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

1 Įvadas

Dirbtinis neuronų tinklas, toliau DNT (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.

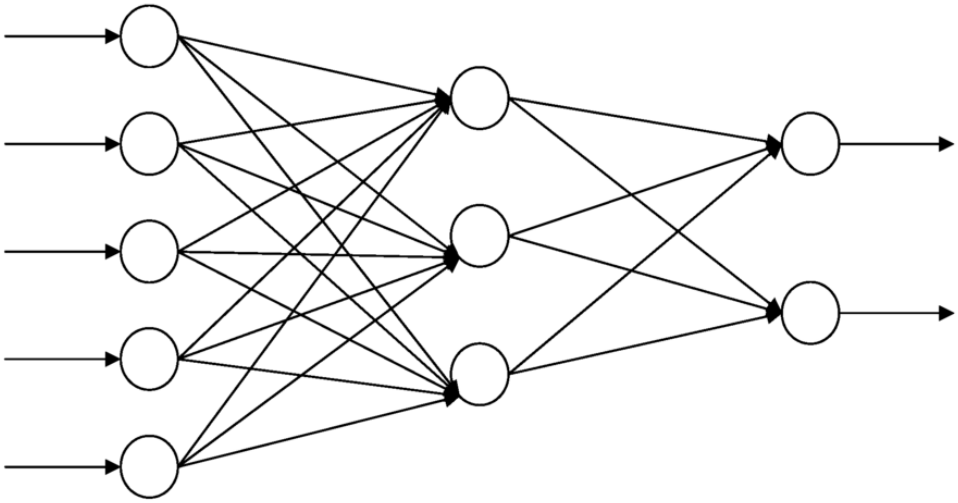
Šiame darbe yra nagrinėjamas genetinis algoritmas, kuris sprendžia tinkamos DNT struktūros radimo uždavinį artėjant prie sprendinio, minimizuojant tinklo sudėtingumą evoliucijos proceso metu.

Mano susidomėjimą DNT ir genetinais algoritmais sužadino „YouTube“ vaizdo įrašas pavadinimu „MarI/O - Machine Learning for Video Games“, kurį sukūrė vartotojas vardu SethBling (2015). Šiame vaizdo įraše DNT genetinės evoliucijos būdu apmokomas įveikti pirmąjį „Super Mario World“ lygį. Tai yra įrodymas, kad genetinės evoliucijos taikymo sritis yra labai plati. Naudoto algoritmo ypatumus aptarsiu šiame darbe.

2 Dirbtiniai neuronų tinklai

Neuronų tinklo struktūrą gali apibūdinti orientuoto beciklio svorinio grafo analogija, tinklas susideda iš viršūnių, N (angl. Neuron)ir lankų L (angl. Link). DNT tinklo struktūra:

- Nekintantis kiekis įvedimo N
- Nekintantis kiekis išvedimo N
- Dinaminis arba nekintantis kiekis paslėptų N
- Kiekvienas N turi aktyvacijos funkciją ir (nebūtina) slenksčio reikšmę
- Dinaminis arba nekintantis kiekis L, bei apjungia N ir taip sudaro DNT struktūrą
- Kiekvienas L turi svorio reikšmę



Rezultatas yra pasiekiamas sudauginant kievieno N įeinančius svorius ir perleidžiant juos pre aktyvacijos funkciją. Kadangi DNT yra beciklio grafo struktūros, per kiekvieną N yra pereinama topologine tvarka.

DNT yra *mokomas* t. y. koreguojami L svoriai, kurie duoda vis kitokį rezultatą. Kartais, problema yra tokia, kada žinome norimą rezultatą, tada galima naudoti *paspartintą mokymą (*angl. Reenforced learning), tačiau dažnai galime tik palyginti kuris iš rezultatų yra geresnis. Taip pat labai dažnai rezultatas priklauso ne tik nuo svorių, tačiau ir nuo pačios tinklo struktūros. 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.

3 Genetiniai algoritmai

Genetiniai algoritmai (toliau GA) yra evoliucinių algoritmų (toliau EA) poklasis, todėl prasminga aptarti kas būdinga EA klasei.

EA simuliuoja organizų evoliuciją ir panašiai kaip ir gamtoje, išlieka stipriausi. EA veikimo principas:

1. Sugeneruoti individualių organizmų aibę (generaciją)
2. Įvertinti jų pajėgumą (angl. Fitness)
3. Mažiausiai pajėgūs organizmai panaikinami
4. Generuoti naują aibę atsižvelgiant į pajėgiausius organizmus
5. Mutuoti (keisti atsitiktinai) naujus organizmus
6. Kartoti kol gaunamas tenkinamas rezultatas

Generuojanti naują organizmų aibę gali yra naudojami įvairūs metodai kaip kryžminimas (angl. Crossover) ar paprasčiausias klonavimas. GA naudoja organizmų kryžniminą ir atsitiktines mutacijas.

4 Neuro-Evoliucija (NE)

„Neuro-Evoliucija (toliau NE), tai DNT evoliucija naudojant GA, kuri parodė gerus rezultatus sudėtingose mokymo su mokytoju (angl. Reinforced learning) problemose“. [GFM1] [GFM2] [MM]

4.1 Neuro-Evoliucijos Aktualumas

“Pilnai jungus DNT iš principo gali apytiksliai apskaičiuoti visas tolydžias funkcijas“ [GC,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 funkcionalumą.“ [Ken1, 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. [GRU]. Tačiau vėliau buvo įrodyta, kad tokios sudėtingos tinklo struktūros nereikėjo. Statinės topologijos metodas pavadinimu „Enforced Subpopulations“ (ESP) [GM3] rado sprendimą 5 kartus greičiau, tiesiog pradėdamas su atsitiktiniu paslėptų neuronų skaičiumi kada progresas užstrigdavo.

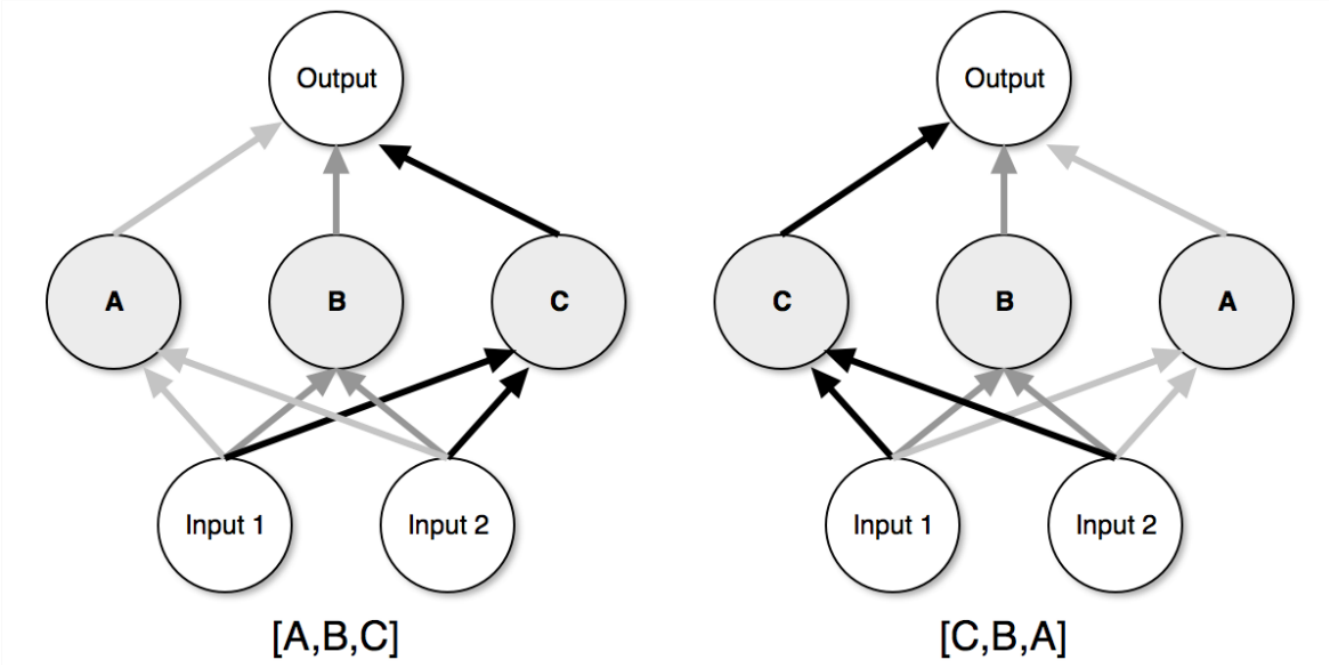
Šiame darbe pateiksiu priešingą išvadą, kad NE gali būti net labai efektyvus metodas, jeigu yra atliktas teisingai.

5 NeuroEvolution of Augmenting Topologies (NEAT)

Kintančių topologijų neuro-evoliucija yra NE algoritmas, sukurtas Kenneth O. Stanley ir Risto Miikkulainen.[Ken1] 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 lankų.

5.1 Konkuriuojantys sprendimai (Competing Conventions)

NE buvo vengiama kryžminimo, kas yra vienas iš pagrindinių GA principų, todėl kad kryžminimas dažnai yra nenaudingas, nes DNT praranda funkcionalumą. Dėl šios priežasties NE metoduose kryžminimo yra visiškai atsisakyta. Tai vadinama *Evoliuciniu Programavimu* [YAL]



Kaip matome, turime 2 funkcionaliai vienodus tinklus, tačiau jų neuronų išsidėstymas daro juos skirtingais kryžminimo procese. Kryžminimo metu (B yra bendras požymis) [A, B, C]×[C, B, A] įmanomi gaiti vaikai yra [A, B, A] ir [C, B, C]; todėl prarandama informacija. Vienas šios problemos iš sprendimų yra atlikti topologinę analizę, tačiau ji yra sudėtinga ir lėta, todėl reikia DNT kodavimo bei būdo juos efektyviai kryžminti.

5.2 Genetinis Kodavimas

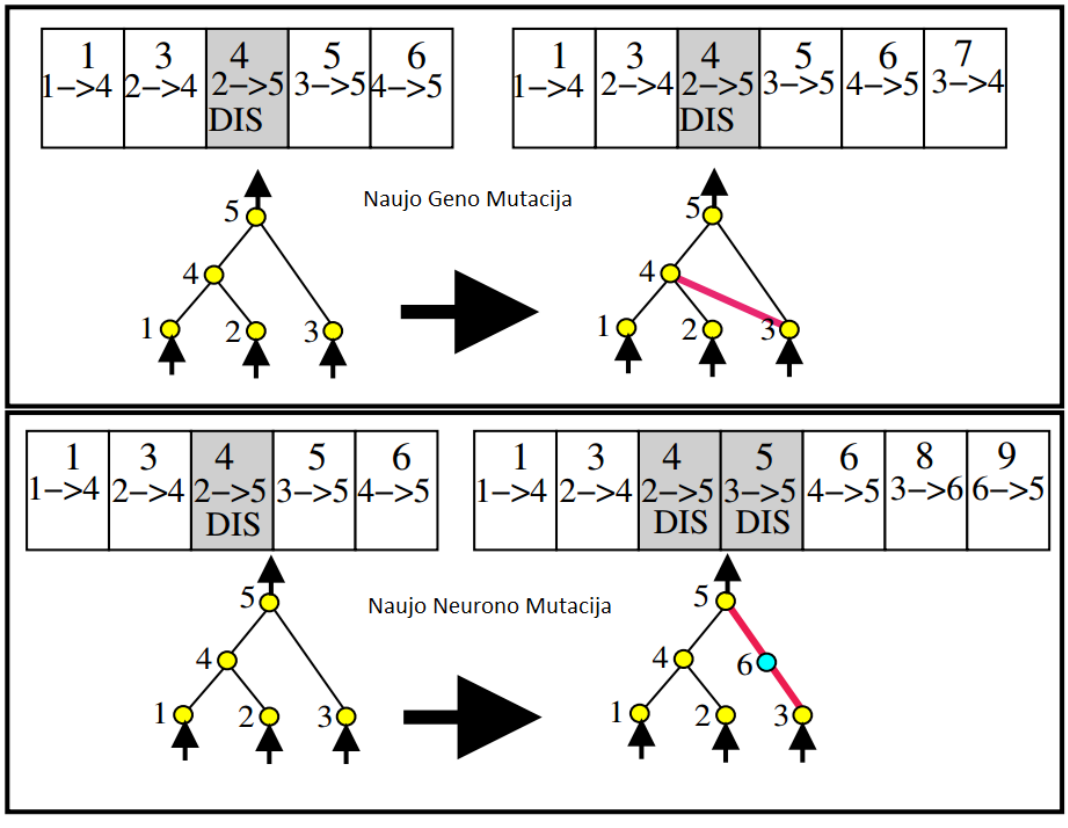
NEAT algoritmas *genq* apibrėžia kaip DNT lanką, kuris turi svorį, įėjimo ir išėjimo neuroną, gali būti išjungtas bei turi *inovacijos žymę*, kuri gali būti paprastas globalus skaitliukas. *Inovacijos žyme* yra *homologinis* rodiklis tarp tėvų ir vaikų, i.e. paveldėti bruožai (šiuo atveju DNT struktūra). Ši žymė yra naudojama kryžminimo procese, nes tada galima atskirti sutanpančius genus (svoriai gali skirtis) ir informacija yra išsaugojama. Visų genų sąrašas yra g*enotipas*, o grafinis jo vaizdas yra f*enotipas*.

5.3 Mutacijos

NEAT algoritme naudojamos mutacijos:

- Keičiančios lanko svorį
- Keičiančios DNT topologiją
  - Pridėti naują geną tarp 2 jau esamų *neuronų*
  - Geno vietoje įterpti naują neuroną, t. y. išskirti geną į 2 naujus ir apjungti su nauju neuronu, o seną geną išjungti
  - Įjungti/Išjungti jau esamą geną

Kieviena naujo geno sukūrimo mutacija turi savo inovacijos žymę (viršuje). Taip pat vienodos genų mutacijos skirtinguose genotipuose įgauna vienodą inovacijos žymę.



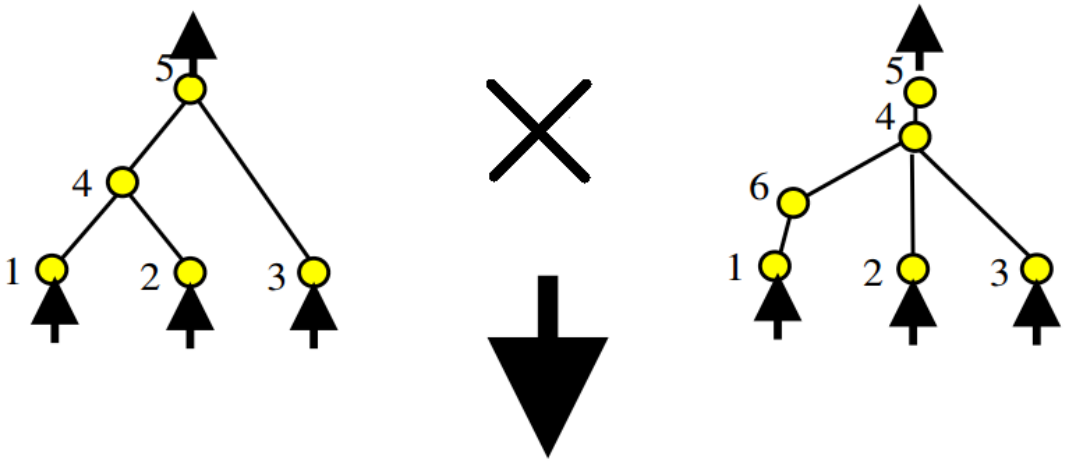
5.4 Kryžminimas (Crossover)

Tėvas 1

1	2	4	5	6
1 → 4	2 → 4	2 → 5	3 → 5	4 → 5

Tėvas 2

1	2	3	4	6	7	8
1 → 4	2 → 4	3 → 4	2 → 5	4 → 5	1 → 6	6 → 4



Tėvas 1

1	2
1 → 4	2 → 4

4	5	6
2 → 5	3 → 5	4 → 5

neb. neb. atl. atl.

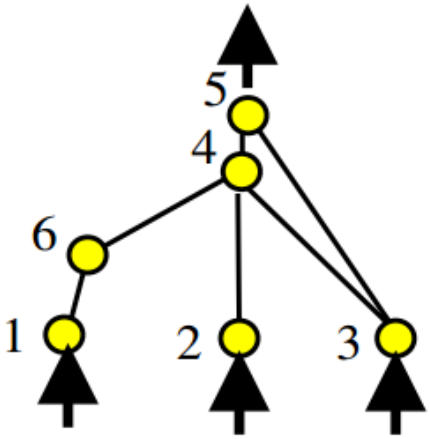
Tėvas 2

1	2	3	4
1 → 4	2 → 4	3 → 4	2 → 5

6	7	8
4 → 5	1 → 6	6 → 4

Vaikas

1	2	3	4	5	6	7	8
1 → 4	2 → 4	3 → 4	2 → 5	3 → 5	4 → 5	1 → 6	6 → 4



Nors ir tėvai vizualiai skiriasi, inovacijos žymės (geno viršuje) parodo sutampančius genus, kuriuos išlyginus galima vykdyti kryžminimą.

- 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

5.5 Konkuravimo išskirstymas naudojant rases (*speciation)*

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

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

E - atliekamų (excess) genų skaičius
D - nebendrų (disjoint) genų skaičius
W - sutampančių genų svorių skirtumų vidurkis
N - didesniojo genotipo genų skaičius
c_1, c_2 ir c_3 nustato šių daugiklių svarbą

„Kuo mažiau yra atliekamų ir nebendrų genų genotipuose, tuo jie yra artimesni evoliucinės istorijos prasme“ [Ken1, 112]

Kiekvienai rasei atstovauja atsitiktinis genotipas iš ankstesnės generacijos, kuris yra naudojamas naujos generacijos genotipams sugrupuoti. Jeigu naujas genotipas netinka visoms esamoms resėms, 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 (*Explicit fitness sharing)* [DEJ], kada pajėgios rasės genotipai turi dalintis savo pajėgumo su visa populiacija.

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

Naujas pasvertas genotipo *i* pajėgumas yra apskaičiuojamas pagal genotipų panašumo funkciją su visais kitais populiacijoje esančiais genotipais *j*. Pajėgumo paskirstymo funkcija *sh* gauna reikšmę 0, jeigu *d* funkcijos reikšmė viršija nustatytą ribą, kuri gali kisti nuo generacijos numerio, priešingu atveju gauna 1. [SP]

Mažiausiai pajėgūs rasės genotipai yra pašalinami ir generuojama nauja populiacija proporcingai kryžminant kiekvienos rasės genotipus.

Skirstymas į rases apsaugo genotipų topoligines mutacijas nuo ankstyvo eliminavimo ir suteikia šansą suoptimizuoti naujai įgytus genus.Taip pat skirstymas į rases toleruoja dažnas mutacijas, kas spartina optimizavimą.

„Tokios sistemos galutinis tikslas yra rasti sprendimą minimizuojant paieškos erdvės dimensiją ir kaip įmanoma efektyviau“ [Ken, 112].

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.

Genotipo ciklas:

1. Pasirinkti sekantį genotipą *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 genotipai buvo priskirti, baigti

5.6 NEAT variantų palyginimas

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

Įvairaus pobudžio NEAT variantai buvo išbandyti dvigubo strypo balansavimo užduotyje [Ken2, 52]. Eksperimentinio tyrimo netu populiacijos dydis buvo 150. Jeigu sprendinio nepavyko rasti per 1000 generacijų yra bandoma iš naujo. Rezultatai gaunami iš 20 kiekvieno varianto atvejų vidurkio.

5.7 NEAT struktūra

NEAT algoritmo pradinė DNT struktūra yra 2 sluoksnių, kada kiekvienas įėjimo neuronas yra sujungtas su išėjimo neuronu. 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 vistiek bus papildomas optimizavimo darbas. Taigi, pradedant su minimalia DNT struktūra, evoliucijos metu ji sudėtingėja ir kiekvienas topologinis pokytis yra pateisintas.

DNT struktūroje gali būti naudojamos įvairios aktivacijos funkcijos (sigmoidinė, Gauso, slenkstinė, tiesinė) bei jų modifikacijos.

5.8 Galimi pokyčiai algoritme

Gali būti atvejis, kada NEAT algoritmas išoptimizuos tam tikrus genus, kurių iš tikrūjų reikia korektiškam sprendimui pasiekti, nors tokia situacija yra reta. Galimas problemos sprendimas yra vietoje genų išjungimo pakeisti jų svorį į 0. Tokiu būdu yra suteikiamas šansas atstatyti reikšmę vėliau, kada (jeigu) jos prireiks.

NEAT algoritmas visiškai atsiriboja nuo pačių neuronų mutavimo. Pvz, galima suteikti neuronams *slenkščio (bias)* reikšmę, kuri gali kisti (mutuoti) ir kurios evoliucijos raidos nereikia sekti su inovacijos žymėmis, tačiau nežinia ar tai paspartins algoritmą.

5.9 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“ [WH]. Taigi, natūralu NEAT peiškoje naudoti lygiagrečius skaičiavimus.

Paprasčiausiai išlygiagretinami atskirų genotipų pajėgumo skaičiavimai, nes jų operacijos yra visiškai nepriklausomos. Sunkiau tampa išlygiagretinti naujos generacijos kūrimą. Prieš generavimą reikia pašalinti mažiausiai pajėgius genotipus, o tai galima padaryti tik apskaičiavus visų genotipų pajėgumus, nes reikia vidutinio pajėgumo įverčio. Paprasto barjero mechanizmo tam pilnai pakanka. Naujų genotipų pajėgumų įverčiai gali būti pradėti skaičiuoti dar nebaigus generuoti naujos genotipų aibės, todėl tai yra gerai spartinamas algoritmas.

5.10 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 navigavimas grėsmingoje aplinkoje link maisto, optimalios vėjo jėgainės formos nustatymas ar dirbtinio intelekto mokymas įvairiuose žaidimuose. Problemų iškyla kai simuliacijos turi nedeternimistinių kintamųjų. Tada genotipo pajėgumas gali ženkliai suprastėti, jeigu simuliacijos eigoje pasitaikys nepalankių sąlygų. Tokią problemą galima lengvai išspręsti pakartotinai vykdant simuliaciją ir genotipo pajėgumą įvertinant vidutiniškai. Toks sprendimas įmanomas tik jeigu simuliacijos įvykdymo kaina yra pigi.

Vienas iš NEAT pritaikymų yra 2 ar daugiau žaidėjų žaidime. Tokiu atveju nebūtina tūrėti jau apmokytą ar žmogaus sukurtą dirbtinio intelekto priešininką. NEAT algoritmas gali žaisti prieš save patį ir tobulėti, vis atrasdamas naujų strategijų. Tada evoliuciją galima vykdyti turnyro principu, t. y. geriausiai pasirodęs genotipas savo rasėje konkuriuoja su kitų rasių nugalėtojais ir tada nugalėtojai nukreipia visos populiacijos evoliucijos kryptį, nes nugalėtoji rasė gauna daugiausiai išteklių daugintis. Tokiu atveju nereikia naudoti pajėgumo funkcijos, nes pajėgumą nurodo turnyre užimta vieta.

6 Genotipo pajėgumo įvertinimas

Tikriausiai pats sunkiausias ir svarbiausias kiekvieno GA aspektas yra tinkamos pajėgumo įverčio funkcijos pasirinkimas. Labai dažnai prasta pajėgumo funkcija gali nukreipti evoliuciją visiškai klaidinga linkme.

Pvz.: Užduotis yra rasti išėjimą labirinte. Pajėgumas matuojamas pagal tiesioginį atstumą nuo išėjimo. Jeigu yra aklavietė 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.

Taip pat reikia atsargiai skirti taškus už įgytą mažą pranašumą, nes algoritmas gali įklimpti į *lokalų ekstremumą*.

Pvz.: Primityvių organizmų genčių simuliacija. Taškai yra skiriami už organizmo išgyvenimo trukmę, išteklių išnaudojimą ir daug kitokių faktorių (pvz. nužudytų organizmų kiekį). Užduoties tikslas yra ne tik išgyventi, o ir dominuoti tarp kitų genčių. Kadangi už išgyvenimą pelnyti taškus yra žymiai lengviau negu už kitų genčių žudimą (tam gali prireikti reikia sudėtingos struktūros DNT), organizmai optimizuos linj išgyvenimo, nes jie tik tai temoka ir pagal pajėgumo kriterijų progresas nesustoja. Galutinis tokios simuliacijos evoliucijos rezultatas bus daugybė skirtingų genčių, kurios vengia kontakto su kitomis gentimis ir ieško saugių, lengvai prieinamų išteklių. Jeigu buvo siekiama gančių dominacijos, tai tokia pajėgumo funkcija nėra tinkama.

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

7 Išvados

- Neuro-Evoliucija yra galingas įrankis įvairaus mąsto ir spektro problemoms spręsti
- NE metu DNT topologijos keitimas yra naudingas, jeigu atliekamas teisingai
- NEAT algrotimo pagrindinė idėja, kuri leiždia efektyviai vykdyti NE, yra inovacijos žymės
- NE organizmų išskaidymas į rases yra efektyvus būdas paspartinti sprendimo radimą
- NE yra lengvai spartinimama multiprocesinėje aplinkoje
- Sunkiausias ir svarbiausias NE taikymo aspektas yra tinkamos pajėgumo funkcijos parinkimas
- Specifinėse simuliacijose NE gali būti taikoma be pajėgumo funkcijos

8 Literatūra

[Ken1] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99, 127, 2002.

[Ken2] Kenneth O. Stanley. *Efficient Evoliution of Neural Networks through Complexification*. PhD thesis, The University of Texas, 2004

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

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

[GM1] Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342.

[GM2] 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.

[GM3] Gomez, F. and Miikkulainen, R. (2002). *Learning robust nonlinear control with neuroevolution*.
Technical Report AI02-292, Department of Computer Sciences, The University of Texas at Austin, Austin, Texas

[GRU] 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.

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

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

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

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