VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Kursinis darbas

Genetiniai algoritmai dirbtiniams neuronų tinklams

(Genetic algorithms for neural networks)

Atliko: 3 kurso 1 grupės studentas
Laimonas Beniušis (parašas)

Darbo vadovas:
Linas Litvinas (parašas)

Vilnius – 2017

Turinys

Įvadas

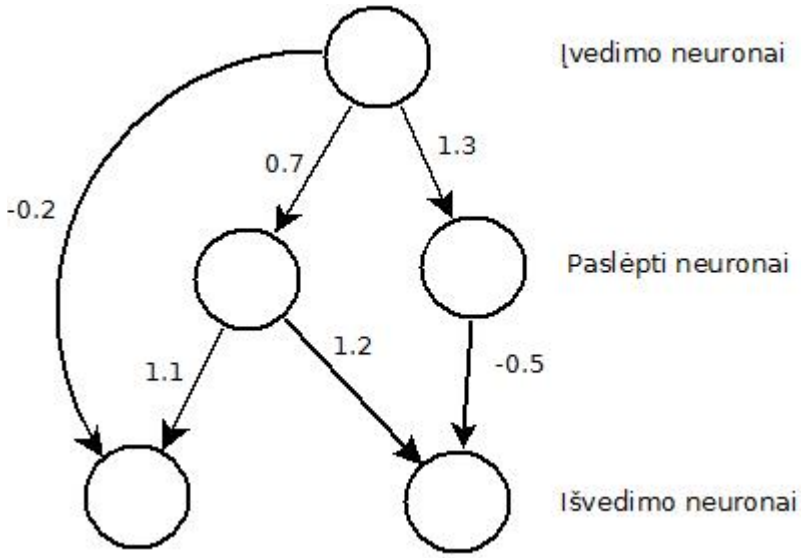
Dirbtinis neuronų tinklas (angl. *Artificial Neural Network*) gali būti pritaikomas įvairiems uždaviniams spręsti, kaip simbolių atpažinimas, paveikslėlių kategorizavimas ar dirbtinio intelekto mokymas. Didžioji tokių uždavinių dalis yra struktūrų atpažinimas (angl. pattern recognition), ką žmonės geba atlikti natūraliai.

Populiarus „YouTube“ vaizdo įrašas pavadinimu „MarI/O - Machine Learning for Video Games“ [Set15], pritraukė didžiulį susidomėjimą genetiniais algoritmais ir dirbtiniais neuronų tinklais. Šiame vaizdo įraše dirbtinis neuronų tinklas genetinės evoliucijos būdu apmokomas greitai įveikti pirmąjį „Super Mario World“ žaidimo lygį. Tai tik įrodo, kad genetinės evoliucijos taikymo sritis yra labai plati. Šiame darbe yra nagrinėjamas naudotas genetinis algoritmas, kuris sprendžia tinkamos DNT struktūros radimo uždavinį artėjant prie sprendinio, minimizuojant tinklo sudėtingumą evoliucijos proceso metu.

1 Dirbtiniai neuronų tinklai (DNT)

Dirbtinių neuronų tinklų yra įvairiausių formų, tačiau šiame darbe yra nagrinėjami tik tiesioginio sklidimo dirbtiniai neuronų tinklai. Tokio tipo tinklas matematiškai yra funkcija [Phi94, 8] ir tokio tinklo struktūrą gali apibūdinti orientuoto beciklio svorinio grafo analogija. Tinklą sudaro neuronai (grafo viršūnės) ir jungtys. DNT tinklo struktūra:

- Nekintantis kiekis įvedimo neuronų
- Nekintantis kiekis išvedimo neuronų
- Dinaminis arba nekintantis kiekis paslėptų neuronų
- Dinaminis arba nekintantis kiekis paslėptų neuronų sluoksnių
- Kiekvienas neuronas turi aktyvacijos funkciją ir (nebūtina) slenksčio reikšmę
- Dinaminis arba nekintantis kiekis jungčių, kurios apjungia neuronus
- Kiekviena jungtis turi svorio reikšmę



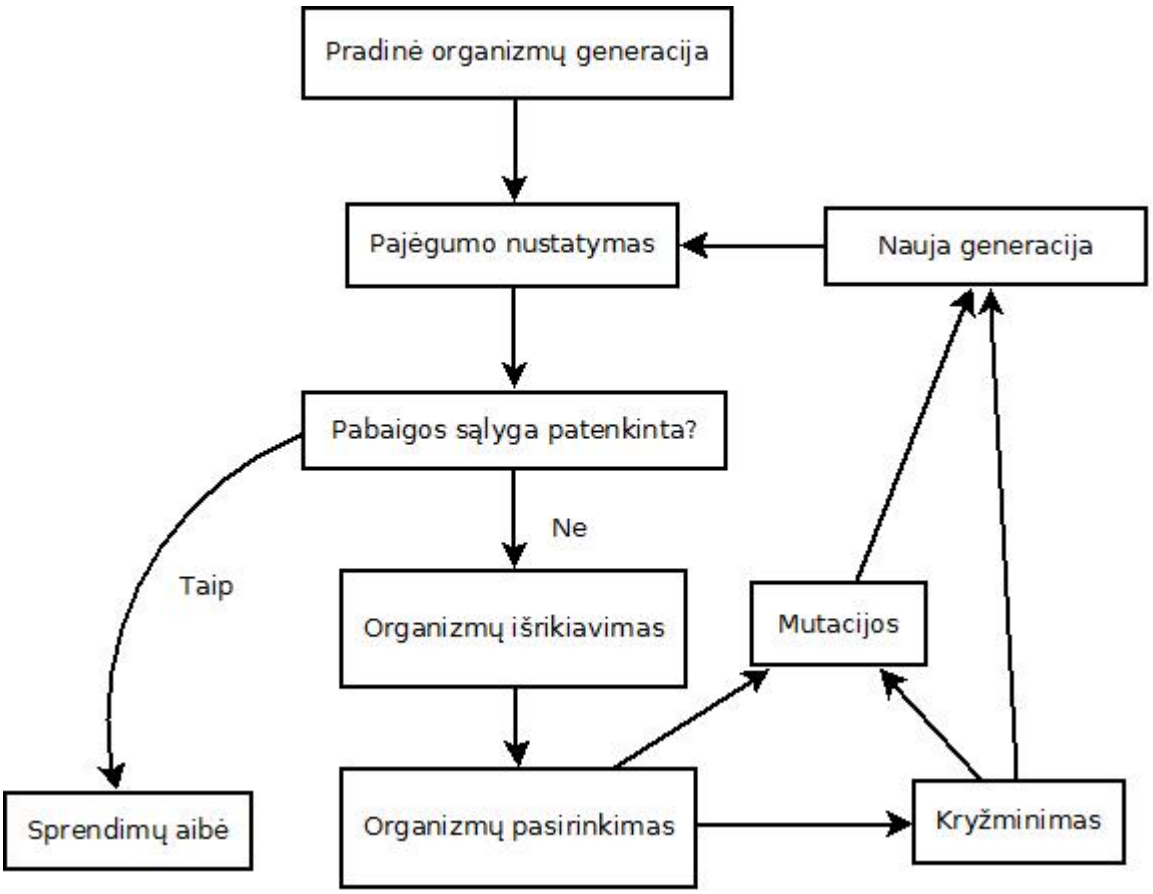
1 pav. Paprastas dirbtinis neuronų tinklas

Rezultatas yra pasiekiamas sudauginant kievieno neurono įeinančius svorius ir perleidžiant juos per aktyvacijos funkciją. Kadangi DNT yra beciklio grafo struktūros per kiekvieną neuroną yra pereinama topologine tvarka. Tokiu būdu yra gaunama kompozicinė funkcija.

DNT yra mokomas t. y. koreguojami jungčių svoriai, kurie duoda vis kitokį rezultatą. Kartais, problema yra tokia, kada žinome norimą rezultatą, tada galima naudoti mokymą su mokytoju (angl.

Supervised learning), tačiau dažnai galime tik palyginti kuris iš rezultatų yra geresnis, toks mokymas yra vadinamas paspartintu (angl. *Reinforcement learning*). DNT apmokymo greitis priklauso nuo DNT struktūros, nes tam tikros taisyklės ar dėsningumai, kurie pasirodo sprendžiamo uždavinio duomenyse gali atitikti skirtingus ryšius. Todėl tą pačią problemą gali išspręsti skirtingų struktūrų DNT ir jų apmokymo greitis gali ženkliai skirtis.

2 Genetiniai algoritmai (GA)



2 pav. Genetinio algoritmo veiksmų seka

Genetiniai algoritmai (toliau GA) yra evoliucinių algoritmų poklasis. Evoliuciniai algoritmai simuliuoja organizmų evoliuciją ir panašiai kaip gamtoje, išlieka stipriausi. Tai yra metaeuristinis optimizavimo metodas. Generuojant naują organizmų aibę yra naudojami įvairūs metodai kaip kryžminimas (angl. crossover) ar paprasčiausias klonavimas. Evoliucijoje dalyvaujantys organizmus galima vadinti genomais (genotipais). Genotipas yra genų sąrašas, o vizuali genotipo forma arba genų realizacija yra fenotipas [Phi94, 11][KW16, 3].

3 Neuro-Evoliucija (NE) ir jos aktualumas

Neuro-Evoliucija (toliau NE), tai DNT evoliucija naudojant GA, kuri pasirodė perspektyvi sudėtingose paspartinto mokymo problemose [GM97] [GM99] [MM96].

Pilnai jungus dviejų paslėptų sluoksnių DNT iš principo gali apytiksliai apskaičiuoti visas tolydžias funkcijas [Cyb89,305], todėl gali kilti klausimas, kodėl reikia gaišti laiką apskaičiuojant įvairias topologijas? Tradiciškai, Neuro-Evoliucijos metoduose tinklo topologija yra pasirenkama prieš pradedant ekspetimentą, todėl tikslas tampa optimizuoti lankų svorius, tačiau tai nėra vienintelis DNT aspektas, darantis įtaką jų elgesiui. Topologija, arba DNT struktūra irgi keičia jo funkcionalumą [KR02, 102].

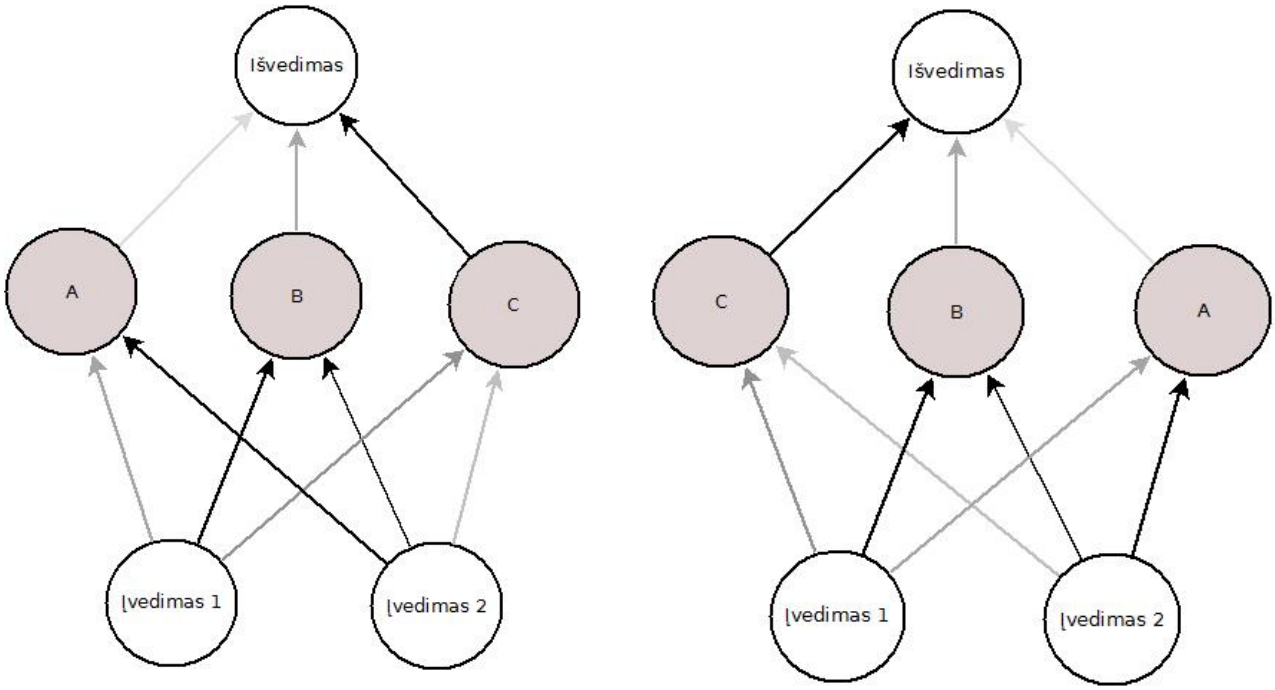
Nors visi NE metodai naudoja pilnai jungią DNT, tinkamo paslėptų neuronų skačiaus parinkimas bandymų ir klaidų būdu yra laiko švaistymas. Efektyvus NE metodas keičiant lankų svorius ir tinklo topologiją buvo pritaikytas strypo balansavimo problemai spręsti [GWP96]. Tačiau vėliau buvo įrodyta, kad tokios sudėtingos tinklo struktūros nereikėjo. Statinės topologijos NE metodas pavadinimu „Enforced Subpopulations“ (ESP) [GM02] rado sprendimą 5 kartus greičiau, tiesiog pradėdamas su atsitiktiniu paslėptų neuronų skaičiumi kada progresas užstrigdavo.

4 Neuro-Evoliucija Augančioms Topologijoms (NEAT)

Augančių topologijų neuro-evoliucija (angl.*Neuro Evolution of Augmenting Topologies*) yra NE algoritmas, sukurtas Kenneth O. Stanley ir Risto Miikkulainen [KR02]. Šis algoritmas minimizuoja topologijas evoliucijos metu, o ne tik jos pabaigoje, nenaudojant specialios funkcijos, kuri matuoja DNT sudėtingumą. Taip pat DNT struktūros sudėtingėjimas koreliuoja su jo sprendinio korektiškumu, t. y. nėra nereikalingų neuronų ar jungčių.

4.1 Konkuruojantys sprendimai

Neuro-Evoliucijoje buvo vengiama kryžminimo (kas yra vienas iš pagrindinių GA principų) todėl, kad kryžminimas dažnai yra nenaudingas, nes DNT praranda funkcionalumą. Dėl šios priežasties kaikuriuose Neuro-Evoliucijos metoduose kryžminimo yra visiškai atsisakyta. Tai vadinama Evoliuciniu Programavimu [YL96].



[A, B, C] [C, B, A]

3 pav. Funkciškai vienodi tinklai. Jungčių ryškumas simbolizuoja reikšmę.

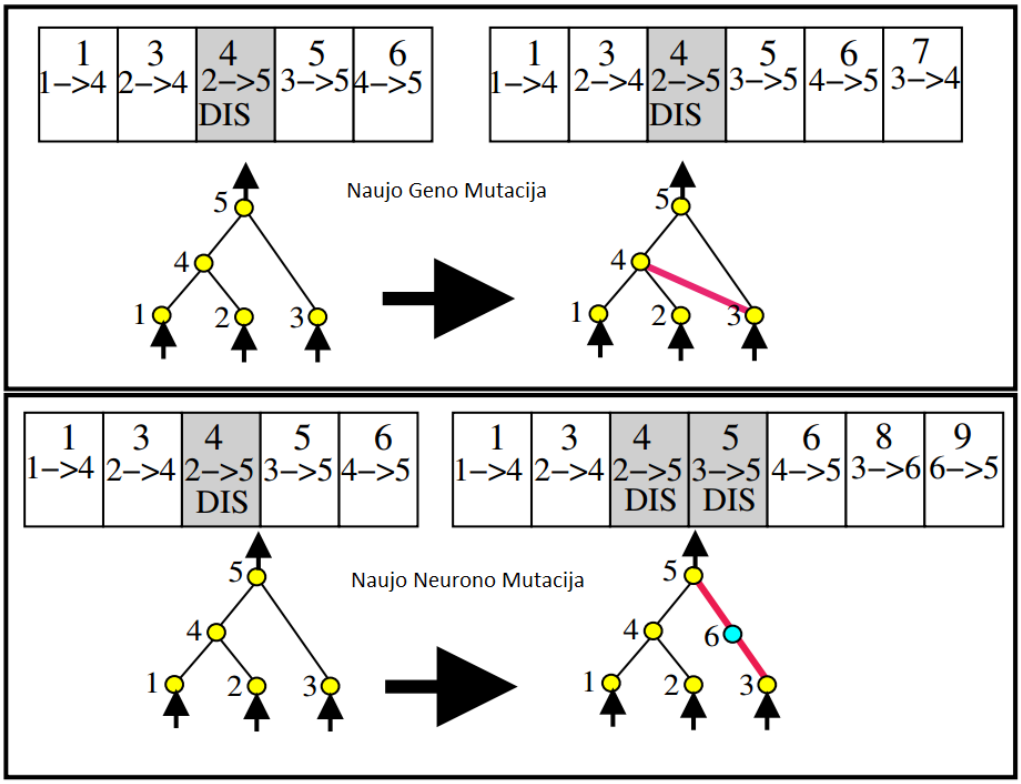
Galime nesunkiai įsitikinti, kad turime du funkcionaliai vienodus tinklus, tačiau jų paslėptų neuronų išsidėstymas daro juos skirtingais kryžminimo procese. Kryžminimo metu (B yra bendras

požymis) [A, B, C]×[C, B, A] galimi vaikai yra [A, B, A] ir [C, B, C]; todėl prarandama informacija. Ši problema yra žinoma kaip konkuruojantys sprendimai (angl. *Competing Conventions*). Vienas šios problemos iš sprendimų yra atlikti topologinę analizę, tačiau ji yra sudėtinga ir lėta, todėl reikia DNT kodavimo bei būdo juos efektyviai kryžminti [KR02, 103].

4.2 Genetinis Kodavimas

NEAT algoritme genas yra DNT lankas, kuris turi svorį, įėjimo ir išėjimo neuroną, gali būti išjungtas bei turi inovacijos žymę, kuri gali būti paprastas globalus skaitliukas [Ken04, 34]. Inovacijos žyme yra homologinis rodiklis tarp tėvų ir vaikų, t.y. paveldėti bruožai (šiuo atveju DNT struktūra). Ši žymė yra naudojama kryžminimo procese, nes tada galima atskirti sutampančius genus (svoriai gali skirtis) ir informacija yra išsaugojama. Naudojamas tiesioginis kodavimas, t. y. genai tiesiogiai atitinka fenotipą.

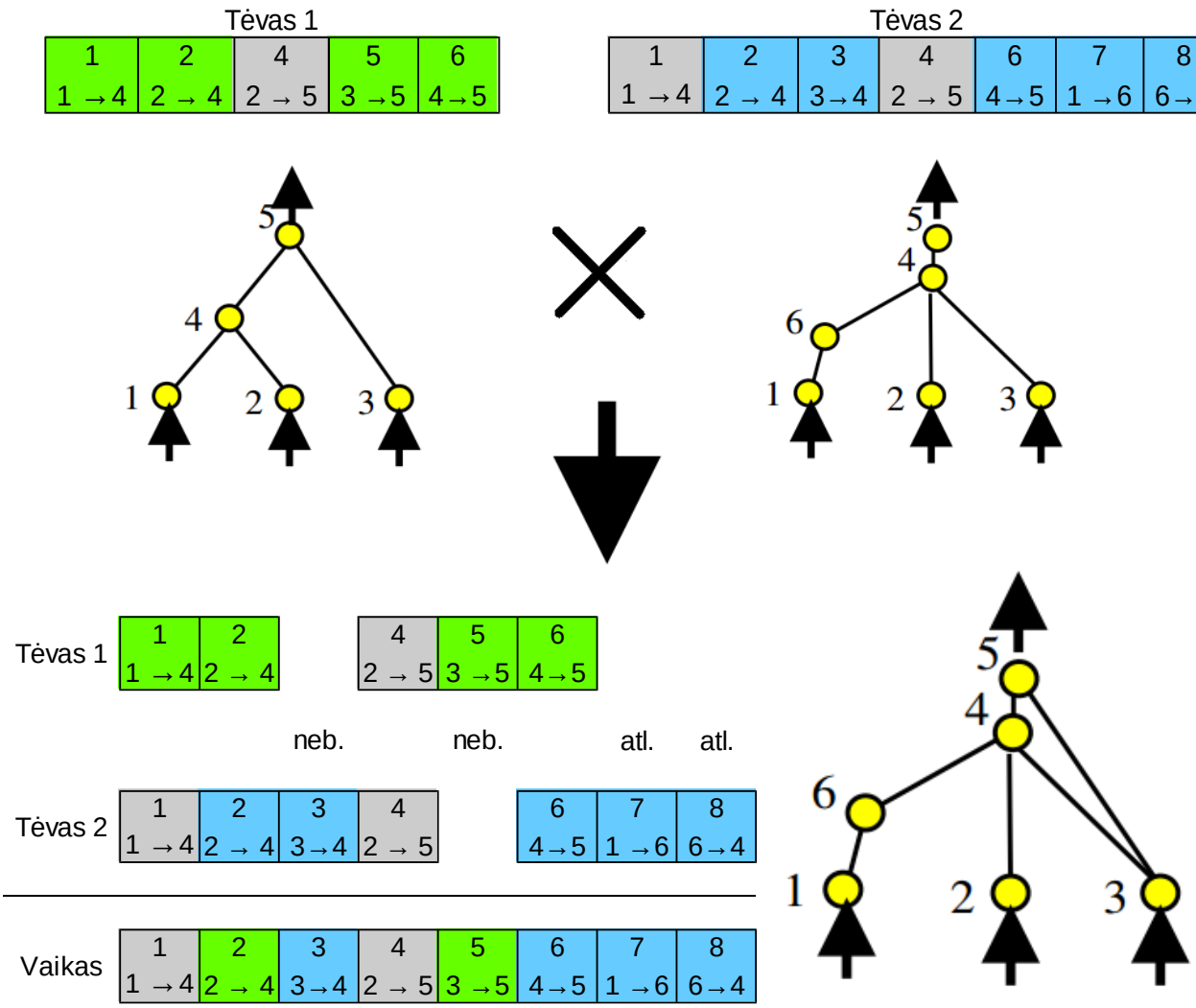
4.3 Mutacijos



4 pav. NEAT algoritme naudojamos mutacijos [Ken04, 36]. Pilki genai yra išjungti. Naujo neurono atveju senas genas išjungiamas, o jo vietoje atsiranda 2 nauji genai, sujungti naujo neurono. Mutacijų metu genai gali būti išjungiami.

Kieviena naujo geno sukūrimo mutacija turi savo inovacijos žymę (viršuje). Vienodos genų mutacijos skirtinguose genomuose įgauna vienodą inovacijos žymę. Taip pat yra naudojamos standartinės NE mutacijos, kurios keičia lankų svorius.

4.4 Kryžminimas



5 pav. Skirtingų DNT kryžminimas pagal inovacijos žymes.

Nors ir tėvų fenotipai skiriasi, inovacijos žymės (geno viršuje) parodo sutampančius genus, kuriuos suporavus galima vykdyti kryžminimą [Ken04, 37]. Naudoti žymėjimai:

- neb. (nebendri) genai yra tie, kurių inovacijos žymės neviršija mažesniojo tėvo didžiausiosios.
- atl. (atliekami) genai yra tie, kurie viršija mažesniojo tėvo didžiausiąją inovacijos žymę.
- pilki genai yra išjungti

4.5 Konkuravimo išskirstymas naudojant rases

Dažniausiai, įterpiant naują atsitiktinį geną į genomą sumažėja jo pajėgumas, todėl globalioje populiacijoje modifikuotas genomas ilgai neišliks. Dėl šios priežasties yra įvedama lokali populiacija (angl. *species*), kurios genomai yra homologiškai panašūs ir konkuriuoja tarpusavyje, o ne globaliai.Tačiau kaip atskirti ar genomai yra iš tos pačios rasės? Inovacijos žymės ir vėl suteikia paprastą būdą šiai problemai spręsti. Genomų panašumo funkcija:

(1) $$d = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 W$$

E - atliekamų (angl. excess) genų skaičius, D - nebendrų (angl. disjoint) genų skaičius, W - sutampančių genų svorių skirtumų vidurkis, N - didesniojo genomo genų skaičius, c_1, c_2, c_3 nustato šių daugiklių įtaką. „Kuo mažiau yra atliekamų ir nebendrų genų genomuose, tuo jie yra artimesni evoliucinės istorijos prasme“ [KR02, 112].

Kiekvienai rasei atstovauja atsitiktinis genomas iš ankstesnės generacijos, kuris yra naudojamas naujos generacijos genomams sugrupuoti. Jeigu naujas genomas netinka visoms esamoms rasėms, yra priskiriamas naujai ir tampa jos atstovu.

Norint, kad viena rasė nedominuotų globalios populiacijos, būtinas specialus reprodukcijos mechanizmas, nes paprastos topologijos DNT optimizuojasi greičiau. NEAT algoritmas naudoja pajėgumo paskirstymą (angl. Explicit fitness sharing) [GR87], kada pajėgios rasės genomai turi dalintis savo pajėgumu su visa populiacija.

(2) $$f_i^{new} = \frac{f_i}{\sum_{j=1}^{n} sh\left(d\left(i,j\right)\right)}$$

Naujas pasvertas genomo *i* pajėgumas yra apskaičiuojamas pagal genomų panašumo funkciją su visais kitais populiacijoje esančiais genomais *j*. Pajėgumo paskirstymo funkcija *sh* gauna reikšmę 0, jeigu *d* funkcijos reikšmė viršija nustatytą panašumo ribą, kuri gali kisti nuo generacijos numerio, priešingu atveju gauna 1 [Spe95]. Tokiu būdu per pajėgumas yra mažinamas proporcingai nuo rasės dydžio, todėl yra išvengiama vienos rasės dominavimo.

Skirstymas į rases apsaugo genomų topoligines mutacijas nuo ankstyvo eliminavimo ir suteikia šansą suoptimizuoti naujai įgytus genus. Mažiausiai pajėgūs rasės genomai yra pašalinami (pvz.

mažesnio negu vidutinio pajėgumo). Generuojama nauja populiacija proporcingai kryžminant kiekvienos rasės genomus bei klonuojant geriausio pajėgumo genomus iš buvusios generacijos. Taip pat skirstymas į rases toleruoja dažnas mutacijas, kas spartina optimizavimą [Ken04, 41].

Jeigu taip atsitiks, kad tam tikra rasė nepadaro progreso per pasirinktą kiekį generacijų, galima jai neleisti daugintis ir sutelkti resursus kitoms rasėms. Reikėtų išlaikyti bent 2 skirtingas rases, nes tada pajėgumo paskirstymo funkcija nebeturi prasmės ir stabdo progresą.

Genomo priskirimo rasei ciklas [Ken04, 39]:

1. Pasirinkti sekantį genomą *g* iš populiacijos *P*
2. Rasės ciklas:
    1. Jeigu visos rasės iš rasių sąrašo *S* buvo patikrintos, sukurti naują rasę ir įterpti į ją *g*
    2. Priešingu atveju:
        1. Pasirinkti netikrintą rasę *s* ir jeigu *g* turi pakankamai panšumų su *s* rasės atstovu, *g* priskirti *s* rasei
    3. Jeigu *g* nebuvo priskirtas rasei, tęsti rasės ciklą
3. Jeigu visi genomai buvo priskirti, baigti

4.6 NEAT rezultatai ir variantų palyginimas

Yra daugybė kontroliavimo užduočių, kurioms galima pritaikyti NEAT algoritmą kaip robotų navigaciją ar žaidimų žaidimas [Ken04, 44]. Dvigubo strypo balansavimo užduotis (du skirtingo ilgio strypai ant vagono) yra žinomas algoritmų tikrinimo įrankis, kuris buvo pasirinktas NEAT algoritmo efektyvumui tirti, nes vieno strypo balansavimas tapo per lengvas moderniems algoritmams. Rezultatai buvo palyginti su 3 tradiciniais funkcinio pagrindo paspartinto mokymo metodais ir 4 kitais
neuro-evoliucijos metodais.

4.6.1 Funkciniai metodai

Funkciniai metodai išanalizuoja kiekvieno veiksmo rezultatą bandydami specifinius veiksmus esamoje sistemos būsenoje. Veiksmų erdvė yra diskreti ir funkcijos aproksimatorius išmokina Q-funkciją sudarydamas būsenos-veiksmo poras [Ken04, 45].

Multisluoksninio perceptrono Q-Mokymas (Q-MLP) [WD92] naudoja tiesioginio sklidimo DNT kaip savo funkcijos aproksimatorių. Tinklas įvedimo reikšmės yra būseną ir veiksmo kintamuosius, o išvedimo viena Q-reikšmė.

Sarsa(λ) su funkciniu aproksimatoriu skirtingiems atvejams (angl. *Sarsa(λ) Case Based Approximator*) (SARSA-CABA)[SJS+98] išsaugoja būsenos-veiksmo poras kaip distinktyvius atvejus. Nauji atvejai yra apskaičiuojami kombinuojant kaimyninių atvejų reikšmes.

Sarsa(λ) su smegenų modeliu artikuliacijos kontrole (angl. *Sarsa(λ) Cerebellar Model Articulation Controller*) (CMAC)[Alb75], kuris pakeičia SARSA-CABA distinktyvių atvejų aibės kūrimą į smegenų modelio artikuliacijos kontrolės modelį. Šis metodas padalina būsenos-veiksmo paieškos erdvę į persidengiančius sluoksnius. Q-reikšmės yra apskaičiuojamos sudedant reikšmes iš skirtingų sluoksnių su vienoda reikšme.

Reikšmių ir taisyklių paieška (*angl. Value and Policy Search*) (VAPS) [MPK+96] ieško ne būsenos-veiksmo porų, o taisyklių. VAPS ieško baigtinio automato formos grafų naudojanti gradientinį nusileidimą.

4.6.2 Neuro-evoliuciniai metodai

Standartiniai evoliuciniai metodai (populiacijų pagrindu) naudoja kryžminimą bei mutacijas [Sar95] ir evoliucinis programavimas, kuris apsiriboja svorių mutacijomis [Wie91].

Kitos dvi sistemos yra SANE [MM96] ir ESP [GMF3], evoliucionuoja neuronų populiacijas ir neuronų tinklo projektų populiacijas, kuri nurodo kaip sukonstruoti statinės topologijos DNT. SANE metodas išsaugo tik vieną populiaciją, o ESP išsaugo atskirą populiaciją kiekvieno paslėpto neurono pozicijai.

1 lentelė. Dvigubo strypo balansavimo užduoties rezultatai. Evoliucinio Programavimo rezultatų šaltinis [Sar95]. SARSA ir VAPS metodai nesugebėjo rasti sprendimo per 12 valandų [Gom03]. NEAT metodas išsprendė užduotį su mažiausiai DNT įvertinimų negu visi kiti metodai ir yra greitesnis negu kiti paspartinto mokymo metodai, išskyrus ESP.

Metodas	**DNT įvertinimai**	**Sugaištas laikas (sekundės)**
Q-MLP	**10582**	153
SARSA-CABA	Nebaigė	-
SARSA-CMAC	Nebaigė	-
VAPS	Nebaigė	-
Evoliucinis Programavimas	**307200**	-
Tradicinė Neuro-evoliucija	**22100**	73
SANE	**12600**	37
ESP	**3800**	22
NEAT	**3600**	31

2 lentelė. NEAT variantų palyginimas dvigubo strypo balansavimo užduotyje. Eksperimento rezultatų šaltinis [Ken04, 52].

NEAT variantas	**DNT įvertinimai**	**Sprendinys rastas**
Statinės topologijos	**630239**	20%
Nenaudojantis rasių	**75600**	75%
Su pradine atsitiktine topologija	**30927**	95%
Nenaudojantis kryžminimo	**5557**	100%
Pilnas	**3600**	100%

4.7 NEAT struktūra

NEAT algoritmo pradinė DNT struktūra yra dviejų sluoksnių, kada kiekvienas įėjimo neuronas yra sujungtas su išėjimo neuronu [Ken04, 40]. Galima pradėti su atsitiktiniu pasleptų neuronų skaičiumi, tačiau dažnai tokia stuktūra nėra išnaudojama, beto, tai prieštarauja pagrindinei NEAT idėjai, kuri siekia užtikrinti minimalaus sudėtingumo DNT struktūrą ištisos evoliucijos metu. Žinoma, jeigu pasirodys, kad tam tikrų genų nereikia, jie bus išjungti, tačiau tai bus papildomas optimizavimo darbas. Taigi, pradedant su minimalia DNT struktūra, evoliucijos metu ji sudėtingėja, auga ir kiekvienas topologinis pokytis yra pateisintas [Ken04, 40]. DNT augimui aktivacijos funkcijos neturi tiesioginės įtakos.

4.8 Galimi pokyčiai algoritme

Gali pasitaikyti reta situacija, kada NEAT algoritmas išoptimizuos tam tikrus genus, kurių reikia korektiškam sprendimui pasiekti. Galimas problemos sprendimas yra vietoje genų išjungimo pakeisti jų svorį į 0. Tokiu būdu yra suteikiamas šansas atstatyti reikšmę vėliau, kada (jeigu) jos prireiks [NFAQ]. Taip pat NEAT algoritmas visiškai atsiriboja nuo pačių neuronų mutavimo, todėl galima suteikti neuronams slenkščio (angl. *bias*) reikšmę, kuri gali kisti (mutuoti) ir kurios evoliucijos raidos nereikia sekti su inovacijos žymėmis, nes kryžminime neuronai tiesiogiai nedalyvauja.

4.9 NEAT išlygiagretinimas

„Vienas iš didžiausių motyvacijų testi genetinę paiešką yra tai, kad ji yra iš prigimties lygiagreti. Natūraliose populiacijose tūkstančiai ar net milionai individų egzituoja lygiagrečiai“ [Whi]. Taigi, natūralu NEAT peiškoje naudoti lygiagrečius skaičiavimus.

Paprasčiausiai išlygiagretinami atskirų genomų pajėgumo skaičiavimai, nes jų operacijos yra visiškai nepriklausomos. Prieš generavimą reikia pašalinti mažiausiai pajėgius genomus, o tai galima padaryti tik išrikiavus juos pagal pajėgumą [KW16, 11]. Barjero mechanizmo tam pilnai pakanka. Genomų pajėgumų įverčiai gali būti pradėti skaičiuoti dar nebaigus generuoti naujos genomų aibės, tačiau pasvertam įverčiui apskačiuoti reikia žinoti visų kitų genomų pajėgumą pagal (2) funkciją.

4.10 NEAT pritaikymas simuliacijose

NEAT algoritmas turi labai platų pritaikymo spektrą, t. y. uždaviniai, kuriuos galima spręsti gali ženkliai skirtis. Labai dažnas NEAT pritaikymo scenarijus yra simuliacijos. Tai gali būti tam tikro skeleto išmokymas eiti tieisiai ir balansuojant, organizmo orientavimasis grėsmingoje aplinkoje [Ken04, 114], automobilio vairavimas ir kliūčių vengimas [Ken04, 101] bei dirbtinio intelekto mokymas įvairiuose žaidimuose [Ken04, 90]. Problemų iškyla kai simuliacijos turi nedeternimistinių kintamųjų. Tada genomo pajėgumas gali ženkliai suprastėti, jeigu simuliacijos eigoje pasitaikys nepalankių sąlygų. Tokią problemą galima lengvai išspręsti pakartotinai vykdant simuliaciją ir genomo pajėgumą įvertinant vidutiniškai [Ken, 105]. Toks sprendimas yra praktiškas tik jeigu simuliacijos įvykdymas yra pakankamai greitas.

5 Realaus laiko NEAT (rtNEAT)

Tradiciniai evoliuciniai algoritmai vyksta generacijų iteracijomis, t. y. naujos generacijos genomai skiriasi elgesiu nuo ankstesnės, todėl tai yra nenatūralu. Taip pat genomų elgesys išlieka toks pats tarpuose tarp generacijos iteracijų. rtNEAT (angl. *Real time Neuro-Evolution of Augmenting Topologies)* algoritmas modifikuoja NEAT, kad neuro-evoliucija galėtų vykti pastoviai (realiu laiku). Nustatytu laiko intervalu vienas genomas yra pakeičiamas nauju. Vienas iš mažiausiai pajėgiausių genomų yra pakeičiamas labiausiai pajėgių tėvų vaiku, todėl reikia naujo reprodukcijos algoritmo [Ken04, 116], norint išlaikyti originalaus NEAT algoritmo aspektus kaip paskirtytas pajėgumas (2) ir skirtymas į rases. rtNEAT reprodukcijos ciklas:

1. Pašalinti genomą, kuris turi mažiausią pasvertą pajėgumą pagal (2) ir egzistavo pakankamai ilgą laiką, kad galėtų būti įvertintas
2. Perskaičiuoti visų genomų pasvertą pajėgumą
3. Pasirinkti tėvų rasę ir sukurti naują genomą g
4. Perskaičiuoti genomų panašumo ribą
5. Perskaičiuoti visų genomų rases

5.1 Reprodukcijos ciklo žingsnių aptarimas

Šio žingsnio tikslas yra pašalinti mažesnio pajėgumo genomą, kuris potencialiai bus pakeistas labiau pajėgiu. Genomas turėtų būti įvertintas pagal pasvertą pajėgumą (2), nes nepasvertas pajėgumas dažniausiai sumažėja padidinant DNT topologiją, todėl naujovės (naujos jungtys) būtų greitai pašalinamos, taip stabdydamos progresą. Taip pat yra prasmingą įvesti amžiaus parametrą į kurį atsižvelgiant nebūtų pašalinimui pasirenkami per jauni genomai. Visų genomų pasvertas pajėgumas turi būti perskaičiuojamas, nes pasikeitė globali populiacija. Naujo genomo tėvai turėtų būti tos pačios rasės, nes stengiamasi maksimizuoti panašumus, todėl reikia būdo pasirinkti rasę. Šis būdas yra tikimybinis siekianti proporcingai išlaikyti rasių dydžius [Ken,117].

(3) $$Pr(S_k) = \frac{Avg_k}{Avg_{global}}$$

Avg_k- k rasės vidutinis pajėgumas, Avg_{global} - globalios populiacijos vidutinis pajėgumas, S_k – k rasė.

Kadangi genomų paskirstymas po rases priklauso nuo panašumo funkcijos, panašumo riba gali kisti kas kiekvieną reprodukcijos ciklą, išlaikydama norimą skirtingų rasių kiekį. Kadangi panašumo riba pakito, reikia perskaičiuoti visas rases.

Reprodukcijos ciklo intervalas taip pat yra svarbus rtNEAT parametras. Jeigu intervalas per mažas, evoliucija nevyks, nes nauji genomai negaus pakankamai laiko įvertinti savo pajėgumo. Jeigu intervalas per didelis, evoliucija vyks lėtai.

Apytikslis intervalo dažnumas gali būti apskaičiuojamas pagal šį principą [Ken,119]:

Tarkime I yra globalios populiacijos dalis, kuri yra per jauna, kad galėtų būti pašalinama, n yra reprodukcijos ciklo laiko intervalas, m yra minimalus pakeitimui tinkamo genomo laikas (amžius), o $|P|$ yra globalios populiacijos dydis. Tada gauname formulę:

(4) $$I=\frac{m}{|P|n}$$

Kuo didesnė populiacija ir reprodukcijos ciklo intervalas, tuo mažiau lieka per jaunų genomų.

Iš (4) formulės gauname reprodukcijos ciklo dinamiško intervalo formulę:

(5) $$n=\frac{m}{|P|I}$$

Bendru atveju yra patogu pasirinkti I parametro reikšmę ir dinamiškai keisti n parametrą, nes jeigu vienu yra per daug jaunų genomų, kryžminimui tinkamų genomų yra per mažai.
Pvz. jeigu $|P|$ yra 50, m yra 500 ir n yra 20, tada 50% genomų bus per jauni.

6 Genomo pajėgumo įvertinimas

Svarbus kiekvieno GA aspektas yra tinkamos pajėgumo įverčio funkcijos pasirinkimas. Labai dažnai prasta pajėgumo funkcija gali nukreipti evoliuciją visiškai klaidinga linkme [Ken04, 128]. Pvz.: Užduotis yra rasti išėjimą labirinte. Pajėgumas matuojamas pagal tiesioginį (euklido) atstumą nuo išėjimo. Jeigu aptikta aklavietė yra šalia išėjimo, tada algoritmui atrodys, kad tai yra teisinga linkmė ir visi ištekliai bus skirti būtent tai aklavietei tirti. Tokiu atveju atsitiktinai klaidžiojantis algoritmas turės geresnius šansus rasti išėjimą, negu aklavietę ištyręs algoritmas.

Be to reikia atsargiai skirti taškus už įgytą mažą pranašumą, nes algoritmas gali įklimpti į lokalų ekstremumą [Ken04, 112]. Pvz.: Vairuotojo modelio išmokymas vengti kliučių ir efektyviai naviguoti lenktynių trasoje. Tokiu atveju per daug atsargi pajėgumo funkcija gali įspėti vairuotoją apie kliūti, kuri neturi įtakos pasirinktai trajektorijai, todėl vairuotojas neoptimaliai apvažiuos kliūtis. Jeigu pajėgumo funkcija prioritizuoja greitį, vairuotojas pasirinks potencialiai nesaugią trajektoriją, todėl šių pajėgumo požymių balanso radimas nėra trivialus uždavinys ir dažniausiai gali būti išspręstas tik eksperimentuojant su skirtingomis reikšmėmis.

Taigi nėra universalaus būdo nustatyti geriausią pajėgumo funkciją pasirinktam uždaviniui, ypač jeigu tarpiniai žingsniai link problemos sprendinio nėra aiškūs.

7 Išvados

- Neuro-Evoliucija (NE) yra galingas įrankis įvairaus mąsto ir spektro problemoms spręsti
- NEAT algoritmo pagrindinė idėja, kuri leiždia efektyviai vykdyti NE, yra inovacijos žymės
- NEAT algoritmas yra efektyviai pritaikomas kontrolės sistemoms
- NE organizmų išskaidymas į rases yra efektyvus būdas paspartinti sprendimo radimą
- NE yra lengvai spartinama multiprocesinėje aplinkoje
- rtNEAT gali būti pritaikomas nuosekliose evoliucijos aplnkose pvz. simuliacijose
- Svarbus NE taikymo aspektas yra tinkamos pajėgumo funkcijos parinkimas

Literatūra

[KR02] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99, 127, 2002.

[Ken04] Kenneth O. Stanley. Efficient Evoliution of Neural Networks through Complexification. PhD thesis, The University of Texas, 2004

[Alb75] Albus, J. S. (1975). A new approach to manipulator control: The cerebellar model articulation controller. Journal of Dynamic Systems, Measurement, and Control, 97(3):220–227.

[GR87] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette. Proceedings of the Second International Conference on Genetic Algorithms, p. 148–154. San Francisco, CA: Morgan Kaufmann, 1987

[Cyb89] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems, 2(4):303–314.

[GM97] Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. Adaptive Behavior, 5:317–342.

[GM99] Gomez, F. and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In Dean, T., editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1356–1361, Morgan KaufmDNT, San Francisco, California.

[GM02] Gomez, F. and Miikkulainen, R. (2002). Learning robust nonlinear control with neuroevolution. *Technical Report* AI02-292, Department of Computer Sciences, The University of Texas, Austin, Texas

[Gom03] Gomez, F. J. (2003). Robust non-linear control through neuroevolution. PhD thesis, Department of Computer Sciences, The University of Texas at Austin. Technical Report AI-TR-03-303

[GWP96] Gruau, F., Whitley, D., and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R. et al., editors, Genetic Programming 1996: *Proceedings of the First DNTual Conference*, pages 81–89, MIT Press, Cambridge, Massachusetts.

[KW16] Kearney, William T., Using Genetic Algorithms to Evolve Artifcial Neural Networks (2016).Honors Theses. Paper 818. htp://digitalcommons.colby.edu/honorstheses/818

[MM96] Moriarty, D. E., and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32

[MPK+96] Meuleau, N., Peshkin, L., Kim, K.-E., and Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. *In Proceedings of the Fifteenth International Conference on Uncertainty in Artificial Intelligence.*

[NFAQ] Kenneth O. Stanley, The NeuroEvolution of Augmenting Topologies (NEAT) Users Page, FAQ, https://www.cs.ucf.edu/~kstanley/neat.html

[Phi94] Phillipp Koehn (1994) Combining Genetic Algorithms and Neural Networks: The Encoding Problem. Master Thesis. University of Tennessee, Knoxville

[SJS+98] Santamaria, J. C., Sutton, R. S., and Ram, A. (1998). Experiments with reinforcement learning in problems with continuous state and action spaces. Adaptive Behavior, 6(2):163–218.

[Set15] SethBling. MarI/O - Machine Learning for Video Games. (2015). https://youtu.be/qv6UVOQ0F44

[Sar95] Saravanan, N., and Fogel, D. B. (1995). Evolving neural control systems. *IEEE Expert*, 23–27.

[Spe95] Spears, W. (1995). Speciation using tag bits. *Handbook of Evolutionary Computation*. IOP Publishing Ltd. and Oxford University Press, Oxford, UK.

[WD92] Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[Whi] Darrell Whitley. A genetic algorithm tutorial. Computer Science Department, Colorado State University

[Wie91] Wieland, A. (1991). Evolving neural network controllers for unstable systems. *Proceedings of the International Joint Conference on Neural Networks,* p. 667–673. Piscataway, NJ:IEEE.

[YL96] Yao, X. and Liu, Y. (1996). Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, 91(1):83–90.