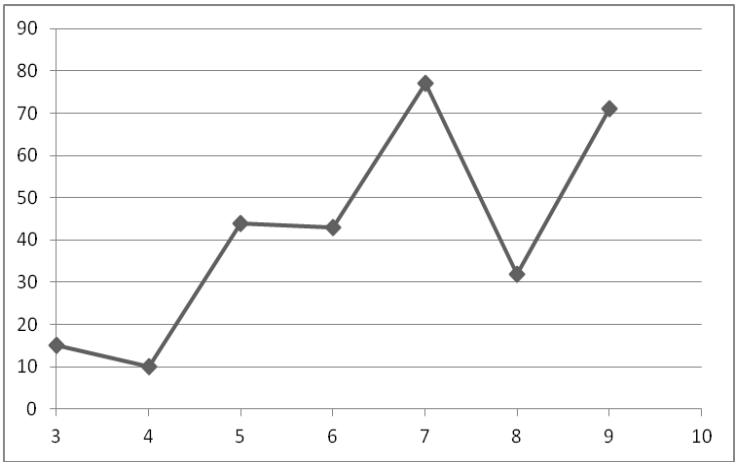


Pav. 4.3 Rezultatų kitimas, lyginant su penkiems sprendimams apmokytu tinklu

Apmokymams naudotų pavyzdžių skaičiui didėjant, didėja neuroninio tinklo galimybė tiksliau reaguoti į susidariusią situaciją. Kol tinklo apmokymo lygis žemas, situacija vertinama labai grubiai. Tokio tinklo sudarytas sprendimas gali atrodyti atsitiktinis. Tai ypač aiškiai matosi lyginant mažai apmokytą mėlynąsias pajėgas valdantį neuroninį tinklą su skirtingais lygiais apmokytu, raudonąsias pajėgas valdančiu neuroniniu tinklu. Šį reiškinį galima stebėti 4.2, 4.3, 4.4 pav. grafikų pradžiose. Apmokyti tik su pavyzdžiais priimtais kovos pradžioje, neuroniniai tinklai neturi žinių bazės adekvačiai reaguoti į situacijas susidarančias velėsniais kovos etapais.



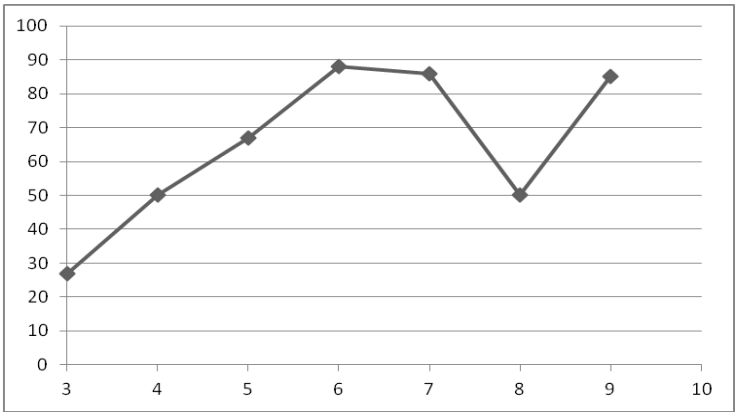
Pav. 4.4 Rezultatų kitimas, lyginant su šešiems sprendimams apmokytu tinklu



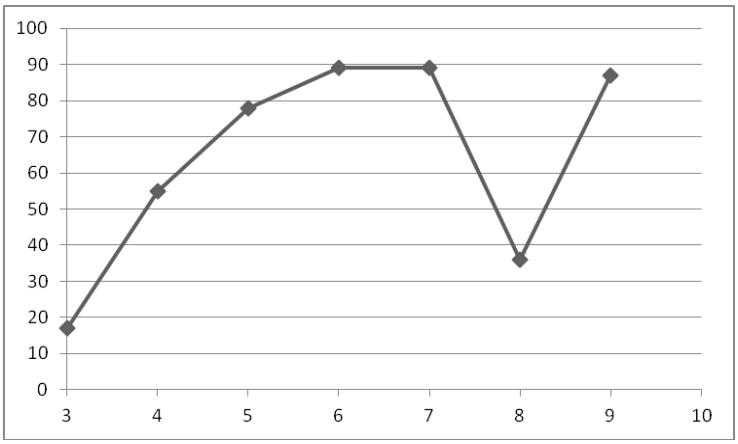
Pav. 4.5 Rezultatų kitimas, lyginant su septyniems sprendimams apmokytu tinklu

Didėjant raudonųjų pajėgas valdančio neuroninio tinklo apmokymų skaičiui, rezultatai tampa tolygūs. Tai galima stebėti 4.6, 4.7 pav. grafikų pradžiose. Nors mėlynosios pajėgos išlaiko persvara kurią joms suteikia aplinka, raudonąsias pajėgas valdančio neuroninio tinklo pasirinkti sprendimai

suformuoja tikslingesnį raudonųjų pajėgų judėjimą aplinkoje, ir pašalina galimybę išgauti atsitiktinį rezultatą. Tolesnis mėlynąsias pajėgas valdančio neuroninio tinklo apmokymas tik didina jų persvara.



Pav. 4.6 Rezultatų kitimas, lyginant su aštuoniems sprendimams apmokytu tinklu



Pav. 4.7 Rezultatų kitimas, lyginant su devyniems sprendimams apmokytu tinklu

Rezultatų, gautų įvykdžius eksperimentą dėsningumai parodo, jog neuroninis tinklas gali tikslingai valdyti veikėjus realaus laiko virtualioje aplinkoje. Jo sprendimų kokybė priklauso nuo sukauptos žinių bazės dydžio ir tikslumo.

5. IŠVADOS

1) Išanalizavus būdus pasirinkti veikėjo veiksmus, pasirinktas procedūrinio taktinių veiksmų parinkimo būdas. Nustatyta, kad šiuo būdu kur kas paprasčiau įvertinti daugybe įvairių veikėjo aplinkos parametrų, nei sudarinėjant sudėtingas logines struktūras, tokias kaip sprendimų medis.
2) Atlikus galimų būdų optimizuoti veikėjų veiksmus analizę, tam pasirinktas genetinis algoritmas.
3) Išanalizavus būdus atpažinti, daugybe tiesiogiai nesusijusių parametrų išreikštą situaciją ir pateikti jai sprendimą, pasirinktas aciklinis neuroninis tinklas.
4) Atlikus eksperimentą parodyta, kad didesni neuroniniai tinklai gali išreikšti didesnę sprendimų erdvę. Tačiau jų apmokymas užtrunka daug daugiau laiko. Todėl neuroninio tinklo dydis turi būti pritaikytas apmokymo pavyzdžių kiekiui, norint optimizuoti tinklo apmokymo trukmę ir galimybes.
5) Atlikus eksperimentą paaiškėjo, kad specialiai apmokytas aciklinis neuroninis tinklas gali tikslingai reaguoti į situacijas, susidarančias realaus laiko virtualioje aplinkoje, ir modeliuoti veikėjų veiksmus joje.

6. LITERATŪRA

1. Straatman R., Sterren W., Beij A., Killzone‘s AI: dynamic procedural combat tactics, 2005, [interaktyvus] [žiūrėta 2013-04-10]. Prieiga per internetą: http://www.cgf-ai.com/docs/straatman_remco_killzone_ai.pdf
2. Champandard A., Waypoint Cover Maps and Efficient Raycasts on PS3 in Killzone 2, 2009, [interaktyvus] [žiūrėta 2013-04-10]. Prieiga per internetą: http://aigamedev.com/insider/coverage/waypoint-cover-maps/
3. Milington I. Artificial Intelligence For Games. Elsevier, 2005.
4. Kennedy J., Eberhart R. Particle Swarm Optimization, 1995, [interaktyvus] [žiūrėta 2013-05-10]. Prieiga per internetą: http://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading6%201995%20particle%20swarming.pdf
5. Russell S., Norvig P. Artificial Intelligence A Modern Approach Second Edition. Pearson Education, 2003.
6. Rojas R. Neural Networks A Systematic Introduction. Springer, 1996

7. PRIEDAI

7.1. priedas. Vartotojo instrukcija

Pagrindiniai sistemos veikimo nustatymai nurodomi setup.xml faile. Sistemos darbas skirstomas į tris pagrindinius procesus: žinių bazės formavimą genetiniu algoritmu, neuroninio tinklo apmokymą ir veikėjų veiksmų modeliavimą realiu laiku.

Norint genetiniu algoritmu suformuoti žinių bazę neuroninio tinklo apmokymui, nustatymų faile parametras „RunGA“ nustatomas „true“, parametre „Iterations“ įrašomas norimų atlikti iteracijų skaičius, parametre „BlueBatchSaveFile“ nurodomas failas, kuriame išsaugoma mėlynųjų pajėgų žinių bazė, o parametre „RedBatchSaveFile“ nurodomas failas, kuriame išsaugoma raudonųjų pajėgų žinių bazė. Paleidus programą genetinis algoritmas generuoja sprendimus, o jų veikimas atvaizduojamas ekrane. Įvykdžius nurodytą kiekį iteracijų veikėjai sustoja ir programą galima išjungti.

Norint apmokyti neuroninį tinklą turima žinių baze, nustatymų faile parametras „TrainNN“ nustatomas „true“, o parametras „RunGA“ nustatomas “false“. Parametru „TrainTeam“ pasirenkama kurias pajėgas valdantį neuroninį tinklą norima apmokyti. Galimi variantai yra „blue“ arba „red“. Parametru „BatchSize“ nurodomas sprendimų iš žinių bazės, kuriais bus apmokomas neuroninis tinklas kiekis. Parametrais „BlueBatchLoadFile“ ir „RedBatchLoadFile“ pasirenkami apmokyme naudojamų duomenų bazių failai, atitinkamų pajėgų neuroniniams tinklams. Parametrais „BlueNNSaveFile“ ir „RedNNSaveFile“ nurodomi failai, kuriuose bus saugojami sudaryti neuroniniai tinklai. Paleidus programą bus apmokomas tik vienas neuroninis tinklas. Apmokymas gali labai užsitesti. Jam įvykus, sistema pradės modeliuoti veikėjų veiksmus pagal ką tik apmokytą neuroninį tinklą.

Norint modeliuoti veikėjų veiksmus parametrai „RunGA“ ir „TrainNN“ nustatomi „false“. Parametrais „BlueNN“ ir „RedNN“ pasirenkami modeliavime naudojami neuroniniai tinklai atitinkamos pajėgoms. Parametru „RunMatches“ nurodomas modeliuojamu susirėmimų kiekis. Paleidus programą bus modeliuojami veikėjų veiksmai, iki bendras laimėjimų kiekis pasieks nurodytą skaičių.

Sistemos veikimo metu papildomam valdymui naudojami šie mygtukai:

1) Esc – išjungia programą.
2) F1 – nustato įprastą sistemos veikimo greitį.
3) F2 – nustato 4 kartus didesnį nei įprastą sistemos greitį.
4) F3 – nustato 9 kartus didesnį nei įprastą sistemos greitį.
5) F4 – nustato 15 kartų didesnį nei įprastą sistemos greitį.
6) F5 – nustato 100 kartų didesnį nei įprastą sistemos greitį.
7) Space – sustabdo sistemos veikimą, paspaudus antrą kartą nustato įprastą sistemos veikimo greitį.

Stebėjimo taškas valdomas klaviatūros rodyklėmis. Stebėjimo kampas pasukamas pele, įspaudus dešinyjį pelės mygtuką.

7.2. priedas. Duomenų laikmena