**LATVIJAS UNIVERSITĀTE**

JĀNIS ZUTERS

# NEIRONU TĪKLI STUNDU SARAKSTA SASTĀDĪŠANAS MODELĪ

Promocijas darbs

Rīga – 2007

LATVIJAS UNIVERSITĀTE

**Fizikas un matemātikas fakultāte**

JĀNIS ZUTERS

# NEIRONU TĪKLI STUNDU SARAKSTA SASTĀDĪŠANAS MODELĪ

Promocijas darbs

Datorzinātņu doktora (Dr. sc. comp.) zinātniskā grāda iegūšanai

Nozare: datorzinātnes

Apakšnozare: datu apstrādes sistēmas un tīkli

Zinātniskais vadītājs:

Profesors, Dr. sc. comp.

JĀNIS BIČEVSKIS

Rīga – 2007

*Promocijas darbs. Latvijas Universitāte, 2007*

# Saturs

# Ievads

**Stundu saraksta sastādīšanas problēma** (*timetabling problem*) ir plaši sastopams grafiku (sarakstu) sastādīšanas problēmu veids. Viens no izplatītākajiem stundu saraksta sastādīšanas problēmas paveidiem ir **skolu stundu saraksta sastādīšana** (*school timetabling*). Šāda veida problēma var tikt risināta, izmantojot ģenētiskos algoritmus, tomēr ģenētiskie algoritmi paši par sevi veido tikai karkasu, kurā tiek ievietoti dažādi algoritmi, kas lielā daļā gadījumu ir specifiski risināmajai problēmai. Viena no specifiskām apakšproblēmām ģenētisko algoritmu pielietošanas gadījumā ir objektu, t.i., stundu sarakstu novērtēšana to derīguma noskaidrošanai. Stundu sarakstu gadījumā tas nav vienkāršs uzdevums, jo stundu saraksti ir ļoti sarežģīti objekti, un to sastādīšanu lielā mērā ietekmē cilvēciskais faktors. Tas ir pamatā autora idejai izmantot neironu tīklus kā papildus līdzekli stundu sarakstu novērtēšanai.

**Promocijas darba mērķis** ir izstrādāt metodes, kas nodrošinātu neironu tīkla kā skaitļošanas sistēmas iesaistīšanu uz ģenētiskajiem algoritmiem balstītā stundu saraksta sastādīšanas modelī ar nolūku uzlabot stundu saraksta sastādīšanas procesā iegūto stundu sarakstu kvalitāti.

Tādu metožu izstrādei ir veicams šāds galvenais **uzdevums**: izpētīt neironu tīklus kā derīguma funkcijas (*fitness function*) komponenti uz ģenētiskajiem algoritmiem (*genetic algorithms*) bāzētā sistēmā, kas paredzēta skolu stundu sarakstu sastādīšanai (*school timetabling*).

Darbs atrodas vairāku datorzinātņu nozaru saskares punktā, ietverot tādus apgabalus kā mākslīgais intelekts (t.sk. neironu tīkli, ģenētiskie algoritmi) un **ierobežojumu izpildīšanas problēmu** (*constraint satisfaction problems*) risināšana.

**Neironu tīkli** (*neural networks*) ir skaitļošanas paradigma, kas savā attīstībā ietekmējusies no bioloģiskām nervu sistēmām, t.sk. no cilvēka smadzenēm [haykin99]. Daudzos avotos tie vairāk vai mazāk alegoriski pat ir saukti par “cilvēka smadzeņu modeli”. Tomēr pēc risināmo problēmu apgabala neironu tīkli jau sen netiek uzskatīti par alternatīvu cilvēka smadzenēm un universālu līdzekli “visam” – tie vairumā gadījumu tiek veidoti ļoti konkrētu un šauri specifisku problēmu risināšanai.

Atkāpšanās no skaļiem saukļiem un cēliem mērķiem raksturīga ne tikai neironu tīkliem, bet vai visām mākslīgā intelekta apakšnozarēm. Attieksme pret šo sfēru kļuvusi daudz praktiskāka, un definīcijas piezemētākas. **Mākslīgais intelekts** (*artificial intelligence*) ir datorzinātņu nozare, kas nodarbojas ar saprātīgas uzvedības automatizāciju [luger02]. Mākslīgā intelekta pielietošanai ir raksturīga liela datu apjoma izmantošana gala rezultāta iegūšanai – piemēram, ekspertu sistēmas, ģenētiskie algoritmi, neironu tīkli. No vienas puses, tas ir, pateicoties datortehnikas attīstībai, kas sniedz šādas iespējas, no

otras puses, ir skaidrs, ka ar "tīrajiem" algoritmiem risināmo problēmu klāsts un risinājumu kvalitāte ir stipri ierobežota.

Liela datu apjoma izmantošana problēmu risināšanai (un neironu tīkli šeit ir ļoti tipisks piemērs) var dot papildus iespējas veikt darbības ātrāk, drošāk un ar augstāku uzticamības pakāpi. Tai pat laikā liela datu apjoma izmantošana aprēķinos ir saistīta gan ar praktiskām, gan teorētiskām grūtībām, kas lielākoties izpaužas kā nepieciešamība pēc papildus laika resursiem:

- aprēķiniem nepieciešamo datu savākšana un sagatavošana noteiktas sistēmas darbības nodrošināšanai var prasīt apjomīgu noteiktas nozares ekspertu, kā arī datu apstrādes speciālistu darbu,
- ir nepieciešams daudz apjomīgāks eksperimentāls darbs un testēšana šādas sistēmas īpašību novērtēšanai vai sistēmas novērtēšanai kopumā.

Promocijas darba teorētiskā daļa ir veltīta neironu tīklu uzbūves, darbināšanas un pielietošanas principiālo jautājumu izpētei. Darba praktiskajā un eksperimentālajā daļā ir tikusi ietverta aptauju veikšana, datu savākšana, specializētas oriģinālas datorsistēmas veidošana, datoreksperimentu veikšana un to rezultātu apkopošana un analīze.

Promocijas darba galvenais rezultāts ir autora piedāvātais **ģenētisko algoritmu un neironu tīklu hibrīdais modelis** stundu saraksta sastādīšanas problēmas risināšanai.

Darba pamatdaļa strukturēta 5 nodaļās, sākot ar neironu tīklu un citu tehnoloģiju aprakstu, turpinot ar metodēm neironu tīklu darbības optimizācijā dažādos aspektos un noslēdzot ar piedāvāto neironu tīklu un ģenētisko algoritmu hibrīdo modeli skolu stundu saraksta sastādīšanai.

**1. nodaļa** dod ieskatu neironu tīklu skaitļošanas paradigmā. Tieši ar neironu tīkliem saistītas autora piedāvātās un darbā aprakstītās novitātes. Sākumā dots īss ieskats neironu tīklu vēsturē un uzbūves un darbības vispārīgajos principos, akcentējot apmācības algoritmu kā neironu tīklu paradigmas ļoti būtisku sastāvdaļu. Tālāk aprakstīti 3 pazīstami neironu tīklu modeļi – (a) vairākslāņu perceptrons ar kļūdu atgriezeniskās izplatīšanās (*backpropagation*) apmācības algoritmu, (b) radiālo bāzes funkciju tīkls un (c) Kohonena tīkls. Pirmie divi modeļi (vairākslāņu perceptrons un radiālo bāzes funkciju tīkls) ir funkcionāli ļoti līdzīgi un tiešā veidā izmantoti autora veiktajos eksperimentos, un uz to bāzes būvēti autora piedāvātie jauninājumi neironu darbināšanā un apmācībā (nodaļas 4.3.2, 4.4.3) un neironu tīklu iesaistīšanā stundu sarakstu novērtēšanā (nodaļa 5). Savukārt Kohonena tīkla uzbūves un darbības idejas ir kalpojušas par iedvesmu autora piedāvātajā vairākslāņu perceptrona paplašinājuma izstrādē (nodaļa 4.3.2).

**2. nodaļa** stāsta par ģenētiskajiem algoritmiem – uz dabisko paaudžu nomaiņu un evolūciju balstītu optimizācijas shēmu, uz kuras bāzes veidota autora piedāvātais stundu

saraksta sastādīšanas modelis. Šajā nodaļā ģenētiskie algoritmi aprakstīti gan kā tādi, gan zināmie veidi ģenētisko algoritmu un neironu tīklu kopdarbībai, kas izpaužas kā ģenētisko algoritmu izmantošana neironu tīklu izstrādē.

**3. nodaļa** apraksta stundu saraksta sastādīšanas problēmu kā ierobežojumu izpildīšanas problēmu (*constraint satisfaction problem*), stundu sarakstu sastādīšanas specifiku skolās. Bez tam aprakstītas vairākas zināmas metodes un paņēmieni ģenētisko algoritmu izmantošanai stundu sarakstu sastādīšanai, kas ietver gan risinājumu reprezentāciju apstrādei ģenētiskajos algoritmos, gan dažādas ģenētisko algoritmu specifiskas konstrukcijas kā, piemēram, ģenētiskie operatori un derīguma funkcija.

**4. nodaļa** apraksta neironu tīkla īpašības un to uzlabošanu. Šī ir pirmā no divām nodaļām, kurā aprakstītas autora piedāvātās novitātes. Pie neironu tīklu īpašībām īpaši izcelta neironu tīklu vispārināšanas spēja, ietverot arī metožu klāstu tās palielināšanai. Pie metodēm dažādu neironu tīklu īpašību uzlabošanā izceltas divas grupas: (a) heiristikas un (b) ar nejaušību izmantošanu saistītas metodes. Abu šo grupu ietvaros autors piedāvā un apraksta divas dažādas metodes, kas saistītas ar neironu tīklu uzbūves un darbības paplašināšanu. Vispirms nodaļā 4.3.2 tiek demonstrēts **uz slāņu topoloģiju balstīts vairākslāņu perceptrona paplašinājums**, kur neironu tīklā ar kontrolētu (*supervised*) apmācību tiek inkorporētas idejas, kas sakņojas pašorganizējošos tīklu (konkrēti, Kohonena tīkla) uzbūvē un tajā ietvertajā viena slāņa neironu kooperācijā. Kaut arī šajā nodaļā aprakstītais autora piedāvājums nav tieši saistīts ar šī darba galamērķi – stundu saraksta sastādīšanas sistēmas izveidošanu, tomēr tas demonstrē autora darbu neironu tīklu izpētē, un no šāda viedokļa arī šis ir uzskatāms par blakus rezultātu ceļā uz galamērķi. Pēc tam, nodaļā 4.4.3, autors lieto heiristisku piegājienu nejaušības izmantošanā, piedāvājot **pēc nejaušības principa ģenerētu paraugu izmantošanu neironu tīklu apmācībā** gadījumos, kad apmācībai pieejams nepilnīgs un vienpusīgs paraugu kopums. Galvenā ideja ir apmācīt neironu tīklu noteiktās proporcijās gan ar esošajiem paraugiem, gan ar pēc nejaušības principa ģenerētajiem, turklāt tādā veidā, ka vēlamā vērtība atbildei uz paraugu (kas tiek izmantota apmācības algoritmā) esošajiem paraugiem tiek noteikta maksimālā, bet ģenerētajiem – minimālā. Uz to balstās arī neironu tīklu komponentes uzbūve, kas ietverta nākošajā nodaļā aprakstītajā stundu sarakstu sastādīšanas modelī.

**5. nodaļa** ir centrālā un galvenā šī darba nodaļa, kas apraksta autora piedāvāto **neironu tīklu un ģenētisko algoritmu hibrīdo modeli skolu stundu sarakstu sastādīšanā**. Aprakstītais modelis ir vairāku nozaru aktivitāšu un dažādu tehniku apvienojums, kas tika izmantotas ne tikai paša modeļa izveidošanā, bet arī tā novērtēšanā. Nodaļā aprakstītā piedāvātā modeļa uzbūve un darbība visu 3 tā konstruktīvo bloku ietvaros: (a) stundu saraksta sastādīšanas bloks, (b) neironu tīklu bloks un (c) ģenētisko algoritmu bloks. Galvenā autora piedāvātā novitāte ir neironu tīklu iesaistīšanas veids modelī, nosakot neironu tīklam stundu sarakstu vērtētāja lomu ģenētiskā algoritma derīguma

funkcijas ietvaros. Tālāk tiek piedāvāta speciāli izstrādāta metodika šī modeļa novērtēšanai, precīzāk, neironu tīklu pielietošanas efekta novērtēšanā šī modeļa ietvaros. Šī metodika arī ir svarīga, jo modeļa novērtēšana ne tuvu nav triviāla un tā pielietojuma efekts nav acīmredzams. Metodikas pamatā ir tieši definēto kritēriju (t.i., iepriekš noteikto kritēriju stundu saraksta novērtēšanai) izmešana no derīguma funkcijas vai aizvietošana ar neironu tīklu, beigās salīdzinot ar attiecīgajām sistēmām iegūtos stundu sarakstus, par mērauklu izmantojot pilnu tieši definēto kritēriju komplektu (t.i. derīguma funkciju bez modifikācijām). Beigās doti eksperimentu rezultāti, analīze un secinājumi, kas parāda, ka piedāvātais modelis ir izmantojams stundu saraksta sastādīšanas problēmas risināšanā.

Promocijas darbs tapis uz autora apmēram četrus gadus veikto pētījumu pamata neironu tīklu nozarē, kā arī stundu saraksta sastādīšanā, izmantojot ģenētiskos algoritmus. Eksperimentu veikšanai autors ir izstrādājis apjomīgu oriģinālu programmatūru, kuras pirmkods satur vairākus desmitus tūkstošu koda rindiņu valodā C++. Veikto pētījumu rezultāti atspoguļoti piecās autora publikācijās, kas veido promocijas darba galveno līniju:

- Darbā [zuters05a] tiek konceptuāli piedāvāts neironu tīklu un ģenētisko algoritmu hibrīdais modelis stundu sarakstu sastādīšanai, parādot modeļa tapšanas motivāciju un iespējamo struktūru un darbību. Modelis balstās uz neironu tīklu iesaistīšanu stundu sarakstu vērtēšanā ģenētisko algoritmu derīguma funkcijas ietvaros blakus tieši definētajiem stundu sarakstu vērtēšanas kritērijiem. Viens no šajā darbā izvirzītajiem uzdevumiem ir izveidot stundu sarakstus vērtēt spējīgu neironu tīklu.
- Veicot eksperimentus ar neironu tīkliem, autors nonāca pie interesanta blakus rezultāta – neironu tīkla vispārināšanas spējas uzlabošanās, izmantojot uz ieejas slāņa topoloģiju balstītu ieejas signālu kooperāciju, kas aprakstīts [zuters05b]. Piedāvātā pieeja apvieno vairākslāņu perceptrona modeli ar neironu slāņa topoloģijas izmantošanu, kas aizgūta no Kohonena tīkla apmācības algoritma.
- Darbā [zuters06a] autors prezentē apmācības procesa organizācijas metodi, kas paredzēta vērtējoša neironu tīkla ieguvei, izmantojot nepilnīgu un vienpusīgu apmācības kopu, kurā ir tikai paraugi ar pozitīvu vērtējumu. Šādas metodes izstrāde balstījās uz iepriekš [zuters05a] izvirzīto uzdevumu stundu sarakstus vērtējoša neironu tīkla izstrādei, kas tiktu apmācīts ar gataviem stundu sarakstiem. Šajā darbā tika parādīts metodes pielietojums ar roku rakstītu simbolu atpazīšanā.
- Darbs [zuters06b] ir tiešs darba [zuters05a] turpinājums, kas precizē neironu tīklu un ģenētisko algoritmu hibrīdo modeli stundu sarakstu sastādīšanai skolās un piedāvā metodiku neironu tīklu izmantošanas efekta noskaidrošanai. Metodikas pamatā ir tieši definēto kritēriju aizvietošana ar neironu tīkliem. To veicot, tiek salīdzināti

ģenētisko algoritmu ģenerētie stundu saraksti, kas iegūti attiecīgi ar un bez neironu tīkla iesaistīšanās stundu sarakstu vērtēšanā.

- [zuters07] ir darbu [zuters06a] un [zuters06b] turpinājums un kopsavilkums, precizējot [zuters06b] aprakstīto modeli, parādot [zuters06a] aprakstītas neironu tīklu apmācības metodes pielietošanas iespēju, veicot apmācību ar stundu sarakstiem. Bez tam darbs sniedz eksperimentu gala rezultātus, parādot neironu tīklu izmantošanas efektu derīguma funkcijas ietvaros uz ģenētiskajiem algoritmiem bāzētā stundu sarakstu sastādīšanas sistēmā.

Promocijas darbs ir autora iepriekš uzskaitītajās publikācijās aprakstītā pētnieciskā darba loģisks nobeigums, veidojot pabeigtu darbu kopumā.

Darbā aprakstītais **autora ieguldījums** neironu tīklu attīstībā un stundu saraksta sastādīšanas problēmas risināšanā:

1. Piedāvāta neironu tīklu **apmācības metode**, kas izmanto nejauši ģenerētus apmācības paraugus, lai aizvietotu iztrūkstošus problēmu reprezentējošos paraugus.
2. Izstrādāts ģenētisko algoritmu un neironu tīklu **hibrīdais modelis** stundu saraksta sastādīšanas problēmas risināšanai, kura novitāte balstās uz diviem aspektiem:
    a. neironu tīkla iekļaušana ģenētiskā algoritma derīguma funkcijas ietvaros,
    b. neironu tīkla apmācība, izmantojot tikai pozitīvos stundu sarakstus, izmantojot iepriekš piedāvāto apmācības metodi.
3. Piedāvāts neironu tīkla (konkrēti, vairākslāņu perceptrona) **apmācības un darbināšanas algoritma paplašinājums**, papildus izmantojot informāciju par ieejas slāņa topoloģiju, kas nosaka ieejas signālu kooperāciju.

Promocijas darba apraksta objekts ir zinātnisks eksperiments, un tā rezultāts ir skaitļošanas modelis. Izstrādātā modeļa praktisks pielietojums stundu saraksta sastādīšanai būtu nākošais solis autora piedāvāto ideju un metožu attīstīšanā.

# 1. Neironu tīkli – skaitļošanas paradigma

Šajā nodaļā dots vispārīgs ieskats neironu tīklu tehnoloģijā kā līdzeklī risināt noteiktus uzdevumus. Dots pārskats pār neironu tīklu izcelšanos un attīstību, galvenajiem darbības principiem un iespējamo pielietojumu dažādu problēmu risināšanai.

## *1.1. Neironu tīkli, to rašanās motivācija un attīstība*

### 1.1.1. Neironu tīkls – cilvēka smadzeņu modelis

**Neironu tīkli** (*neural networks*) vai, precīzāk, **mākslīgie neironu tīkli** (*artificial neural networks*) ir tehnoloģija, kas sakņojas daudzās sfērās, tādās kā, piemēram, neirofizioloģija, matemātika, statistika, fizika, datorzinātne un inženierzinātnes. Neironu tīkli ir raduši pielietojumu tik atšķirīgās jomās, kā modelēšana, laika rindu analīze, paraugu atpazīšana, signālu apstrāde un kontrole, ļoti lielā mērā pateicoties savai spējai mācīties no ieejas datiem [haykin99].

Jau no pašiem pirmsākumiem neironu tīklu izpēti ir motivējusi atziņa, ka cilvēka smadzenes rēķina pilnīgi savādākā veidā, nekā to dara parastie ciparu datori. Smadzenes ir ļoti kompleksa, nelineāra un paralēla informācijas apstrādes sistēma. Uzreiz pēc cilvēka dzimšanas smadzenēm ir dota izcila struktūra un līdz ar to spēja veidot pašām savus noteikumus, ko mēs saucam par pieredzi. Protams, cilvēkam pieredze nāk ar laiku un tās iegūšana notiek visu mūžu, tomēr visstraujāk tas notiek pirmajos divos cilvēka dzīve gados. [haykin99]

Vispārīgā nozīmē neironu tīkls ir mašīna, kas ir veidota, lai modelētu smadzeņu darbību noteikta uzdevuma sasniegšanai vai problēmas risināšanai. [haykin99] Šādā nozīmē varētu teikt, ka neironu tīkls ir cilvēka smadzeņu modelis. Tomēr, salīdzinot ar cilvēka smadzenēm, mākslīgie neironu tīkli ir uzskatāmi tikai par ļoti vienkāršotiem modeļiem. Ir tikai divas īpašības (principi), ko ikviens neironu tīkls ir pārņēmis no smadzenēm:

1. **Apmācība**. Neironu tīkls zināšanas iegūst apmācības ceļā.
2. **Konekcionisms.** Neironu tīkls sastāv no liela daudzuma neironu, un to savstarpējās saites nosaka tīklā glabātās zināšanas.

Katrs neirons ir relatīvi vienkārša skaitļošanas vienība, tomēr liels daudzums neironu, kas pēc noteiktiem likumiem savienoti viens ar otru, kopā var paveikt ļoti sarežģītus uzdevumus. Bioloģiskā neirona takts frekvence ir aptuveni sešas kārtas zemāka nekā mūsdienu procesoriem (aptuveni 1 ms), tomēr cilvēka smadzeņu pārliecinoši lielo potenciālu nodrošina vismaz 10 miljardi neironu, kas katrs savienots ar citiem neironiem ar vairāku tūkstošu saišu starpniecību, un to kopskaits sasniedz 60 triljonus.

### 1.1.2. Neironu tīklu attīstības īsa hronoloģija

Ne visā neironu tīklu attīstības laikā ir valdījusi nešaubīga pārliecība par pētāmā objekta iespējām un noderīgumu.

Par neironu tīklu pirmsākumu uzskatāms 1943. gads, kad **Makaloks** un **Pitss** (*McCulloch, Pitts*) savā darbā *"A logical calculus of the ideas immanent in nervous activity"* [mcculloch43] aprakstīja struktūru, kuru šobrīd mēs pazīstam kā neironu tīklus. Abi zinātnieki parādīja, ka, izmantojot pietiekoši lielu skaitu vienkāršu skaitļošanas elementu, var panākt, ka "neironu tīkls" principā spēj izrēķināt jebkuru izrēķināmu funkciju.

1949. gadā **D. Hebs** (*Donald Hebb*) savā darbā *"The Organization of Behavior"* [hebb49] pirmo reizi aprakstīja to, ko mēs saprotam ar apmācības algoritmu. D. Hebs postulēja nu jau slaveno neironu tīklu apmācības stratēģiju – t.s. **Heba apmācības likumu** (*Hebb's learning rule*). Heba apmācības likums balstās uz vienkāršu principu: ja divi savstarpēji savienoti neironi ir aktīvi vienlaicīgi, tad saites svars starp neironiem tiek noteiktā veidā pastiprinātas. Heba darbs savā laikā atstāja lielu ietekmi un psihologiem, taču diemžēl ne uz matemātiķiem un inženieriem.

Viens no pirmajiem populārajiem rezultātiem neironu tīklu nozarē tika iegūts 1958. gadā, kad **F. Rozenblats** (*Frank Rosenblatt*) kopā ar saviem līdzstrādniekiem Masačūsetsas tehnoloģiskajā institūtā (*MIT*) izveidoja t.s. **perceptronu** (*perceptron*) [rosenblatt62]. Rozenblata darbs vainagojās ar t.s. **perceptrona konverģences teorēmu**, kas nosaka ka perceptrona apmācības process galīgā laikā konverģē jebkurai problēmai, ja vien tā principā ir risināma ar perceptronu.

1960. gadā **B. Vidrovs** un **M. Hofs** (*Bernhard Widrow, Marcian E. Hoff*) izveidoja perceptronam diezgan līdzīgu modeli – ***Adaline*** [widrow60].

1969. gadā iznāca **M. Minska** un **S. Peiperta** (*Marvin Minsky, Seymour Papert*) slavenā grāmata *"Perceptrons"*, kurā tika parādīti daudzi perceptrona modeļa ierobežojumi (piemēram, nespēja risināt **XOR problēmu**), kas lielā mērā bija par pamatu intereses apsīkumam par neironu tīkliem [minsky69].

Tajā laikā jau bija zināmas arī neironu tīklu vairākslāņu struktūras, kas pārvarēja perceptronam uzstādītos ierobežojumus, tomēr nebija apmācības algoritma tām.

1974. gadā **Pols Verboss** (*Paul Werbos*) savā disertācijā aprakstīja ideju, kā vairākslāņu tīklos apmācīt iekšējos neironus. Diemžēl t.s. ***backpropagation*** algoritms ienāca apritē daudz vēlāk – 1986. gadā, un tikai tad tika novērtēts P. Verbosa sasniegums, kurš savā laikā palika nepamanīts.

Jau, sākot ar 20. gs. 70. gadu sākumu, neironu tīklu lauciņā aktīvs bija **T. Kohonens** (*Teuvo Kohonen*), kurš 80. gadu sākumā izveidoja t.s. **Kohonena modeli** [kohonen84].

70. gados lielu ieguldījumu neironu tīklu attīstībā deva arī **S. Grosbergs** (*Stephen Grossberg*), kurš cita starpā izveidoja t.s. **ART modeli**.

1982. gadā fiziķis **Dž. Hopfīlds** (*John Hopfield*) izveidoja t.s. **Hopfīlda modeli**, kas balstās uz teorētiskās fizikas (termodinamikas) metodēm.

Starp 70. un 80. gadu sasniegumiem vēl varēt minēt **Fukušimas** (*K. Fukushima*) modeļus ***cognitron*** un ***neocognitron***.

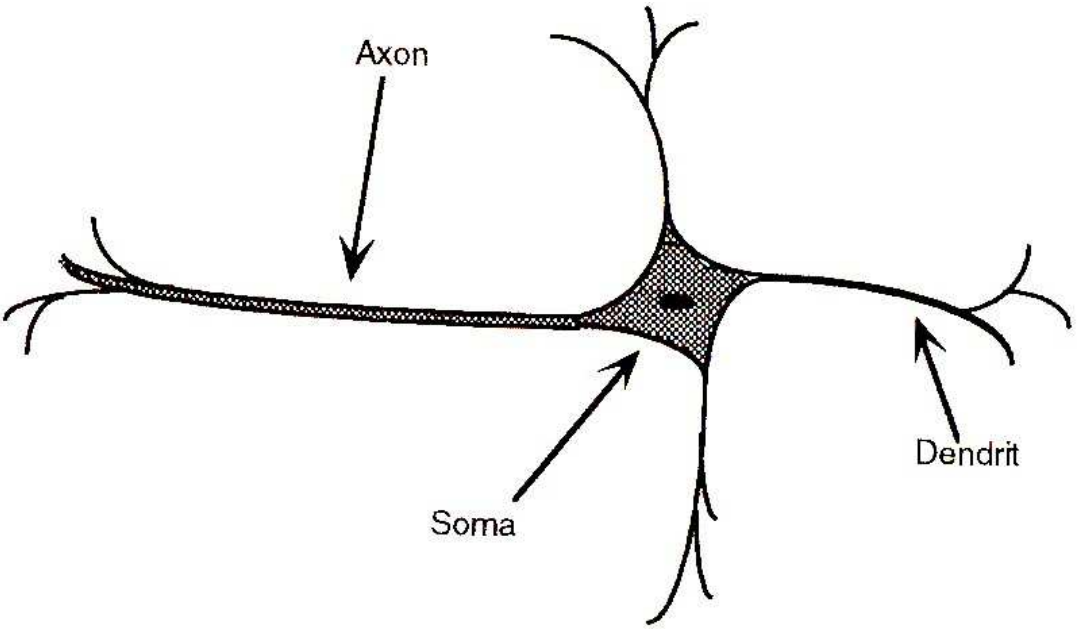
Tomēr 70. gados un līdz par 80. gadu pirmajai pusei neironu tīklu zinātnē bija vērojams izteikts panīkums, kuru 1986. gadā pārtrauca trīs autoru (*D.E. Rumelhart, G.E. Hinton, R.J. Williams*) piedāvātais t.s. ***backpropagation*** (**kļūdas atgriezeniskās izplatīšanās**) algoritms vairākslāņu perceptrona apmācībā [rummelhart86]. Paralēli algoritms tika atklāts vēl dažās vietās, un, kā jau minēts, to aprakstīja jau P. Verboss 1974. gadā, tomēr lielākie nopelni tika piešķirti tieši šiem 3 autoriem, jo tie arī ieviesa šo ideju un parādīja to darbībā.

1988. gadā tika aprakstīti t.s. **Radiālo bāzes funkciju** (*RBF*) tīkli.

20. gs. 90. gadu pašā sākumā **V.N. Vapniks** (*Vapnik*) un līdzstrādnieki piedāvāja jaudīgu neironu tīklu modeli – t.s. ***support vector machines***, kas pašlaik blakus vairākslāņu perceptronam ir viens no populārākajiem neironu tīklu modeļiem.

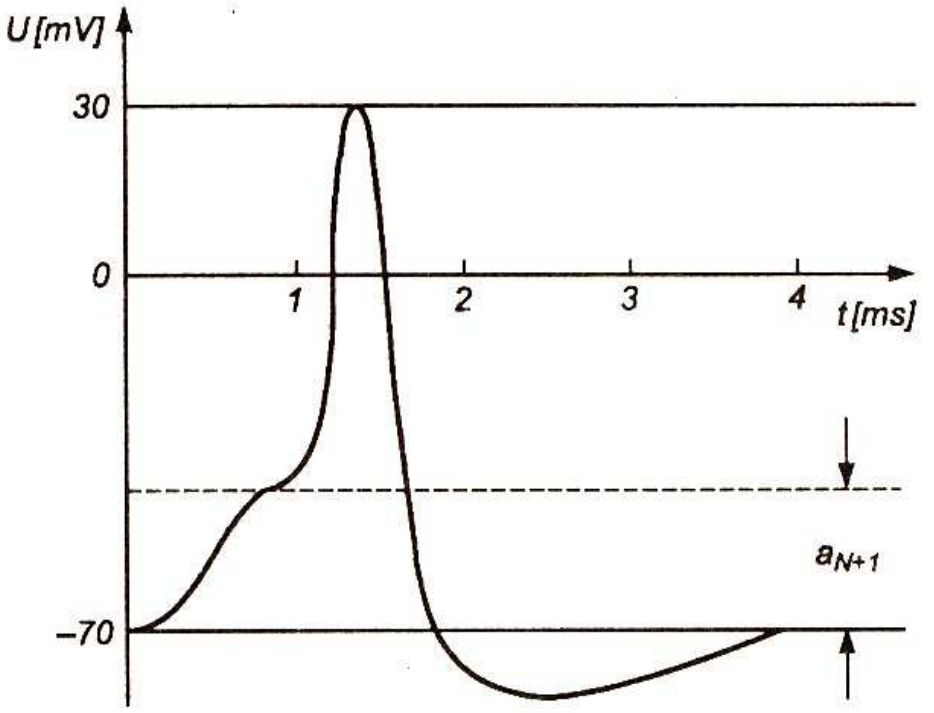
### 1.1.3. Mākslīgo neironu tīklu bioloģiskie prototipi

Bioloģiskas nervu sistēmas galvenais elements ir **neirons** jeb nervu šūna. Kā jau katrai šūnai, arī neironam ir ķermenis, saukts par **somu** (*soma*). No somas iziet neskaitāmi izaugumi, kurus var sadalīt divās grupās – tievie un biezi sazarotie **dendrīti** (*dendrites*) un resnākais **aksons** (*axon*). Ienākošie signāli nervu šūnā ienāk caur **sinapsēm** (*synapses*), bet izejošais signāls tiek aizvadīts pa aksonu, kas tālāk sazarojas, lai novadītu signālu uz vairākiem citiem neironiem.




**Attēls 1. Bioloģiskā neirona shematiska uzbūve [scherer97]**

Sinapse ir savienojuma elements, ka savieno viena neirona aksonu ar cita neirona dendrītu, somu vai aksonu. Sinapses atšķiras viena no otras pēc caurlaidības, tāpēc vienāda stipruma impulsi, nonākot neironā caur dažādām sinapsēm, to var kairināt dažādās pakāpēs. Tādējādi katram neironā ienākošajam signālam var piekārtot skaitlisku koeficientu, kā tas arī tiek darīts mākslīgajos neironu modeļos. Sinapses ietekme uz signālu var būt gan pastiprinoša, gan pavājinoša. Caur sinapsēm ienākošie signāli neironu regulāri kairina. Ja kairinājuma (neirona aktivizācijas) līmenis pārsniedz noteiktu slieksni, neirons izejā dod signāla impulsu, kura lielums ir atkarīgs no tā, cik daudz pārsniegts slieksnis – neirons "izšauj". Turpretī, ja neirona aktivizācijas līmenis nesasniedz slieksni, tad neirons "neizšāvis" atgriežas iepriekšējā stāvoklī. Pēc kārtējā nervu impulsa ģenerēšanas neirons kādu brīdi zaudē spēju to atkal izdarīt pat pie ļoti spēcīgas kairināmības.




**Attēls 2. Tipiska nervu impulsa forma [osowski02]**

## *1.2. Neironu tīklu uzbūves un darbības galvenie principi*

**Neironu tīkls** ir skaitļošanas sistēma, kas sastāv no liela daudzuma skaitļošanas elementu, sauktu par **neironiem**, kas noteiktā veidā savienoti viens ar otru, un kopā spēj veikt noteiktus uzdevumus. Atšķirībā no tradicionālajām skaitļošanas sistēmām, neironu tīkls ir drīzāk līdzīgs "melnajai kastei" – katra atsevišķa neirona loma kāda konkrēta uzdevuma risināšanā vispārīgā gadījumā nav nosakāma.

Ir daudz dažādu neironu tīklu modeļu, tomēr noteiktā abstrakcijas līmenī to uzbūve ir līdzīga. Trīs galvenie neironu tīkla konstruktīvi-funkcionālie bloki ir:

- neirons,
- tīkla topoloģija,
- apmācības algoritms.

Neironu tīkla uzbūves un darbības apraksts balstīts uz [scherer97].

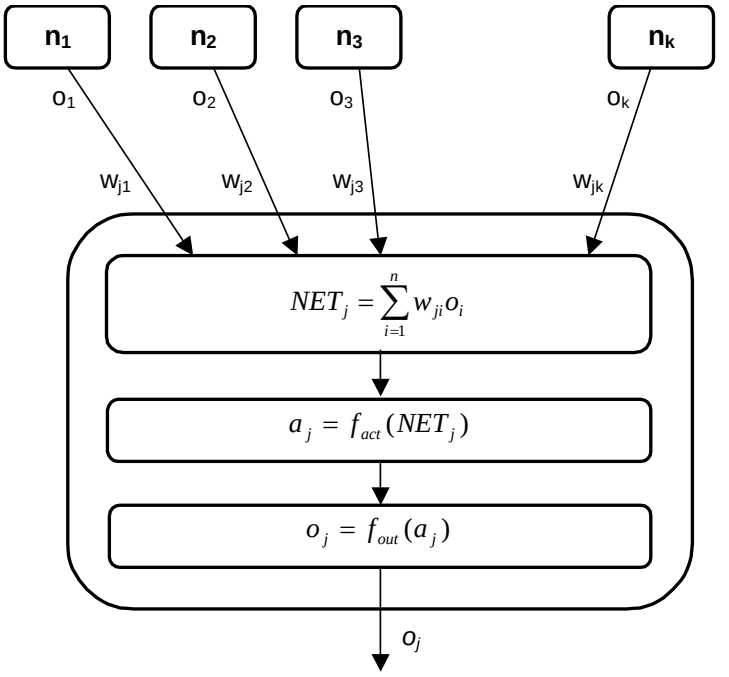
## 1.2.1. Neirons

### 1.2.1.1. Neirona vispārējā uzbūve

**Neirons** (*neuron*, *processing unit*) ir salīdzinoši vienkāršs skaitļošanas elements, kam ir vairākas **ieejas** (*inputs*) un tikai viena **izeja** (*output*) , bet tā darbību nodrošina noteikts daudzums tajā glabāto skaitlisku vērtību, kas tiek sauktas par **svariem** (*weights*).

Neirons funkcionāli līdzinās vairāku argumentu matemātiskai funkcijai, un tā darbināšana notiek šādi (Attēls 3, indekss j aprakstā norāda neirona kārtas numuru):

- uz neirona ieejām $o_i$ *(i=1..k)* tiek padotas vai nu citu neironu $n_1..n_k$ izejas vērtības vai ārējas vērtības,
- **summēšanas funkcija** (*propagation function*), izmantojot (j-tā) neirona svarus (*weights*, $w_{j1}..w_{jk}$) un tiem atbilstošās ieejas $o_i$, izrēķina summēšanas vērtību $NET_j$,
- vērtība $NET_j$ tālāk tiek apstrādāta, izmantojot **aktivizācijas funkciju** (*activation function*, $f_{act}(\cdot)$) un **izejas funkciju** (*output function*, $f_{out}(\cdot)$), un tādējādi tiek iegūta (j-tā) neirona izejas vērtība.




**Attēls 3. Neirona uzbūve**

### 1.2.1.2. Summēšanas funkcija

**Summēšanas** (jeb izplatīšanās) **funkcija** (*propagation function*) nodrošina to, ka no vairākām ieejām, izmantojot svarus, tiek iegūta viena vērtība – *NET*.

Tipiska summēšanas funkcija kontrolēti apmācāmiem neironu tīkliem parādīta formulā (1) – tā ir summa no ieeju ($o_i$) reizinājumiem ar attiecīgajiem neirona svariem ($w_{ji}$):

$$NET_j = \sum_{i=1}^{n} w_{ji} o_i \qquad \textbf{(1)}$$

Pašorganizējošos neironu tīklos bieži izmanto summēšanas funkciju, kur ieejas vērtība netiek reizināta ar attiecīgo svaru, bet gan rēķināts attālums starp tiem, kā parādīts formulā (2):

$$NET_j = \sum_{i=1}^{n} (w_{ji} - o_i)^2 \qquad \textbf{(2)}$$

#### 1.2.1.3. Aktivizācijas funkcija

**Aktivizācijas funkcija** (*activation function*), izmantojot summēšanas funkcijas vērtību *NET*, izrēķina t.s. **aktivitātes stāvokli** (*activation state*). Aktivizācijas funkcijai ir ļoti liela ietekme uz neirona skaitļošanas spēju. Aktivizācijas funkcija var būt lineāra, vai nelineāra. Starp nelineārām funkcijām izšķir sigmoidālās un sliekšņveida. Sigmoidālo funkciju ļoti vērtīga īpašība ir tāda, ka to atvasinājums ir izsakāms ar pašas funkcijas vērtību, kas ir ļoti nozīmīgi tāda neironu tīklu modeļa, kā perceptrons apmācībā. Aktivizācijas funkcijas vispārējā forma parādīta formulā (3):

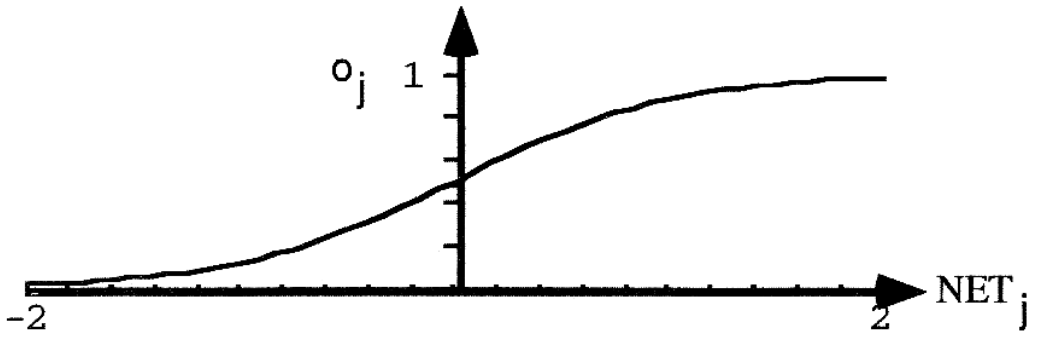
$$o_j = f_{act}(NET_j) \qquad \textbf{(3)}$$

Viena no populārākajām aktivizācijas funkcijām ir **loģistiskā funkcija** (formula (4)):

$$a_j = \frac{1}{1 + \exp(\frac{1}{g} NET_j)}, \qquad \textbf{(4)}$$

kur $g$ – līknes slīpuma koeficients (*gain*), kas tiek eksperimentāli noskaidrots, lai nodrošinātu maksimālu aktivizācijas funkcijas jūtīgumu.

Lai gan vispārējā neirona struktūrā aiz aktivizācijas funkcijas seko izejas funkcija, tomēr praktiski izejas funkciju atsevišķi neizdala un aktivizācijas funkcijas rezultāts ir arī visa neirona rezultāts ($o_j=a_j$).




**Attēls 4. Loģistiskā aktivizācijas funkcija pie g=0.5**

### 1.2.2. Tīkla topoloģija

Neironu tīkla **topoloģija** jeb **arhitektūra** ir neironu izvietojums tīklā un to savstarpējās saites. Neironi ir salīdzinoši vienkārši skaitļošanas elementi, un neironu tīkla iespējas nodrošina tas, ka neironi darbojas kopā, būdami sasaistīti viens ar otru.

Neironu tīklu, līdzīgi kā grafu, var definēt kā neironu un saišu starp neironiem kopumu, tomēr parasti neironi neironu tīklā tiek grupēti t.s. **slāņos** (*layers*) un neironu tīkla darbināšanu veic pa slāņiem.

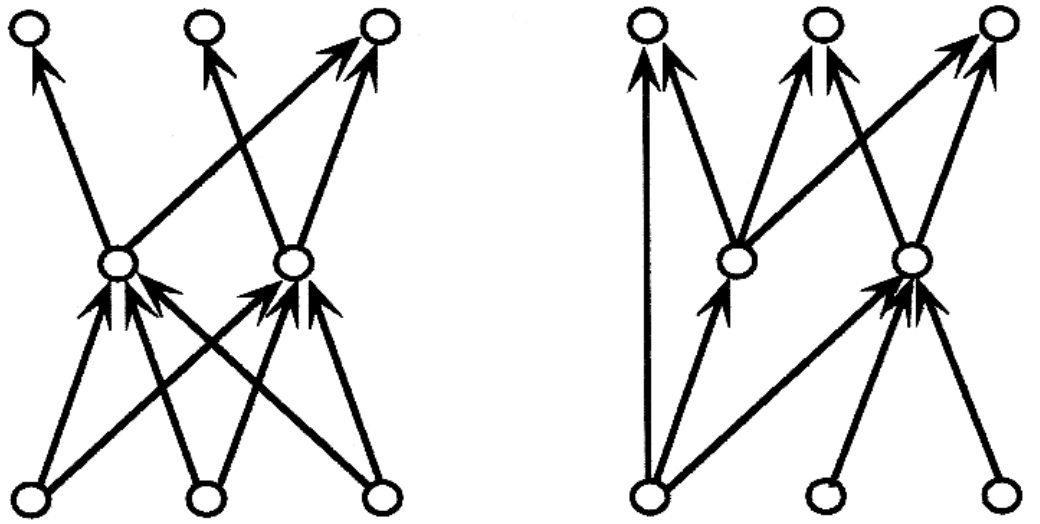
Neironu tīkla topoloģiju nosaka:

- neironu slāņu izkārtojums vienam aiz otra,
- neironu sasaiste starp slāņiem,
- neironu sasaiste (sadarbība) slāņa ietvaros.

Pēc neironu slāņu izkārtojuma neironu tīkli iedalās divos galvenajos veidos:

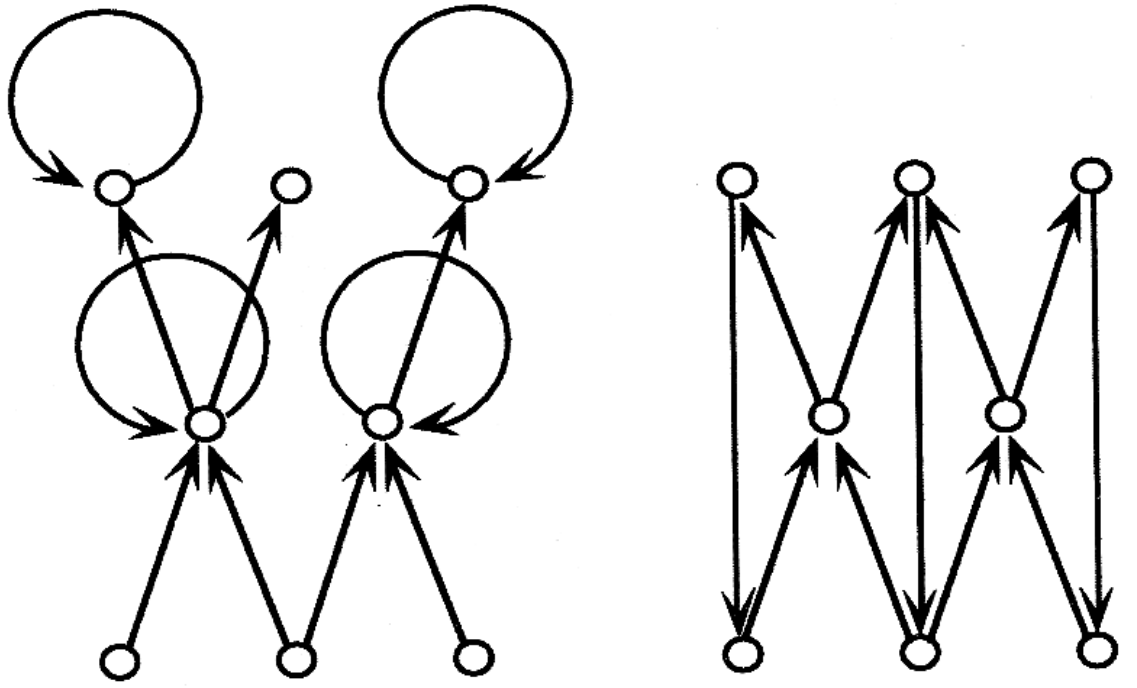
- vienvirziena tīkli;
- tīkli ar atgriezeniskajām saitēm.

**Vienvirziena tīklos** (*feedforward networks*) ir izplatītākais topoloģijas veids, un tajos slāņi ir izkārtoti virknē pēc kārtas, līdz ar to tos var numurēt pēc kārtas. Tīkla darbināšana notiek, sākot ar **ieejas slāni** (*input layer*) un beidzot ar **izejas slāni** (*output layer*). Saites starp neironiem ir tikai virzienā no slāņa ar mazāku numuru uz slāni ar lielāku numuru.



**Attēls 5. Vienvirziena tīklu piemēri**

**Tīkli ar atgriezeniskajām saitēm** (*feedback networks*) pieļauj saites no slāņiem ar lielāku numuru uz slāņiem ar mazāku numuru, kā arī slānim pašam uz sevi



**Attēls 6. Tīklu ar atgriezeniskām saitēm piemēri**

**Neironu sasaiste starp slāņiem.** Parasti starp blakus esošajiem slāņiem neironi ir sakārtoti katrs ar katru. Tādējādi neironam parasti ir tik daudz svaru, cik iepriekšējā slānī neironu (katrai neirona ieejai pretī ir svars).

**Neironu sadarbība slāņa ietvaros.** Dažos neironu tīklu modeļos ir nozīme neironu izvietojumam un sadarbībai viena slāņa ietvaros. Var tikt noteikta topoloģija (specifisks

neironu izkārtojums) slāņa ietvaros un neironu darbināšanas un apmācības algoritmos var parādīties blakus neironu ietekme.

**Ieejas, izejas un slēptie slāņi.** Slāni, kuram tiešā veidā var padot vērtības sauc par **ieejas slāni** (*input layer*), bet slāni, kura izejas ir arī visa neironu tīkla izejas, sauc par **izejas slāni** (*output layer*). Visi pārējie slāņi (tādi, kuriem nav tiešas sasaistes ar apkārtni) tiek saukti par **slēptajiem slāņiem** (*hidden layers*). Slēptajiem slāņiem vairākos neironu tīklu modeļos ir būtiska nozīme neironu tīkla darbības spējas nodrošināšanā.

### 1.2.3. Apmācības algoritms

**Apmācība** (*training; learning*) ir neirona tīkla svaru vērtību (un dažreiz arī citu parametru) uzstādīšana, balstoties uz apmācības paraugiem, kas reprezentē noteiktu problēmu.

**Apmācības algoritms** (*training algorithm; learning algorithm*) ir ļoti svarīga neironu tīkla uzbūves sastāvdaļa, un katram neironu tīklu modelim tā ir atšķirīga, tomēr ir nosaucamas vairākas pamatstratēģijas un pamatprincipus, kuras tiešāk vai netiešāk tiek pielietotas vairuma neironu tīklu apmācībā.

#### 1.2.3.1. Apmācības mērķi un uzdevumi

Runājot par neironu tīklu apmācību, būtu jāizšķir divi saistīti, bet atšķirīgu līmeņu jēdzieni – **apmācības mērķi** un **apmācības uzdevumi**.

1. Neironu tīkla apmācības mērķis – iegūt neironu tīkla spēju risināt noteiktus uzdevumus.
2. Neironu apmācības tiešie uzdevumi – neironam vai neironu grupai iegūt zināmu (lokālu) spēju tā, lai neironu tīkls kopumā spētu risināt noteiktus uzdevumus.

Apmācības process ir lokāli globāls, jo mērķi ir globāli (visa neironu tīkla spējas iegūšana), bet līdzekļi to sasniegšanai ir lokāli – atsevišķu neironu vai neironu grupas apmācība.

Nav tiešas saites starp lokālo un globālo apmācības aspektu, jo tāda jau ir paša neironu tīkla būtība, ka:

- no vienas puses – visi neironi potenciāli piedalās visu problēmu risināšanā,
- no otras puses – vispārīgā gadījumā principā nav definējama katra atsevišķa neirona loma katras atsevišķās problēmas risināšanā.

Bez tam ne visos neironu tīklu modeļos var viegli novērtēt, vai ir sasniegti apmācības mērķi.

Neironu tīklu izpēte, par ideālu pieņemot cilvēka smadzenes, vēl ir ļoti zemā līmenī, un to raksturo salīdzinoši zemie apmācības mērķi, kas bieži vien sakrīt ar tiešajiem uzdevumiem. Vēl viens aspekts, kas raksturo salīdzinoši zemo neironu tīklu attīstības līmeni, ir to pielietojuma pakāpe dažādās sistēmās: neironu tīklu moduļiem lietojumprogrammās parasti ir palīgloma, jo vadību un kontroli veic tradicionāli izveidoti programmas bloki, bet neironu tīkli tiek izmantoti tikai atsevišķu palīgoperāciju izpildei.

#### 1.2.3.2. Divas neironu tīklu apmācības kategorijas

Divas galvenās neironu tīklu apmācības kategorijas ir **apmācība ar skolotāju** (*with a teacher*) un **bez skolotāja** (*without a teacher*). Šī darba ietvaros apmācība ar skolotāju tiek saukta arī par kontrolēto apmācību.

**Neironu tīkli ar apmācību ar skolotāju**, kuru tipiskākais pārstāvis ir perceptrons, lielāko tiesu darbojas kā funkciju aproksimatori, kas citu funkcijas aproksimācijas metožu vidū izceļas ar to, ka savu funkcionēšanas spēju iegūst apmācības ceļā. Apmācības dati sastāv no noteikta skaita "jautājumu" un "pareizo atbilžu". Neironu tīklam tiek uzdots jautājums un tā dotā atbilde tiek salīdzināta ar "pareizo atbildi". Apmācības uzdevums ir kļūdas samazināšana starp neironu tīkla sniegto atbildi un iepriekš zināmo "pareizo atbildi". Šo algoritmu priekšrocība ir iespēja precīzi novērtēt tīkla darbības pareizību un precīzi noteikt veicamās darbības pareizības palielināšanā. Pateicoties šādai noteiktībai, nosauktie algoritmi ir salīdzinoši viegli pielietojami, diezgan efektīvi un tādējādi arī populāri.

**Neironu tīklu apmācība bez skolotāja.** Šādus apmācāmus tīklus bieži sauc par pašorganizējošiem tīkliem. Pašorganizējošo neironu tīklu galvenā iezīme ir tāda, ka tajos nav tāda jēdziena kā "pareizā atbilde", kuram savukārt pie neironu tīkliem ar apmācību "ar skolotāju" ir izšķiroša nozīme. Tas nozīmē, ka nav precīzi definējams kritērijs neironu tīkla funkcionēšanas pareizībai.

Šai iezīmei ir gan pozitīvā, gan negatīvā puse.

1. Pozitīvā puse ir tāda, ka apmācības procesā nav nepieciešami testa piemēri, kuriem būtu dotas arī "pareizās atbildes". Īpaši svarīgi tas ir tādēļ, ka daudzām problēmām nemaz iepriekš nevar noteikt, kādām būtu jābūt "pareizajām atbildēm".
2. Negatīvā puse ir salīdzinoši sarežģītākais šādu neironu tīklu pielietojums, jo ne vienmēr var saredzēt šādu neironu tīklu pielietojamību konkrētu problēmu risināšanā. Bez tam, apskatot to neironu tīklu konstrukcijas līmenī, vispārīgā gadījumā ir daudz sarežģītāk noteikt apmācības procesa uzdevumus.

#### 1.2.3.3. Apmācības process

**Apmācības process** (*training process*) ir darbību kopums, kura laikā neironu tīklam noteiktā secībā vairākkārtīgi tiek padoti apmācības paraugi, pēc katra apmācības parauga vai paraugu kopuma padošanas katram neironam piemērojot apmācības algoritmu.

Apmācības procesa sākumā neironu tīkla svari noteiktā veidā (parasti – nejauši) tiek aizpildīti ar sākuma vērtībām.

Apmācības process parasti notiek pa epohām. **Epoha** (*epoch*) ir apmācības procesa daļa, kuras laikā neironu tīklam tieši vienu reizi tiek padoti visi apmācības paraugi.

Parasti apmācības algoritms tiek izsaukts pēc katra parauga padošanas, un apmācības procesa viens solis sastāv no šādām fāzēm:

1. **Darbināšanas fāze.** Apmācības parauga padošana neironu tīklam un neironu tīkla darbināšana, iegūstot noteiktu rezultātu.
2. **Apmācīšanās fāze.** Balstoties uz darbināšanas fāzē iegūto rezultātu, apmācības paraugu un "vēlamo atbildi" (ja tāda ir pieejama), apmācības algoritms veic svaru izmaiņu neironos.

## *1.3. Neironu tīklu pazīstamākie modeļi*

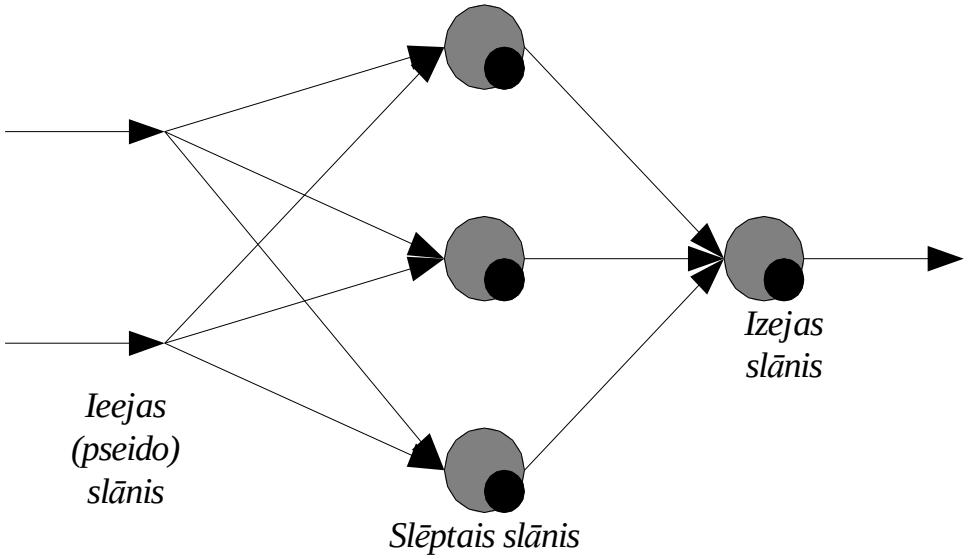
Šajā nodaļā tiks aprakstīti 3 neironu tīklu modeļi: vairākslāņu perceptrons, radiālo bāzes funkciju tīkls un Kohonena tīkls. Pirmie divi ir tiešā veidā izmantoti autora veiktajos eksperimentos, bet trešais modeļa uzbūves un darbības idejas ir kalpojušas par iedvesmu vienai no autora piedāvātajām un tālāk darbā aprakstītajām neironu tīklu apmācības metodēm. Modeļu apraksts balstīts uz [scherer97].

### 1.3.1. Vairākslāņu perceptrons

Viens no populārākajiem neironu tīklu modeļiem ir **vairākslāņu perceptrons** (*multi-layer perceptron*, *MLP*) komplektā ar **kļūdu atgriezeniskās izplatīšanās apmācības algoritmu** (*backpropagation*). Vairākslāņu perceptrons ir izmantojams klasifikācijas, funkciju aproksimācijas, prognozēšanas un citu problēmu risināšanai.

#### 1.3.1.1. Uzbūve un darbības principi

- Vairākslāņu perceptronam ir vismaz trīs slāņi (Attēls 7) – ieejas slānis (*input layer*), vismaz viens slēptais slānis (*hidden layer*) un izejas slānis (*output layer*). Ieejas slānis, uzskatāms par neīstu jeb pseido-slāni, jo tas paredzēts tikai tehniskām vajadzībām, lai atvieglotu tā realizāciju datorprogrammā – uz tā izejām tiek uzstādītas neironu tīkla ieejas, bet paša slāņa neironos nekāda darbība nenotiek.
- Slāņi izvietoti rindā (lineāri) viens aiz otra. Katram slānim, izņemot ieejas slāni, ir iepriekšējais slānis, un katram slānim, izņemot izejas slāni, ir nākošais slānis. Divu blakus stāvošo slāņu neironi ir savienoti katrs ar katru. Tādējādi katrā neironā ir tik daudz svaru, cik iepriekšējā slānī neironu (neskaitot papildus svaru).
- Svari ir robežās *[-q; +q]*, kur *q* ir jebkurš pozitīvs skaitlis, bet parasti ievietojas intervālā *[-1; +1]*.
- Neironu izejošās vērtības ir intervālā *[0; 1]*. Tas attiecas arī uz ieejas slāni, tādējādi arī visa neironu tīkla ieejas vērtības ir padodamas šajā intervālā.
- Katrā neironā ir viens papildus svars (bez atbilstošās ieejas no ieejas slāņa), saukts par nobīdi (*bias*) vai līdzsvarojošo elementu. Ērtības labad šis papildus svars bieži tiek uzskatīts par parastu svaru un numurēts ar indeksu 0, un līdz ar to klāt nāk viena papildus ieeja, kas konstanti aizpildīta ar 1.
- Neirona uzbūve kopumā atbilst visumā atbilst attēlam (Attēls 3). Obligāts nosacījums ir tas, ka tiek lietota nelineāra (sigmoidāla) aktivizācijas funkcija. Izņēmums varētu būt vienīgi izejas slānis. Lietojot lineāru aktivizācijas funkciju, daudzslāņu perceptrons neiegūtu tās priekšrocības, kas tam ir, salīdzinot ar vienslāņa perceptronu, respektīvi, nespētu risināt nelineāras problēmas.




**Attēls 7. Vairākslāņu perceptrona grafs ar 2 ieejām, 3 neironiem slēptajā slānī un vienu izeju. Katrā slēptā un izejas slāņa neironā ir papildus svars – nobīde.**

### 1.3.1.2. Darbināšana

Vairākslāņu perceptrona darbināšana (*operation*) notiek pa slāņiem pēc kārtas – sākot ar vērtību uzstādīšanu ieejas slānī un beidzot ar izejas slāņa darbināšanu. Katra slāņa darbināšana nozīme visu slāņa neironu darbināšana pēc kārtas.

Neirona darbināšana notiek, izmantojot formulas (5) un (4) (sigmoidālas aktivizācijas funkcijas gadījumā).

Formula (5) ir formulas (1) paplašinājums perceptrona gadījumam: perceptrona neironā summēšanas funkcijai ir papildus saskaitāmais – nobīde jeb papildus svars (*bias*) $b_{ij}$:

$$NET_j = \sum_{i=1}^{n} w_{ji} o_i + b_j \qquad \textbf{(5)}$$

Formula (5) vieglākas realizācijas nolūkiem ir izsakāma arī citādākā formā (formula (6)):

$$NET_j = \sum_{i=0}^{n} w_{ji} o_i \ , \qquad \textbf{(6)}$$

kur $w_{j0}$ ir papildus svars (*bias*), kas tiek lietots $b_j$ vietā, bet pseido-ievads $o_0=1$.

#### 1.3.1.3. Apmācība, izmantojot kļūdu atgriezeniskās izplatīšanās algoritmu

Dots neironu tīkls $\{O_1[n_1], W_2[n_2][n_1+1], O_2[n_2], ..., W_m[n_m][n_{m-1}+1], O_m[n_m]\}$, kurā ir $n_1$ ieejas un $n_m$ izejas un $m>2$. (Piemērs, kad $m=3$, $n_1=2$, $n_2=3$ un $n_3=1$, parādīts attēlā (Attēls 7)) Tiek izmantota sigmoidālā aktivizācijas funkcija.

Tāpat kā vienslāņa perceptronam, dots apmācības piemēru kopums $P$, kurā ir piemēri skaitā $r$. Katrs apmācības piemērs tiek identificēts kā $\{P_s, D_s\}$, kur $s=1..r$. Katra ieejas parauga $P_s$ garums ir $n_1$, bet izejas parauga (vēlamās atbildes) $D_s$ garums ir $n_m$.

**Apmācības process notiek šādi:**

1. Visu svaru inicializācija ar gadījuma vērtībām intervālā $[-x, +x]$, kur $0<x<1$. Parasti $x$ vērtība ir tuvāka nullei nekā 1, piemēram, 0.3.
2. Apmācības procesa veikšana pēc kārtas uz visiem apmācības piemēriem, atkārtojot to tik ilgi, kamēr tiek sasniegts konverģences nosacījums – vai nu tīkla kopējā kļūda kļūst pietiekoši maza vai arī iterāciju skaits kļūst pārāk liels.

**Katrā apmācības procesa solī tiek veiktas šādas darbības:**

1. Tīkla inicializācija ar parauga $P_s$ vērtībām.
2. Tīkla darbināšana ar doto paraugu pēc kārtas visiem slāņiem, sākot ar 2. slāni un beidzot ar izejas slāni (formulas (5),(4)).
3. Svaru korekcija pēc kārtas visiem slāņiem pretējā secībā (7) – sākot ar izejas slāni un tad pārējiem slāņiem – beidzot ar 2. slāni. Svarīgi, ka izejas slānim apmācība notiek savādāk nekā pārējiem.

Viena perceptrona neirona apmācība balstās uz kļūdas minimizācijas principu – svari tiek izmainīti tā, lai starpība starp vēlamo vērtību un neirona doto rezultātu samazinātos. Tas tiek veikts, izmantojot gradientu metodi. Vispārināta viena svara izmaiņa *backprop* apmācības algoritmā parādīta formulā (7) [zuters05b]. Pēc šīs formulas ir redzams, ka svara izmaiņa ir tieši proporcionāla caur šo svaru ienākošajam signālam.

$$\Delta w_{ji} = \eta \delta_j o_i , \qquad \textbf{(7)}$$

kur $\delta_j$ – lokālais gradients.

**Lokālais gradients** (*local gradient*) ir lielums, kas reprezentē vēlamo svara apmācības tendenci, balstoties uz neironu tīkla kļūdu, darbinot to uz noteikta apmācības parauga. Tas tiek atšķirīgi noteikts izejas slāņa un slēpto slāņu neironiem (attiecīgi formulas (8),(9)).

Lokālā gradienta aprēķināšana ir atkarīga no diviem būtiskiem faktoriem:

- aktivizācijas funkcijas veida (lokālais gradients ietver aktivizācijas funkcijas atvasinājumu),

- apmācāmā neirona atrašanās vai nu izejas vai slēptajā slānī (lokālajā gradientā ietilpst vai nu kļūda tiešā veidā vai kļūdas vispārinājums).

Tālāk parādīts lokālā gradienta aprēķins, pieņemot, ka par aktivizācijas funkciju lietota (4).

Lokālā gradienta $\delta_j$ aprēķins izejas slāņa neironiem parādīts formulā (8):

$$\delta_j = \frac{1}{g} o_j (1 - o_j)(d_j - o_j), \qquad \textbf{(8)}$$

kur $o_j$ – neirona j izrēķinātā izejas vērtība, $d_j$ – neirona j vēlamā izejas vērtība, $g$ – līknes slīpuma koeficients (tas pats, kas aktivizācijas funkcijā (4))

Lokālā gradienta $\delta_j$ aprēķins slēpto slāņu neironiem parādīts formulā (9):

$$\delta_j = \frac{1}{g} o_j (1 - o_j) \sum_{k=1}^{n} \delta_k w_{kj}, \qquad \textbf{(9)}$$

kur $\delta_j$ – nākošā slāņa neirona $k$ lokālais gradients; $w_{kj}$ – nākošā slāņa neirona $k$ svars $j$; $n$ – neironu skaits nākošajā slānī.

Kā redzams, slēptā slāņa neironu lokālā gradienta aprēķinā nav tiešā veidā pieejama neironu kļūda (vēlamās vērtības starpība ar aprēķināto), tāpēc to aizvieto sarežģītāka izteiksme, kas izmanto nākošā slāņa lokālos gradientus, kas reprezentē kļūdas tendenci. Viena slāņa neironu lokālā gradienta izmantošana cita slāņa neironu lokālā gradienta izrēķināšanā nodrošina kļūdas pakāpenisku izplatīšanos atpakaļ (*error backpropagation*) no izejas slāņa uz pārējiem slāņiem.

## 1.3.2. Kohonena tīkls

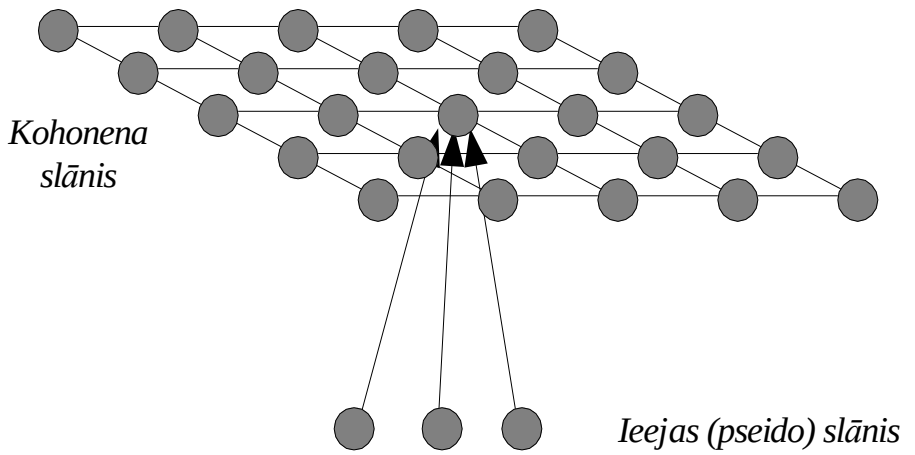
Kohonena tīkls (*Kohonen network*) pieder t.s. pašorganizējošo neironu tīklu modeļiem. Cits tā nosaukums ir "pašorganizējošā karte" (*self-organizing map*, *SOM*). Kohonena tīkls ir izmantojams paraugu klāsterizācijas problēmas risināšanā. Paraugu klāsterizācija no klasifikācijas atšķiras ar to, ka iepriekš nav zināmas grupas, kurās paraugi būtu jāsaklasificē. Paraugu sadalījums pa grupām tiek noteikts pašā klāsterizācijas procesā, zināms ir tikai grupu skaits, kurās klasificēt (neironu skaits Kohonena slānī).

### 1.3.2.1. Uzbūve un darbības principi

Kohonena tīkla galvenās konstruktīvās īpašības:

1. Līdzīgi kā vienslāņa perceptronam – ir viens ieejas slānis (pseido slānis) un viens īstais slānis (Kohonena slānis).

2. Visi ieejas slāņa neironi ir savienoti ar visiem Kohonena (izejas) slāņa neironiem. Ja neironu tīklā ir *m* ieejas un *n* izejas, tad ieejas slānī ir m neironi un attiecīgi katram izejas slāņa neironam ir m svari, kā arī izejas slānī ir n neironi.
3. Kohonena slānī ir svarīgs neironu savstarpējais izvietojums jeb topoloģija. Starp jebkuriem diviem neironiem ir noteikts attālums (kaimiņa līmenis), kam ir nozīme apmācības procesā.
4. Gan ieejas un izejas vērtības, gan svari ir robežās *[0; +1]*.




**Attēls 8. Kohonena tīkla grafs ar 3 ieejām un 25 neironiem Kohonena slānī. Saites parādītas tikai uz vienu no 25 Kohonena slāņa neironiem. Neironu izvietojums Kohonena slānī – 2-dimensiju, kvadrātisks.**

Kohonena tīkla darbības principi:

1. Apmācība notiek iteratīvi. Katrā tīkla apmācības solī uz tīkla ieejām tiek uzstādītas ieejas vērtības, visos neironos tiek veikta darbināšana, un daļai neironu (bieži tikai vienam) tiek veikta svaru korekcija.
2. Apmācības procesu beidz, ja veikts pietiekoši liels apmācības soļu skaits vai arī tīkla nosacītā kļūda kļūst stabila.
3. Atšķirībā no perceptrona – apmācības paraugiem nav dotas vēlamās vērtības.
4. Paraugu klasificēšana notiek vienā solī un salīdzinoši ātri.

#### 1.3.2.2. Darbināšana

Par summēšanas funkciju Kohonena tīkla parasti izmanto Eiklīda attālumu vai tā kvadrātu (2).

Aktivizācijas funkcijas (un attiecīgi neirona izejas) vērtība ir atkarīga no citiem neironiem slānī – tikai neirons-uzvarētājs izejā iegūst no nulles atšķirīgu vērtību.

Formula (10) parāda neirona-uzvarētāja noskaidrošanu, bet pati aktivizācijas funkcija, kas to izmanto parādīta formulā (11).

$$j_{win} = index(\min(\{NET_1, NET_1, ..., NET_n\})), \quad \textbf{(10)}$$

kur $NET_j$ – neirona j summēšanas funkcija; $j_{win}$ – neirona-uzvarētāja kārtas numurs.

$$o_j = \begin{Bmatrix} 1; j = j_{win} \\ 0; j \neq j_{win} \end{Bmatrix} \quad \textbf{(11)}$$

### 1.3.2.3. Apmācība

Dots neironu tīkls $\{O_1[m], W_2[n][m], O_2[n]\}$, kurā ir m ieejas un n izejas. (Piemērs, kad $m$=3 un $n$=25, parādīts attēlā (Attēls 8))

Tāpat kā vienslāņa perceptronam, dots apmācības piemēru kopums $P$, kurā ir piemēri skaitā $r$. Katrs apmācības piemērs tiek identificēts kā $P_s$, kur $s=1..r$. Katra ieejas parauga $P_s$ garums ir $m$.

Apmācības process ir tāds pats kā vienslāņa perceptronam un notiek šādi:

1. Visu svaru inicializācija ar gadījuma vērtībām intervālā $[0, +x]$, kur $0<x<1$.
2. Apmācības procesa veikšana pēc kārtas uz visiem apmācības piemēriem, atkārtojot to tik ilgi, kamēr tiek sasniegts konverģences nosacījums – ir veikts pietiekoši liels apmācības soļu skaits.

Katrā apmācības procesa solī tiek veiktas šādas darbības:

1. Tīkla inicializācija ar parauga $P_s$ vērtībām.
2. Tīkla darbināšana, lai noskaidrotu neironu – uzvarētāju (formulas (2),(10),(11)).
3. Svaru korekcija neironam – uzvarētājam un daļai neironu uzvarētāja apkārtnē (12).

**Svara korekcijas aprēķināšana neironam.**

Svaru koriģēšana ir svaru izmaiņa ieejas parauga virzienā (12):

$$\Delta w_{ji} = \eta(o_i - w_{ji})D(d(j, j_{win})), \quad \textbf{(12)}$$

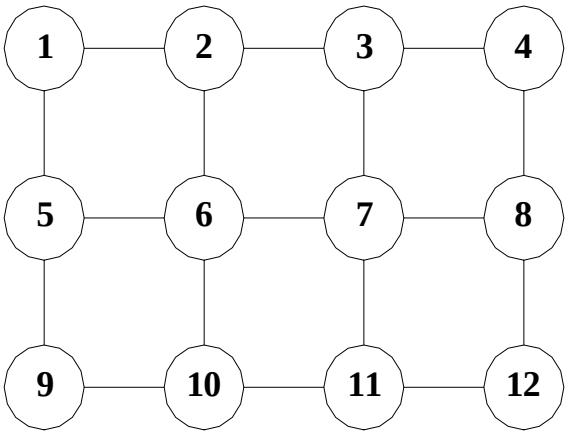
kur $\eta$ – apmācības koeficients; $o_i$ – ieeja $i$; $w_{ji}$ – neirona $j$ svars $i$; $d(\cdot,\cdot)$ – divu neironu savstarpējais attālums, ko nosaka topoloģija (piemēram, (13)); $D(\cdot)$ – kaimiņa faktors (piemēram, (14)).

**Attālums starp neironiem.**

Lai būtu iespējams noteikt attālumu starp neironiem, ir jābūt definētai slāņa topoloģijai. Piemērs ir redzams attēlā (Attēls 9). Šajā piemērā neironi izkārtoti rindās pa 4, un attālums starp neironiem d tiek definēts kā šķautņu skaits, kas nepieciešams, lai aizietu no viena neirona līdz otram (13):

$$d(i,k) = \left|(i-1) \setminus \eta_0 - (k-1) \setminus \eta_0\right| + \left|(i-1)\%n_0 - (k-1)\%n_0\right|, \qquad \textbf{(13)}$$

kur $|\cdot|$ – absolūtā vērtība (modulis); $i,k$ – neironu kārtas numuri; $n_0$ – neironu skaits vienā rindā; $\cdot\setminus\cdot$ - veselo skaitļu dalīšana; $\cdot\%\cdot$ - veselu skaitļu dalījuma atlikums.




**Attēls 9. Kohonena slāņa topoloģijas 2 dimensiju piemērs ar neironu numerāciju (atbilst formulai (13))**

**Kaimiņa funkcija (kaimiņa faktors).**

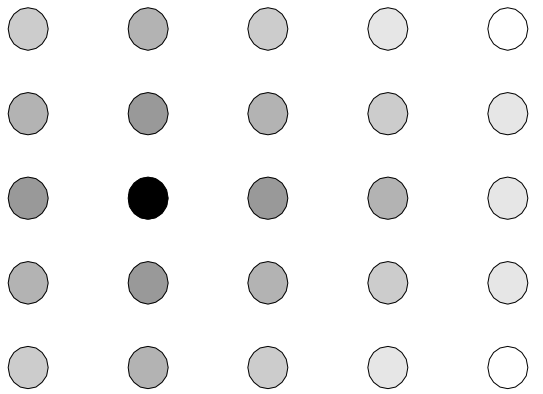
Kaimiņa funkcija nosaka pakāpi, kādā attālums no uzvarētāja neirona ietekmē dotā neirona apmācību (Attēls 10). Vienkāršākā kaimiņa funkcija ir sliekšņveida funkcija, nosakot, ka neirons-uzvarētājs apmācās par 100%, bet pārējie neapmācās nemaz.

Salīdzinoši sarežģītāka ir Gausa funkcija. Tā dod iespēju apmācīties arī citiem neironiem – noteiktu pakāpi mazākā mērā nekā uzvarētājam, un tieši šo parasti lieto kā kaimiņa funkciju (14):

$$D(z) = \exp\left(-\frac{z^2}{2\sigma^2}\right), \qquad \textbf{(14)}$$

kur $z$ – attālums starp neironiem; $\sigma$ – līknes slīpuma koeficients.

Apmācības procesu beidz, kad ir veikts noteikts skaits epohu.



**Attēls 10. Ar Gausa funkciju realizētas kaimiņa funkcijas efekts, ja neirons atrodas 3. rindas 2. pozīcijā**

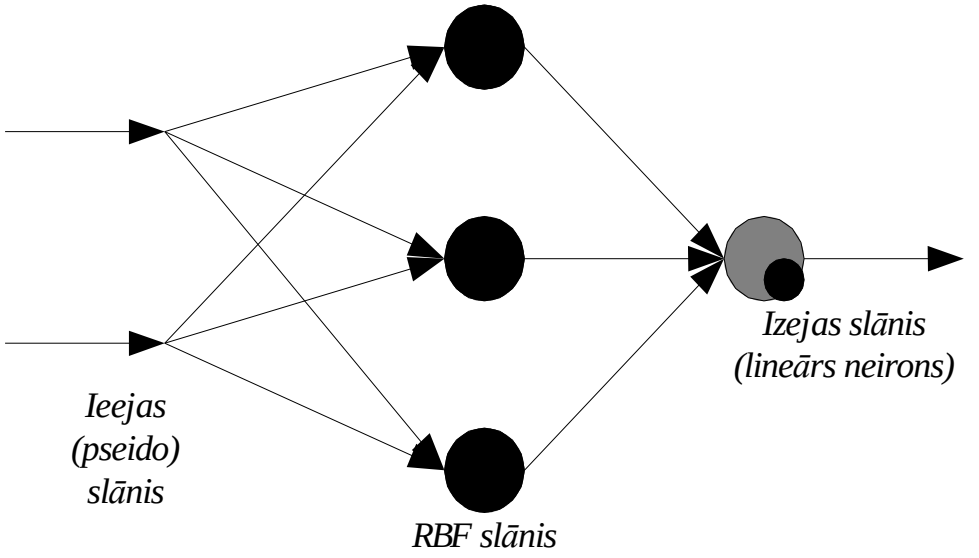
## 1.3.3. Radiālo bāzes funkciju tīkls

**Radiālo bāzes funkciju** (*radial-basis function, RBF*) **tīkls** (RBF tīkls) pieder tīkliem ar kontrolēto apmācību un pēc pielietojuma un funkcionalitātes ir līdzīgs vairākslāņu perceptronam.

### 1.3.3.1. Uzbūve un darbības principi

**RBF tīkla galvenās konstruktīvās īpašības:**

1. Līdzīgi kā daudzslāņu perceptronam ar vienu slēpto slāni – ir viens ieejas slānis (pseido slānis) un divi īstie slāņi, kas izvietoti lineāri viens aiz otra, un blakus esošo slāņu neironi ir savienoti katrs ar katru.
2. Atšķirībā no daudzslāņu perceptrona īstie slāņi ir dažādi pēc uzbūves un darbības principa (Attēls 11). 1. īstais (slēptais) slānis ir t.s. radiālo bāzes funkciju slānis (RBF slānis), bet otrs (izejas) slānis ir perceptrona tipa slānis, kurā parasti izmanto lineāru aktivizācijas funkciju.
3. RBF slāņa neironu specifiskās aktivizācijas funkcijas (Gausa funkcija) dēļ tīkls ieguvis šādu nosaukumu.
4. RBF slāņa ieejas un izejas vērtības, kā arī svari ir robežās *[0; +1]*.
5. Runājot par RBF tīkliem, bieži tiek izmantota nevis neironu tīklu terminoloģija, bet gan matemātiskā terminoloģija, jo arī termins “radiālās bāzes funkcijas” nāk no skaitļu analīzes. Pašus RBF slāņa neironus bieži sauc par radiālajām bāzes funkcijām, bet to svarus – par šo funkciju centriem.




**Attēls 11. RBF tīkla grafs ar 2 ieejām, 3 neironiem RBF slānī un vienu izeju**

**RBF tīkla darbības principi:**

1. RBF slānis apmācības dēļ būtu uzskatāms nevis par konkrētu modeli, bet gan par šablonu, kurā var ievietot dažādus algoritmus.
2. Apmācība notiek divos etapos. Vispirms tiek apmācīts RBF slānis, pēc tam izejas slānis.
3. Apmācības paraugu kopums tāpat kā perceptronam sastāv no pāriem – piemērs un vēlamā vērtība.
4. RBF slāņa apmācība notiek bez skolotāja (tātad, parasti neizmantojot vēlamās vērtības) un ir iespējams izmantot dažādus (vienkāršākus vai sarežģītākus) algoritmus. Vispārīgi runājot, RBF slāņa apmācība ir ievadparaugu klāsterizācija.
5. Izejas slānis ir perceptrons un tā tas arī tiek apmācīts. Vispirms katrs ievadparaugs tiek darbināts RBF slānī, pēc tam ar RBF slāņa izejām un vēlamajām vērtībām tiek darbināts perceptrona algoritms izejas slānī.
6. Paraugu klasificēšana notiek vienā solī un salīdzinoši ātri – tāpat kā perceptronam.

### 1.3.3.2. Darbināšana

Summēšanas funkcija RBF slānī nosaka attālumu starp ievadvērtību virkni un neirona svaru vērtību virkni. Formulā (15) tiek izmantots t.s. Eiklīda attālums, un summēšanas funkcija ir ļoti līdzīga Kohonena tīkla summēšanas funkcijai.

$$NET_j = \left\| W_j - O \right\| = \sqrt{\sum_{i=1}^{m} (w_{ij} - o_i)^2} \,, \tag{15}$$

kur $w_{ji}$ – neirona $j$ svars $i$; $o_i$ – ieejas slāņa neirona $i$ izejas vērtība.

Par aktivizācijas funkciju RBF slānī tiek izmantota Gausa funkcija.

$$o_j = \exp(-\frac{NET_j^2}{2\sigma^2}) \,, \tag{16}$$

kur $NET_j$ – summēšanas funkcija neironam $j$; $\sigma$ – līknes slīpuma koeficients.

Izejas slāņa darbināšana notiek izmantojot parasto summēšanas funkciju (5) un perceptrona aktivizācijas funkciju (piemēram (4)). RBF tīkla izejas slānī mēdz izmantot (un parasti tā arī dara) – lineāro aktivizācijas funkciju (17):

$$o_j = NET_j \tag{17}$$

#### 1.3.3.3. Apmācība

Dots neironu tīkls $\{O_1[n_1], W_2[n_2][n_1], O_2[n_2], W_3[n_3][n_2+1], O_3[n_3]\}$, kurā ir $n_1$ ieejas un $n_m$ izejas un $m$=3. (Piemērs, $n_1$=2, $n_2$=3 un $n_3$=1, parādīts attēlā (Attēls 11)).

Tāpat kā perceptronam, dots testa piemēru kopums $P$, kurā ir piemēri skaitā $r$. Katrs testa piemērs tiek identificēts kā $\{P_s, D_s\}$, kur $s=1..r$. Katra ieejas parauga $P_s$ garums ir $n_1$, bet izejas parauga (vēlamās atbildes) $D_s$ garums ir $n_m$.

RBF tīkla apmācības process notiek 2 etapos:

1. RBF slāņa apmācība.
2. Izejas slāņa apmācība.

**RBF slāņa apmācība.**

RBF slāņa apmācībā nelieto konkrētu algoritmu. RBF slānis ir šablons vai ietvars, kurā var ievietot dažādus algoritmus. RBF slāņa apmācības mērķis ir nevis gūt spēju atpazīt vai klasificēt, bet savā ziņā tieši otrādi – "sajaukt" ieejas datus tā, lai pēc sajaukšanas tie būtu atdalāmi, izmantojot lineāras metodes (piemēram, perceptrona slāni). Šai sajaukšanai dažos gadījumos der arī klāsterizācija, kas atbilst Kohonena tīklā realizētajai.

Pielietotie algoritmi RBF slāņa apmācībai ir diezgan dažādi – no ļoti vienkāršiem līdz sarežģītiem, tālāk īsi tiks precizēta triviālā un klāsterizācijas metode.

**Triviālā metode.** Katrs ieejas paraugs tiek noteikts par svaru vērtību vektoru, tādējādi RBF slānī vajadzīgi tik neironi, cik ir apmācības piemēru. Var redzēt, ka, lai ar šo metodi apmācītu tīklu atpazīt 2-argumenta Būla funkciju, RBF slānī nepieciešami 4

neironi, kas jau tik vienkāršā gadījumā ir, varētu teikt, daudz vairāk, nekā nepieciešams daudzslāņu perceptronam.

**Klāsterizācijas metode.** Ja ir daudz apmācības paraugu, tad triviālā metode nav pielietojama tīri tehnisku ierobežojumu dēļ. Tādēļ var lietot kādu no klāsterizācijas metodēm, lai uzstādītu svarus RBF slānī. Viens no vienkāršākajiem paņēmieniem ir izmantot Kohonena tīkla algoritmu RBF slāņa apmācībai.

**Izejas slāņa apmācība.**

Izejas slāņa apmācības process ir identisks vienslāņa perceptrona apmācībai un notiek šādi:

1. Visu izejas slāņa svaru inicializācija ar gadījuma vērtībām intervālā *[-x, +x]*, kur *0<x<1*. Parasti *x* vērtība ir tuvāka nullei nekā 1, piemēram, 0.3.
2. Apmācības procesa veikšana pēc kārtas uz visiem apmācības piemēriem, atkārtojot to tik ilgi, kamēr tiek sasniegts konverģences nosacījums – vai nu tīkla kopējā kļūda kļūst pietiekoši maza vai arī iterāciju skaits kļūst pārāk liels (formulas (7),(18)).

Katrā izejas slāņa apmācības procesa solī tiek veiktas šādas darbības:

1. Tīkla inicializācija ar parauga $P_s$ vērtībām.
2. RBF slāņa darbināšana.
3. Izejas slāņa darbināšana.

Izejas slāņa apmācība identiski perceptrona izejas slānim (formulas (7),(8)). Ja par aktivizācijas funkciju tiek lietota lineāra funkcija (17), tad formulas (8) vietā izmantojama (18):

$$\delta_j = \frac{1}{g}(d_j - o_j) \qquad \textbf{(18)}$$

Bez tam lineāru neironu apmācībai var vispār neizmantot neironu tīkliem pierastās iteratīvās metodes, bet gan kādu no klasiskajiem algoritmiem.

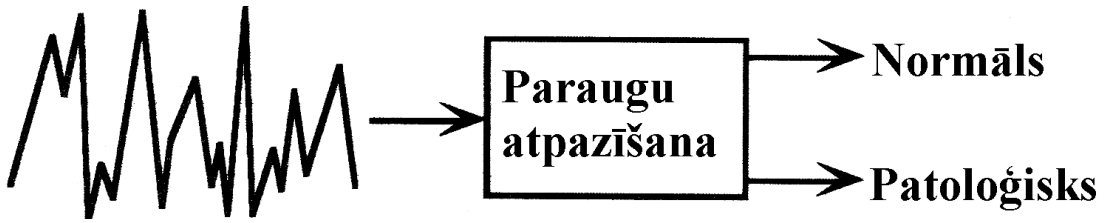
## *1.4. Ar neironu tīkliem risināmās problēmas*

Neironu tīkli nav universāls līdzeklis visu problēmu risināšanai. Pirms izmantot neironu tīklus, ir precīzi jāsaprot, vai neironu tīkli vispār būtu lietojami un kāds būtu labākais neironu tīklu modelis. Skatoties un praktiskā pielietojuma viedokļa, neironu tīkli parasti nerisina problēmu kopumā, bet veic tikai daļu (un parasti nelielu) no problēmas kopējā risinājuma. Pārējā daļa, t.sk. procesa vadība, tiek atstāta tradicionāli veidotas sistēmas ziņā. Tālāk aprakstītas dažas svarīgākās ar neironu tīkliem risināmu problēmu klases. Aprakstā izmantotas ilustrācijas no [scherer97].

### 1.4.1. Klasifikācija

**Klasifikācija** (*classification*) ir ieejas paraugu grupēšana iepriekš noteiktās grupās. Klasifikācija būtu uzskatāma par funkciju aproksimācijas speciālgadījumu.

Klasifikāciju var veikt neironu tīkli, kas pieder pie modeļiem ar apmācību "ar skolotāju", piemēram, perceptrons, RBF tīkls.

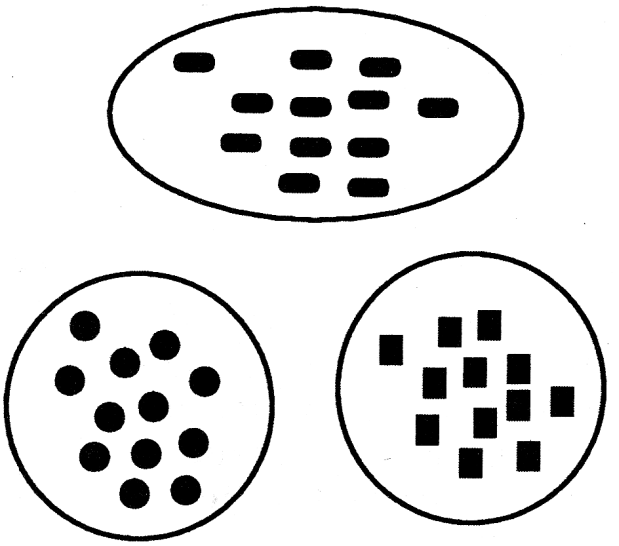



**Attēls 12. Paraugu klasifikācija**

### 1.4.2. Klāsterizācija

**Klāsterizācija** (*clustering*) jeb **kategorizācija** (*categorization*), līdzīgi kā klasifikācija, nodrošina ieejas paraugu grupēšana. Vienīgā atšķirība ir tā, ka kategorijām nav iepriekš noteikti kritēriji, kas tiek definēti apmācības laikā neironu tīkla ietvaros. Iepriekš noteikts ir tikai kategoriju skaits. Atkarībā no svaru inicializācijas vērtībām, kā arī neironu tīkla konfigurācijas parametriem, neironu tīkls katrā no darbināšanas reizēm var atgriezt citu dalījumu kategorijās.

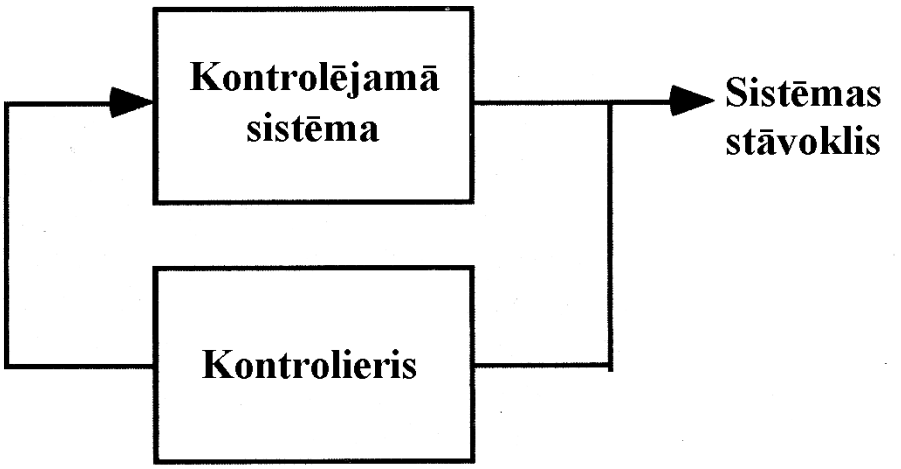
Klasifikāciju var veikt neironu tīkli, kas pieder pie modeļiem ar apmācību "bez skolotāja", piemēram, Kohonena tīkls.



**Attēls 13. Paraugu klāsterizācija**

### 1.4.3. Kontrole

**Kontrole** (*control*) nodrošina pastāvīgu sistēmas darbības uzraudzību, balstoties uz noteiktu parametru kopuma analīzi.

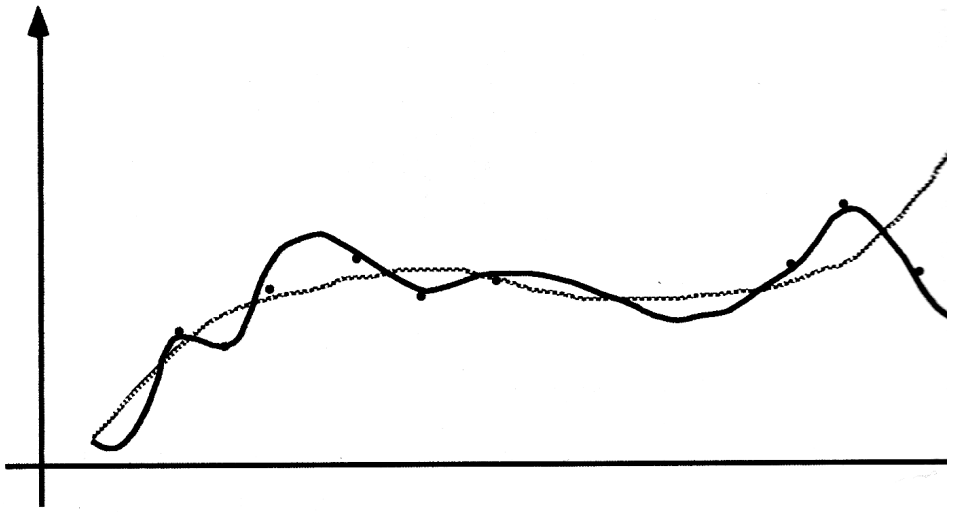



**Attēls 14. Sistēmas kontrole**

### 1.4.4. Funkcijas aproksimācija

**Funkcijas aproksimācija** (*function appoximation*) ir funkcijas tuvinājuma iegūšana, balstoties uz zināmiem funkcijas punktiem.

Aproksimāciju var veikt neironu tīkli, kas pieder pie modeļiem ar apmācību "ar skolotāju", piemēram, perceptrons, RBF tīkls.



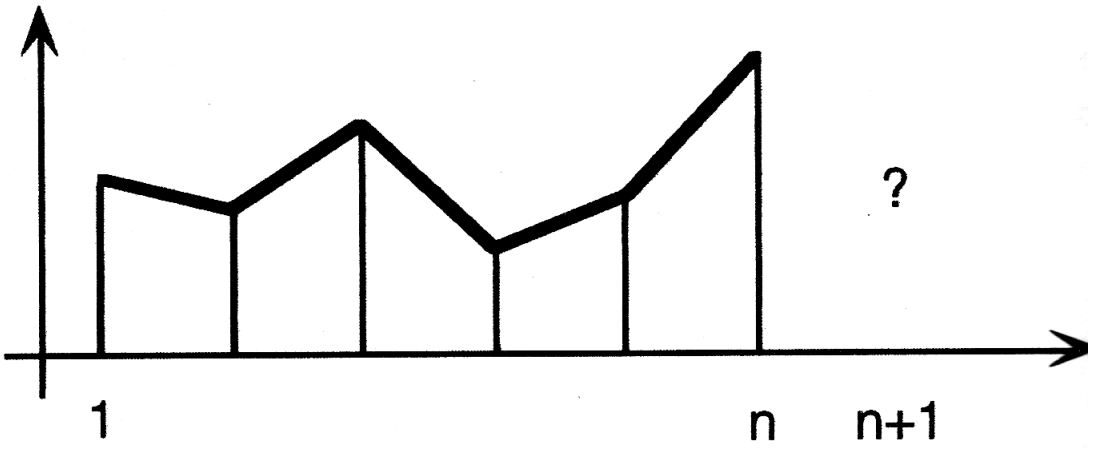
**Attēls 15. Funkcijas aproksimācija**

### 1.4.5. Prognoze

**Prognoze** (*forecast*) noteikta parametru kopuma noteikšana laika posmā *t+1*, balstoties uz noteiktu informāciju laika posmos *1..t*.

Prognoze ir faktiski neironu tīkla aproksimēšanas spējas specifisks pielietošanas gadījums, uzskatot, ka paraugi, kas tikuši izmantoti apmācībā un tiek izmantoti darbināšanai, ir izvērsti laikā.

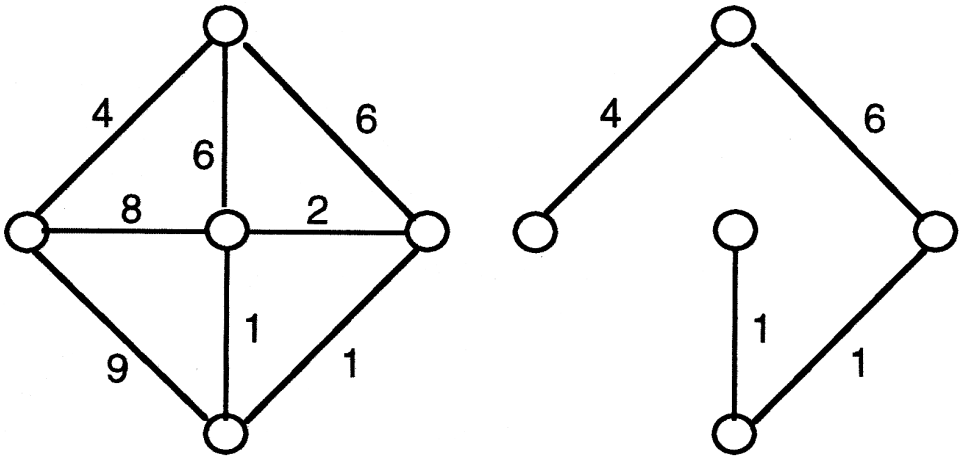
Prognozi parasti veic ar tīkliem ar apmācību ar skolotāju, un prognozes problēmas risināšanā ietvertā laika dimensija tiešā veidā neironu tīkla konstrukcijā neparādās.




**Attēls 16. Prognoze**

### 1.4.6. Optimizācija

**Optimizācija** (*optimization*) ir informācijas vai darbību apjoma samazināšana tādā veidā, lai sistēma turpinātu darboties līdzīgi kā pirms izmaiņas.




**Attēls 17. Optimizācija**

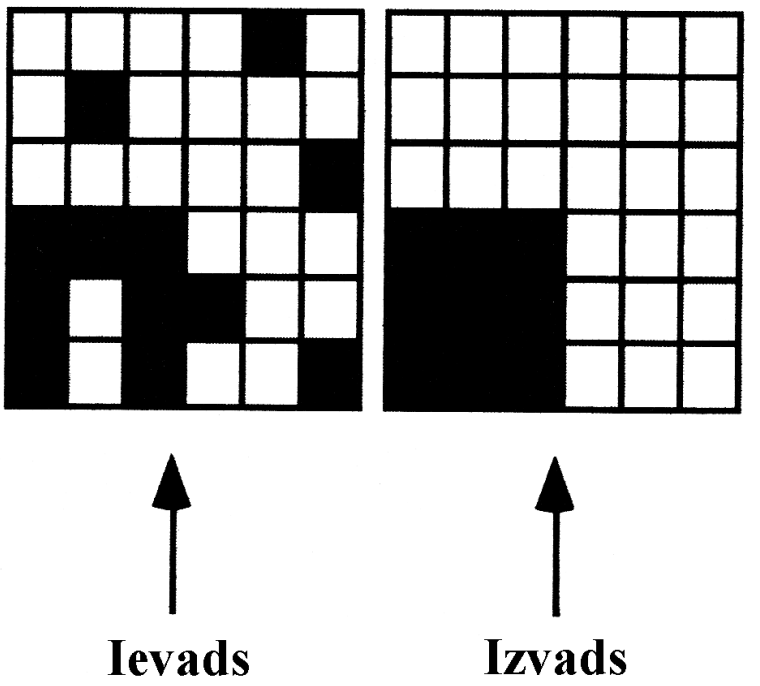
### 1.4.7. Atmiņa ar adresāciju pēc satura

**Atmiņa ar adresāciju pēc satura** (*content-adressable memory*) kaut kādā nozīmē darbojas līdzīgi klasifikācijai, bet ar šādām atšķirībām:

- Neironu tīkls tiek apmācīts tikai ar "pareiziem paraugiem", t.i. tiek apmācīts atcerēties paraugus, nevis tos kaut kādā veidā novērtēt.
- Parasti ievades un izvades apjoms ir līdzvērtīgi (klasifikācijas gadījumā iegūtā informācija par klasi (piemēram, numurs) parasti pēc apjoma ir daudz mazāka nekā pats ievadītais paraugs.

Piemēram, ja klasifikācija pēc parauga nosaka, kurai klasei pieder paraugs, tad pēc satura adresējamas atmiņas gadījumā, padodot neironu tīklam sabojātu paraugu, tiek atgriezts pareizs paraugs.

Atmiņu ar adresāciju pēc satura nodrošina, piemēram, Hopfīlda tīkls un *BAM* (*bidirectional assiciative memory*) tīkls.



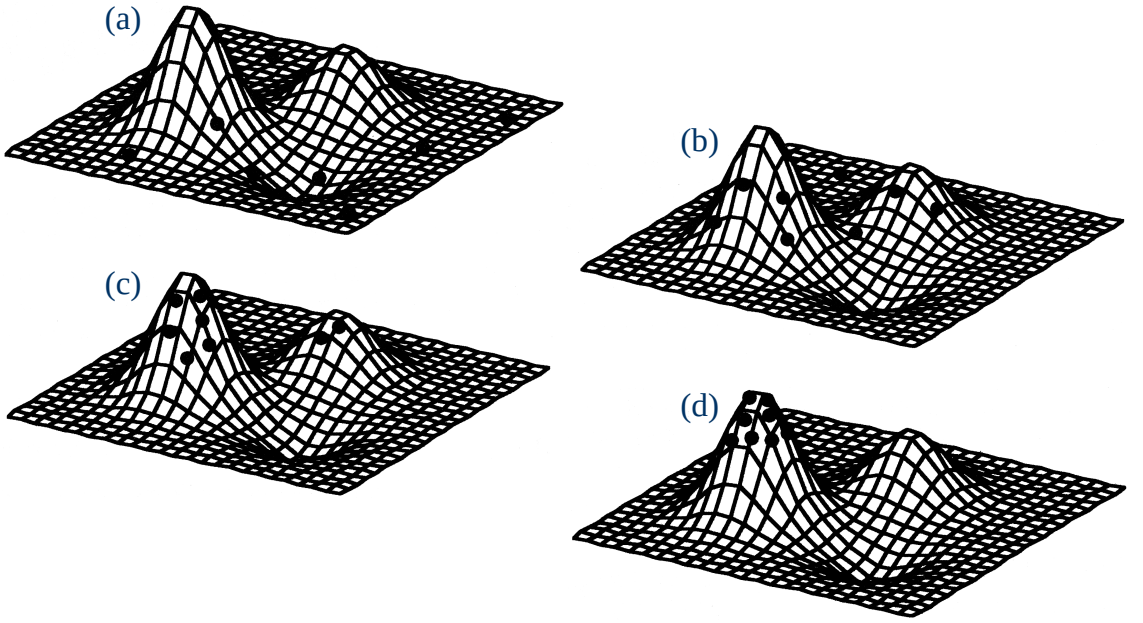

**Attēls 18. Atmiņa ar adresāciju pēc satura**

# 2. Ģenētiskie algoritmi un to saistība ar neironu tīkliem

## *2.1. Ģenētisko algoritmu darbības principi*

### 2.1.1. Vispārīgs apraksts

**Ģenētiskie algoritmi** (*genetic algorithms, GA*) ir problēmu risināšanas stratēģija, kas balstās uz bioloģiskās evolūcijas principiem (paaudžu nomaiņa, konkurence starp populācijas indivīdiem). Ģenētiskie algoritmi ir metožu kopums, lai optimizācijas un meklēšanas problēmām atrastu pareizus vai tuvinātus risinājumus.




**Attēls 19. Optimizācijas process, izmantojot ģenētiskos algoritmus [scherer97]**

Ģenētiskie algoritmi (ĢA) izmanto tādas no bioloģijas aizgūtas metodes un principus kā **iedzimtība** (*inheritance*), **mutācija** (*mutation*), **izlase** (*selection*) un **krustošana** (*crossover*).

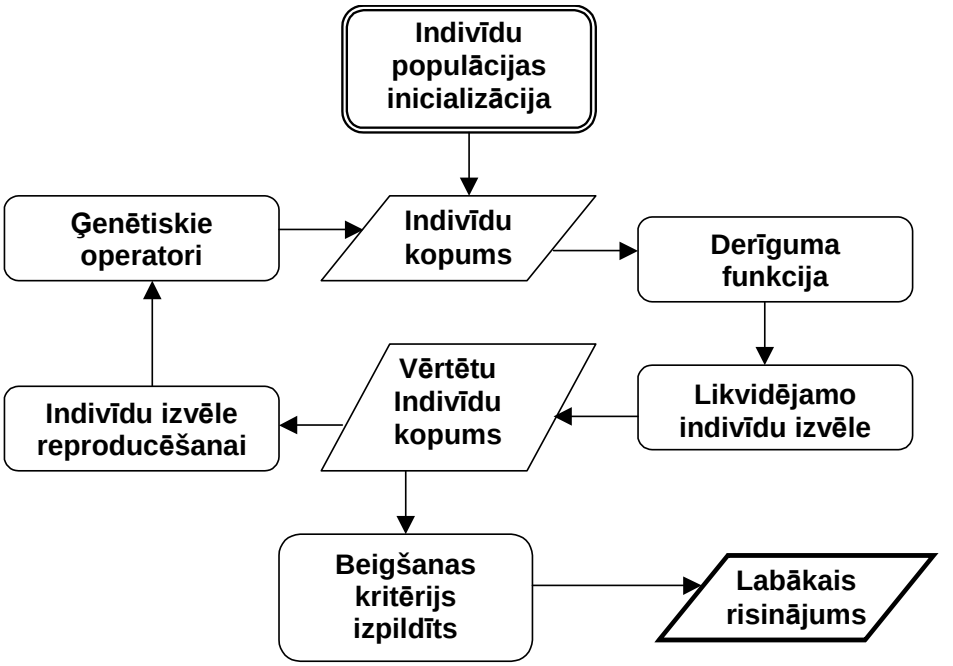
Ģenētiskie algoritmi tiek veidoti kā datorsimulācija, kurā kādas optimizācijas problēmas potenciālo risinājumu reprezentāciju kopums (saukts par **populāciju** (*population*)), attīstās labāku risinājumu virzienā. Potenciālos risinājumus, kas veido populāciju, sauc par **indivīdiem** (*individuals*), bet to abstraktās reprezentācijas, kas tiek izmantotas aprēķinos – par **hromosomām** (*chromosomes*).

Parasti evolucionārais process sākas ar populāciju no nejauši ģenerētiem indivīdiem un turpinās **paaudzi pēc paaudzes** (*generation by generation*).

Katras paaudzes laikā notiek šādas darbības:

- Visu populācijas indivīdu novērtēšana, izmantojot speciālu **derīguma funkciju** (*fitness function*).
- Indivīdu kopuma **izvēle** (*selection*) jaunas paaudzes veidošanai.
- Izvēlēto indivīdu apstrāde (**ģenētisko operatoru** (*genetic operators*) pielietošana), lai iegūtu jaunu paaudzi.

Ģenētiskā algoritma shēma parādīta attēlā (Attēls 20). Ģenētiskā algoritma darbības shēma nav precīzi noteikta, un tā var nedaudz atšķirties pēc soļu secības un dažādu soļu pielietošanas veida.




**Attēls 20. Ģenētiskā algoritma darbības vispārīgā shēma**

## 2.1.2. Derīguma funkcija

**Derīguma funkcija** (*fitness function*) ir metode indivīda novērtēšanai. Derīguma funkcijas dotais rezultāts ir pamatā indivīdu izvēlei, kas nodrošina ĢA darbību.

Derīguma funkcija netieši nosaka prasību pret problēmu klasi, kuru var risināt ar ĢA – tās risinājumus ir jāspēj novērtēt jebkurā gatavības pakāpē. Ja tas nav iespējams vai izdarāms pārāk sarežģīti, ģenētiskie algoritmi nebūtu īstā risināšanas stratēģija.
Tehniski derīguma funkciju bieži realizē kā **sodīšanas funkciju** (*penalty function*), t.i., mazāka vērtība nozīmē augstāku vērtējumu.

### 2.1.3. Izvēle

Ir daudz dažādu metožu indivīdu **izvēlei** (*selection*) nākošajām paaudzēm. Dažas no tām ir savstarpēji izslēdzošas, bet dažas ir lietojamas kombinēti.

**Elitisms** (*elitist selection*) nozīmē to, ka labākajiem (derīgākajiem) tiek garantēta to pārkopēšana nākamajā paaudzē. Vairums ĢA realizāciju tīrā veidā neizmanto elitismu, bet izmanto tā modificēto variantu.

**Derīgumam proporcionāla izvēle** (*fitness-proportionate selection*). Derīgākie indivīdi tiek izvēlēti ar lielāku varbūtību.

**Ruletes izvēle** (*roulette-wheel selection*). Derīgumam proporcionālās izvēles paveids, kas nosaka, ka indivīda izvēles iespēja ir proporcionāla lielumam, par kādu tā derīgums pārsniedz cita indivīda derīgumu. Konceptuāli ruletes izvēle līdzinās izspēlei ar ruletes riteni.

**Mērogojamā izvēle** (*scaling selection*). Tā kā ģenētiskā algoritma darbības laikā vidējais indivīdu derīgums palielinās, tad palielinās arī derīguma funkcijas diskriminējošais raksturs. Šī izvēles metode varētu būt derīga ģenētiskā procesa vēlākās fāzēs, kad visiem indivīdiem ir relatīvi augsts derīgums un tikai nelielas atšķirības starp tiem nosaka izvēli.

**Turnīra izvēle** (*tournament selection*). Populācija tiek sadalīta grupās, un indivīdi cīnās viens ar otru grupas ietvaros. Tikai viens indivīds no katras grupas tiek izvēlēts reproducēšanai.

**Kārtas numura izvēle** (*rank selection*). Visi indivīdi pēc vērtēšanas tiek sakārtoti rindā pēc to derīguma, un izvēle ir atkarīgi no šī sakārtojuma, nevis no derīgumu absolūtajām vērtībām vai to atšķirībām.

**Paaudžu nomaiņas izvēle** (*generational selection*). Tikai indivīdu pēcnācēji tiek iekļauti nākošajā paaudzē. Neviens indivīds nepāriet no paaudzes uz paaudzi.

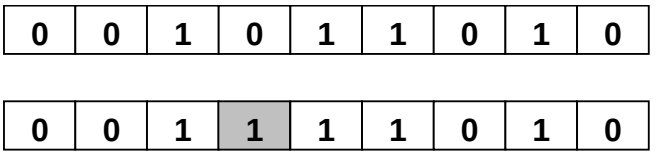
**Statiskā izvēle** (*steady-state selection*). Pēcnācēji daļēji aizvieto vecos indivīdus, tomēr daļa indivīdu pāriet no paaudzes uz paaudzi.

**Hierarhiskā izvēle** (*hierarchical selection*). Katras paaudzes laikā indivīdi iziet daudzpakāpju (vairāku kārtu) izsijāšanu. Zemāka līmeņa izvēles ir ātrākas un mazāk diskriminējošas, bet tie indivīdi, kas izdzīvo līdz augstākiem līmeņiem, tiek vērtēti aizvien bargāk. Šīs metodes priekšrocība ir kopējā skaitļošanas laika samazināšanās uz tā rēķina, ka lielākā daļa indivīdu tiek atsijāti zemākajos līmeņos, kur tiek veikti vienkāršāki un ātrāki aprēķini.

### 2.1.4. Ģenētiskie operatori

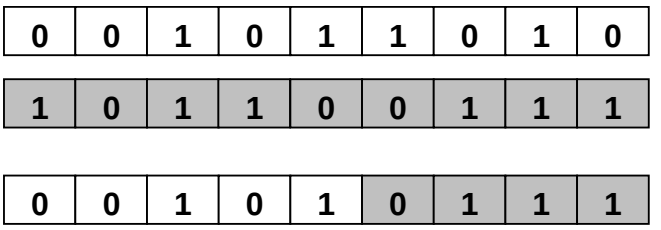
**Ģenētiskie operatori** (*genetic operators*) nodrošina jaunu (savādāku) indivīdu ģenerēšanu uz viena vai vairāku esošo indivīdu bāzes. Ģenētisko operatoru pielietošanas pakāpe un iesaistīto hromosomu skaits katrā algoritma darbības fāzē atkarīga no iepriekš noteiktiem koeficientiem, bet pašu operatoru darbība lielā mērā notiek pēc nejaušības principa. Pazīstamākie ģenētiskie operatori ir **mutācija** (*mutation*) un **krustošana** (*crossover*).

**Mutācija** ir vienkāršākais ģenētiskais operators, kas nosaka viena vai vairāku bitu (elementu) izmaiņu hromosomā (Attēls 21).




**Attēls 21. Mutācija**

**Krustošanā** ir iesaistīts vairāk nekā viens sākotnējais indivīds, kuru fragmenti veido pēcnācēju. Parastā forma ir **viena punkta krustošana** (Attēls 22), kas nodrošina, ka pēcnācējs būs savu vecāku garumā.




**Attēls 22. Viena punkta krustošana**

### 2.1.5. Beigšanas kritērijs

Beigšanas kritērijs parasti tiek definēts vienkārši, nosakot, ka jāizpildās vienam no diviem šādiem nosacījumiem:

1. Ir iegūts pietiekoši labs risinājums.
2. Ir pagājis pietiekoši daudz laika, vai algoritms ir darbojies pietiekoši daudz soļu.

## *2.2. Ģenētiskie algoritmi neironu tīklu izstrādē*

Ģenētisko algoritmu un neironu tīklu sadarbība parasti notiek šādās divās formās [scherer97]:

- svaru optimizācija (alternatīva apmācības algoritmam),

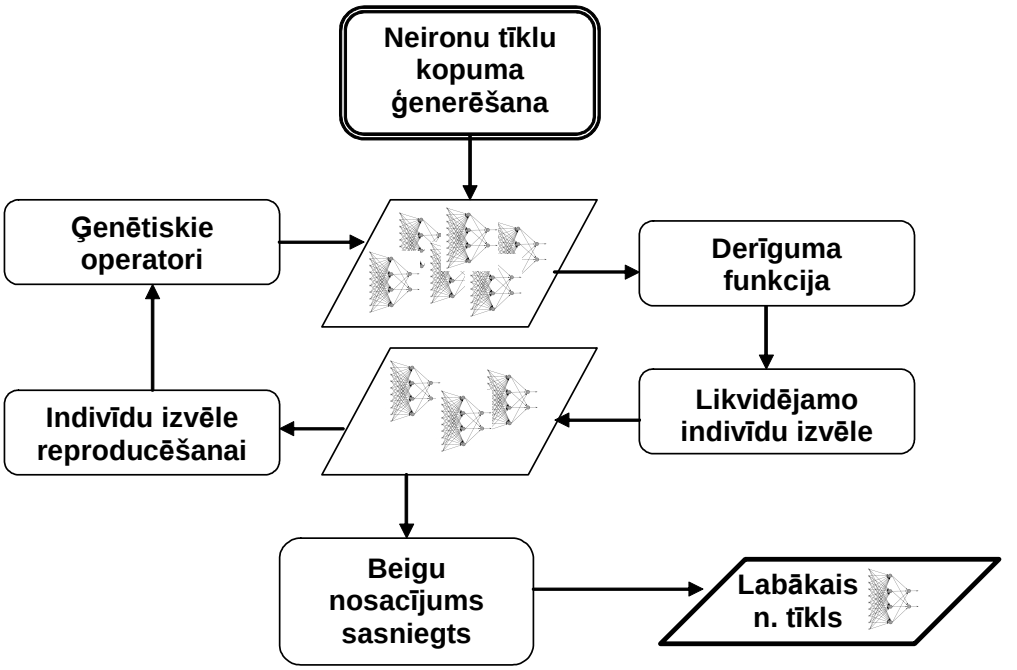
- topoloģijas un citu neironu tīkla modeļa parametru optimizācija.

### 2.2.1. Ģenētisko algoritmu pielietošana svaru vērtību optimizācijai

Viens no tipiskiem ģenētisko algoritmu pielietojumiem neironu tīklos ir to izmantošana tiešā veidā neironu tīklu apmācībā [scherer97],[azzini06],[kim05],[pardoe05]. Daudzu tradicionālo apmācības metožu trūkums ir iekļūšana lokālajos minimumos. Tas ir viens no iemesliem ģenētisko algoritmu pielietojumam neironu tīklu apmācībā, ka ĢA nodrošina izkļūšanu no lokālajiem minimumiem.

Šim nolūkam neironu tīkla svari varētu būt jāreprezentē binārā formā. Liela problēma ĢA lietojumam neironu tīklu svaru uzstādīšanā varētu būt tā, ka ĢA vajadzībām ir jānosaka maksimālā iespējamā svaru absolūtā vērtība, kas nav viegli izdarāms.

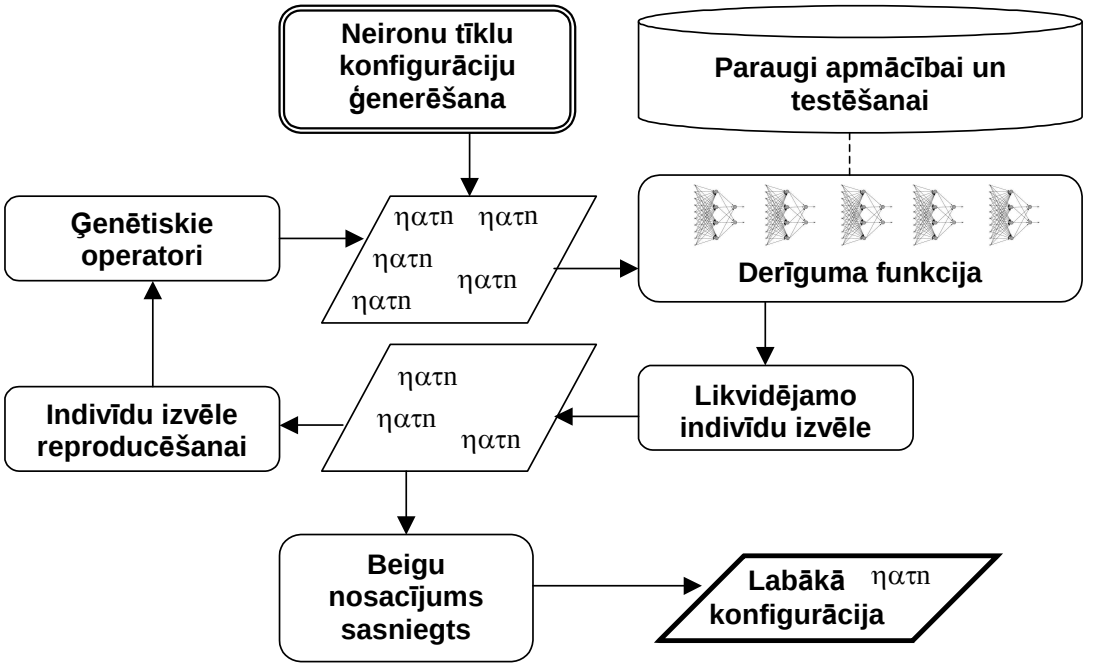
Ģenētisko algoritmu izmantošana ir viena no iespējām, bet ne tuvu ne vienīgā un ne vienmēr labākā. Neironu tīkla apmācība ar ĢA ir daudzkārt lēnāka un atmiņas resursus vairāk prasoša par apmācību ar *backpropagation* algoritmu, jo datora atmiņā vienlaicīgi jātur vairāki neironu tīkli.




**Attēls 23. Ģenētiskie algoritmi neironu tīklu apmācībā**

### 2.2.2. Ģenētisko algoritmu pielietošana neironu tīkla optimālās konfigurācijas noteikšanai

Ģenētiskie algoritmi būtu veiksmīgi izmantojami optimālākās neironu tīkla arhitektūras un citu parametru noskaidrošanai, lai iegūtu labāko neironu tīklu modeli dotās problēmu klases risināšanai. Izmantojot ĢA neironu tīklu arhitektūras optimizācijai, īpaši jāņem vērā potenciāli lielais aprēķinu apjoms, tādēļ jācenšas pēc iespējas samazināt arhitektūras un citu parametru variācijas, lai modeļa optimizācija notiktu saprātīgā laikā.




**Attēls 24. Ģenētiskie algoritmi neironu tīklu konfigurāciju optimizācijā**

# 3. Stundu saraksta sastādīšanas problēma

**Stundu saraksta sastādīšanas** (*timetabling*; *scheduling*) **problēma** (SSP) pieder pie augstas dimensionalitātes kombinatoriskām optimizācijas problēmām un pēc sarežģītības tiek klasificēta kā *NP-hard* [arous99]. Stundu saraksta sastādīšanas problēma ir formulējama kā **ierobežojumu izpildīšanas problēma** (*constraint satisfaction problem, CSP*).

## *3.1. Stundu saraksta sastādīšanas problēma – ierobežojumu izpildīšanas problēma*

### 3.1.1. Ierobežojumu izpildīšanās problēmas definēšana

**Ierobežojumu izpildīšanas problēma** (*constraint satisfaction problem, CSP*) [russell02] tiek definēta kā **mainīgo** (*variables*) kopa $X_1, X_2, ...., X_n$ un **ierobežojumu** (*constraints*) kopa $C_1, C_2, ..., C_m$. Katram mainīgajam $X_i$ ir netukšs definīcijas apgabals. Katrs ierobežojums $C_i$ ietver sevī mainīgo apakškopu un šīs apakškopas mainīgo vērtību pieļaujamās kombinācijas. Par *CSP* **stāvokli** (*state*) sauc pilnīgu vai daļēju mainīgo aizpildījumu ar vērtībām ($X_i=v_i, X_j=v_j, ...$). Mainīgo aizpildījumu, kas nav pretrunā ne ar vienu ierobežojumu, sauc par **pieļaujamu** (*consistent, legal*). Par **pilnīgu** (*complete assignment*) sauc tādu aizpildījumu, kurā ietilpst visi mainīgie. Par **risinājumu** (*solution*) sauc tādu pilnīgu aizpildījumu, kurš atbilst visiem ierobežojumiem.

Stundu saraksta sastādīšanas problēmas (SSP) gadījumā *CSP* mainīgie ir nodarbības, kas ietver sevī klases un skolotājus, kā arī mācāmos priekšmetus, bet piešķiramās vērtības ir laika periodi un telpas. SSP ierobežojumi ir pirmkārt iepriekš definēto nodarbību struktūra, kur skolotāji jau ir piesaistīti klasei un priekšmetam, kā arī dažādu resursu savstarpējā atbilstība, piemēram, telpas atbilstība mācību priekšmetam.

Dažās ierobežojumu izpildīšanas problēmās nepieciešams tāds risinājums, kurā ir maksimizēta noteikta **mērķa funkcija** (*objective function*). Problēmas, kurās blakus ierobežojumu ievērošanai jāveic arī zināma optimizācija (kā tas ir, piemēram, gadījumā ar mērķa funkcijas maksimizēšanu), mēdz saukt par **optimizācijas problēmām**. Stundu saraksta sastādīšanas problēma ir tieši šāda tipa problēma.

### 3.1.2. Ierobežojumu izpildīšanās problēmu risināšanas metodes

Ierobežojumu izpildīšanās problēmu vienkāršākajā gadījumā var formulēt kā meklēšanas (*search*) problēmu un tās risināšanai lietot jebkuru zināmo meklēšanas algoritmu, piemēram, **meklēšanu dziļumā** (*depth-first search*) vai **meklēšanu plašumā**

(*breadth-first serach*). Tomēr CSP tipa problēmām piemīt vēl viena īpašība – **komutativitāte** (*commutativity*). Problēma ir komutatīva, ja jebkura darbību kopuma izpildes secībai šīs problēmas risināšanai nav ietekmes uz galarezultātu. Parastie meklēšanas algoritmi ignorē šo būtisko īpašību.

**Meklēšana ar atpakaļkāpšanos** (*backtracking search*) ir meklēšanas dziļumā (*depth-first search*) paplašinājums, kad atpakaļkāpšanās mehānisms tiek iedarbināts gadījumā, ja mainīgajam vairs nav pieļaujamu vērtību, kuras piešķirt.

Lai uzlabotu meklēšanas ar atpakaļkāpšanos efektivitāti, tiek lietotas dažādas papildus metodes un heiristikas. Lūk, dažas no tām:

1. **Mazākais atlikušo vērtību skaits** (*minimum remaining values*, *MRV*). Kā nākošo meklēšanas procesa laikā izvēlas tādu mainīgo, kuram ir vismazākā izvēle starp iespējamām vērtībām.
2. **Iesaistīšanās pakāpes heiristika** (*degree heuristic*). Tiek izvēlēts tāds mainīgais, kurš ir iesaistīts visvairāk ierobežojumos, kas skar pārējos neaizpildītos mainīgos.
3. **Pārbaude uz priekšu** (*forward checking*). Kad vien mainīgais *X* ir aizpildīts, tiek aplūkoti visi neaizpildītie mainīgie *Y*, kas ir saistīti ar *X* caur kādu ierobežojumu, un no *Y* definīcijas apgabala tiek izmestas visas vērtības, kas ir pretrunā ar mainīgajam *X* piešķirto vērtību.
4. **Konflikta orientēta lēkšana atpakaļ** (*conflict-directed backjumping*). Aizvieto hronoloģisko atpakaļkāpšanos ar metodi, kas izmanto speciāli veidotas konfliktu kopas.

**Mazāko konfliktu** (*min-conflicts*) heiristika nosaka tādas vērtības izvēli mainīgajam, kas vismazāk konfliktē ar citiem mainīgajiem. Mazāko konfliktu heiristika ir ļoti efektīva daudzu ierobežojumu izpildīšanās problēmu risināšanā, īpaši, ja ir pietiekoši labi noteikts sākuma stāvoklis.

Heiristiku pielietošana ir ļoti atkarīgas no konkrētas problēmas specifikas, un bieži lieto vairāku **heiristiku kombinācijas**. [tam03] apraksta universitātes stundu saraksta efektīvu veidošanu, izmantojot mazāko konfliktu heiristiku un skatīšanās uz priekšu (*look-forward*) heiristiku.

## *3.2. Skolu stundu sarakstu sastādīšanas specifika*

Vispārīgas stundu saraksta sastādīšanas problēmas (SSP) gadījumā notikumi ir jāizkārto pa laika periodiem (*time slots*) tā, lai tajos izmantoto resursu pielietojums atbilstu iepriekš uzstādītiem ierobežojumiem, kā arī pēc iespējas tiktu optimizēta papildus nosacījumu izpilde.

Skolu stundu sarakstu sastādīšanas (*school timetabling*) gadījumā, notikumi, kas būtu jāizkārto pa laika periodiem ir nodarbības, kur kāda noteikta skolēnu grupa noteiktu stundu skaitu apgūst kādu mācību priekšmetu. Resursi, uz kuriem attiecas ierobežojumi un papildus nosacījumi, ir skolēnu grupas, skolotāji un parasti arī telpas.

Runājot par stundu saraksta sastādīšanas ierobežojumiem un citiem nosacījumiem, parasti tiek izdalīti t.s. **obligātie un neobligātie noteikumi** (*hard and soft constraints*) [fernandes99],[tam03],[santos05].

Obligātie noteikumi:

- skolēns var vienlaicīgi piedalīties tikai vienā nodarbībā,
- skolotājs var vienlaicīgi piedalīties tikai vienā nodarbībā,
- vienā telpā vienlaicīgi var notikt tikai viena nodarbība,
- telpai, kurā notiek nodarbība jābūt pietiekoši lielai,
- skolēns, skolotājs vai telpa nedrīkst tikt ieplānota tādā laikā, kad ir definēts ierobežojums uz kādu no tiem.

Daži neobligātie noteikumi:

- ieplānot pēc iespējas mazāk "logu" – gan skolēniem, gan skolotājiem,
- līdzsvarot nodarbību izkārtojumu pa nedēļas dienām,
- ievērot ierobežojumus uz nodarbību skaitu vienā dienā – gan skolēniem, gan skolotājiem,
- ievērot ierobežojumus uz pēc kārtas esošu nodarbību skaitu – gan skolēniem, gan skolotājiem,
- pēc iespējas ievērot skolotāju vēlmes,
- līdzsvarot skolēnu slodzi tematiski – lai nebūtu pēc kārtas daudz grūto priekšmetu,
- izvēles priekšmeti var potenciāli veidot "logus", tādēļ tos labāk plānot dienas sākumā vai beigās,
- dažiem priekšmetiem nepieciešamas dubultstundas,
- ir iespējama viena diena nedēļā, kad skolotājs ir pilnīgi brīvs, ja viņa slodze nav pilna.

Vispārizglītojošo skolu gadījumā bieži var izveidot tādu stundu sarakstu, kas atbilst obligātajiem noteikumiem un diezgan lielā mērā arī neobligātajiem, un tomēr ir praktiski nelietojams. Tas tādēļ, ka skolu stundu sarakstu sastādīšanas specifika ir izpildāmajos noteikumos salīdzinoši lielā mērā ietvertais cilvēciskais faktors: ne visus

nosacījumus var pietiekoši labi stingri definēt un pat apzināt, bez tam nav vienkārši noskaidrot katra nosacījuma svarīguma pakāpi. Pie nosacījumiem, kuros lielā mērā ietverts cilvēciskais faktors, var pieskaitīt, piemēram, skolēnu slodzes līdzsvarošanu no tematiskā viedokļa. Grūti formulējamu prasību esamība stundu saraksta sastādīšanas problēmā dod pamatu neironu tīklu izmantošanai šāda veida problēmu risināšanai.

## 3.3. Ģenētiskie algoritmi stundu saraksta sastādīšanas problēmas risināšanā

Optimizācijas problēmu risināšanai, kāda ir arī SSP, plaši tiek lietotie ģenētiskie algoritmi, un tos var uzskatīt par tipisku stundu sarakstu sastādīšanas problēmu risināšanas veidu. Heiristiskās meklēšanas metodes SSP risināšanai ir grūti piemērojamas arī tādēļ, ka SSP ir piesātināta ar lokālajiem minimumiem. Šī nodaļa apraksta dažādus aspektus un pielietojuma metodes stundu saraksta sastādīšanas problēmas risināšanai ar ĢA, un tā galvenokārt balstās uz [arous99] un [fernandes99] piedāvāto pieeju salīdzinājumu.

ĢA lietošana nosaka sešu šādu komponenšu definēšanu:

- hromosomu reprezentācija,
- ģenētiskie operatori, kas nosaka reprodukcijas funkciju,
- izvēles funkcija,
- sākotnējās populācijas ģenerēšana,
- beigšanas kritērijs (sīkāks apraksts nodaļā 2.1.5),
- derīguma funkcija.

### 3.3.1. Hromosomu reprezentācija

#### 3.3.1.1. Stundu saraksta elementi

Stundu saraksta pamatā ir šādi elementi:

1. **Periodi** (*time slots*). Diskrēti laika intervāli nodarbību plānošanai.
2. **Telpas**.
3. **Skolotāji**.
4. **Klases**. Klases var tikt dalītas grupās, bet padara sarežģītāku plānošanu. Klasei ir piekārtoti kursi. Klasei var tikt piekārtota noklusētā telpa.

5. **Kursi**. Katrs kurss atbilst noteiktai klasei, tam ir piekārtots noteikts priekšmets un periodu skaits, kā arī tips (piemēram, parastā nodarbība vai laboratorijas darbs). Kad kurss ir ieplānots, tad tam tiek piesaistīts vēl arī sākotnējais periods un telpa. Atkarībā no stundu saraksta tipa kursa garums var tikt definēts arī minūtēs, nevis periodos.
6. **Noteikumi**. Katrai telpai, skolotājam, klasei vai kursam var tikt piesaistīti noteikumi – piemēram, pieejamo periodu kopums un to prioritātes elementam vai iespējamo telpu kopums kursam vai klasei.

#### 3.3.1.2. Stundu saraksta elementu izvietošana hromosomā

**Tiešais variants**.

Bieži pielietots variants SSP risināšanā ar ĢA ir tiešas hromosomu reprezentācijas lietošana komplektā ar sodīšanas (*penalty*) funkciju. Stundu sarakstu reprezentē matricas veida hromosoma, kurā genotipu (elementus) veido 3 tipu alēles (gēnu variācijas) – skolotāja, kursa un telpas alēles. Dažos gadījumos genotips ir jānosaka biežāk nekā reizi nedēļā. Kolonnas pozīcija genotipā varētu noteikt atbilstošo laika periodu, bet rindas pozīcija – atbilstošo klasi.

Tiešā varianta priekšrocība ir efektivitāte, bet trūkums – būtiskas vienkāršošanas nepieciešamība vairāku reālu problēmu gadījumā.

**Dalītā metode**.

Informācija par kursiem tiek dalīta 2 daļās. Informācijas daļa, kas ir konstanta visiem risinājumiem, tiek glabāta atsevišķi, bet atlikusī daļa – hromosomās. Katra hromosoma tādējādi reprezentē vienu risinājumu noteiktai problēmai.

Konstantā informācija par kursu (nodarbību):

- klase vai klases,
- skolotājs vai skolotāji,
- katras nodarbības daļas garums,
- daļu skaits,
- tips.

Hromosomās glabātā informācija par nodarbību:

- telpa vai telpas,
- katras daļas sākuma periods (*starting time slots*).

Nodarbību sākuma periodi ir galvenā informāciju, kas tiks ģenerētā algoritma darbības laikā, tāpēc hromosomā ir jābūt gēniem, kas to varētu reprezentēt. Tādējādi hromosoma tiek veidota kā gēnu virkne, kur gēni attiecas uz skolotājiem, telpām, un nodarbības sākuma periodiem.

### 3.3.2. Ģenētiskie operatori

Tipiska metode hromosomu izvēlei, kas tiks iesaistītas nākamās paaudzes veidošanā, ir ruletes metode (*roulette-wheel selection*), kas nodrošina hromosomu izvēli proporcionāli to derīgumam.

Ģenētisko operatoru pielietošanas iespējas ir atkarīgas no hromosomu reprezentācijas, kas savukārt ir atkarīga arī no reprezentējamās informācijas sarežģītības. Pie noteiktas reprezentācijas sarežģītības ir iespējams, ka vienīgais pielietojamais ģenētiskais operators ir mutācija, jo var nebūt iespēju sapratīgi sakombinēt divus dažādus sarakstus. [arous99],[fernandes99],[colorni98] aprakstītajās pieejās tiek lietota gan mutācija, gan krustošana.

Ir divi gatavības pakāpes veidi, kādus ģenētiskie operatori spēj nodrošināt jaunizveidotajiem stundu sarakstiem:

1. **Pabeigtais variants.** Ģenētiskais operators pats nodrošina obligātajiem noteikumiem atbilstoša saraksta ģenerēšanu, un nav nepieciešami papildus mehānismi integritātes uzturēšanai. Šāda piegājiena trūkums ir iespējamās problēmas ar ātrdarbību, ģenētiskajam operatoram atsevišķos gadījumos ilgstoši nespējot iegūt korektu risinājumu.
2. **Nepabeigtais variants.** Ģenētiskais operators veic zema līmeņa tehniskas operācijas, nenodrošinot izveidotā saraksta integritāti. Tādējādi nepieciešama speciāla remontēšanas funkcija (*repair function*), kas apstrādā šādi iegūtus nekorektos risinājumus. Vienkāršākais variants ir neveiksmīgi uzģenerēta risinājuma gadījumā to aizvietot ar viņa vecāku. Neskatoties uz pirmajā acumirklī ieraugāmo trūkumu, šāda veida piegājiens nodrošina būtiski augstāku izpildes efektivitāti. Savukārt [colorni98] nosaka ļoti augstu sodu (*penalty*) katram integritātes pārkāpumam, lai stundu saraksts kaut ar vienu pārkāpumu nevarētu vairs sacensties ar nevienu (vienalga cik sliktu) sarakstu bez pārkāpumiem.

### 3.3.3. Izvēles funkcija

Izvēles funkcija (nodaļa 2.1.3) nosaka, kuri indivīdi tiks izmesti no populācijas, pārejot uz nākamo paaudzi. Labākā hromosomu proporcija pārejai uz nākošo paaudzi katras problēmas gadījumā būtu jānosaka eksperimentāli. Papildus izvēlei parasti tiek

piemērots elitisma operators, kas nodrošina labāko risinājumu atstāšanu populācijā uz nākamo paaudzi. Elitisma pielietošana SSP gadījumā ir ļoti nozīmīga. Eksperimenti rāda, ka, neizmantojot elitisma operatoru, var pat neizdoties iegūt labāku risinājumu par sākotnējā populācijā esošajiem.

### 3.3.4. Sākotnējās populācijas ģenerēšana

Inicializācijas procedūra pēc nejaušības principa piekārto katrai nodarbībai noteiktus laika periodus un telpas. Tāpat kā ģenētisko operatoru gadījumā (nodaļa 3.3.2) sākotējās populācijas ģenerēšana iespējama pabeigtajā variantā, kad visi indivīdi garantēti atbilst obligātajiem noteikumiem, vai nepabeigtajā variantā, kad pēc inicializācijas jāpiemēro noteikta veida remontēšanas funkcija.

### 3.3.5. Derīguma funkcija

Derīguma funkcija nosaka stundu saraksta derīgumu, no kā izriet divas darbības ģenētiskā algoritma darbināšanas procesā:

- vecāku izvēle ģenētiskajos operatoros,
- stundu sarakstu izmešana, pārejot uz nākamo paaudzi.

Parasti praktisku iemeslu dēļ par derīguma funkciju tiek lietota t.s. sodīšanas funkcija, kam mazāka vērtība nozīmē augstāku vērtējumu.

[arous99] piedāvā šādus mērķus, uz kuru pamata veidot derīguma funkciju:

- maksimizēt skolotāju vēlmju piepildīšanu pēc iepriekš noteikta vēlmju (noteikumu) saraksta.
- minimizēt brīvās stundas (jeb “logus”) gan audzēkņiem, gan skolotājiem.

[fernandes99] izmanto šādus principus derīguma funkcijas konstruēšanā:

- katram ierobežojumam sākotnēji tiek piekārtots svars, kas nosaka šī ierobežojuma nozīmību,
- stundu saraksta vērtējums tiek definēts kā ierobežojumu pārkāpumu svērtā summa pēc ierobežojumu noteiktā nozīmīguma,
- maksimālajā periodu skaita pārkāpšanai vienas dienas ietvaros un maksimālā pēc kārtas esošo periodu skaita pārkāpšanai tiek piemērots eksponenciāls faktors,
- lai nodrošinātu efektīvu darbību, derīguma funkcija pielāgojas populācijas kvalitātes izmaiņām un ir atkarīga no populācijas labākās hromosomas, kā arī visas populācijas vidējās vērtības.

## *3.4. Neironu tīkli plānošanas problēmu risināšanā*

Pieejamā zinātniskā literatūra parāda, ka neironu tīklu izmantošana dažādu plānošanas problēmu (t.sk. stundu saraksta sastādīšanas problēmu) risināšanā nav ne tuvu tik plaša kā ģenētisko algoritmu izmantošana. Galvenais iemesls tam ir tāds, ka pašreizējā neironu tīklu tehnoloģijas attīstības pakāpē ar neironu tīkliem praktiski nevar risināt šo problēmu kopumā, bet iespējams tikai atbalstīt atsevišķu apakšproblēmu risināšanu.

[alifantis01] aprakstīti pētījumi par neironu tīklu izmantošanu darbu plānošanas (*job shop-scheduling*) problēmas risināšanā. Darbā aprakstītā metodoloģija piedāvā izveidot lēmumus atbalstošu neironu tīklu (*scheduling advisor*). Neironu tīkls tiek apmācīts ar optimāliem plānošanas risinājumiem, kas iegūti, izmantojot simulāciju.

# 4. Neironu tīklu vispārināšanas spēja un tās nodrošināšana

## *4.1. Neironu tīkls kā matemātisks modelis*

Neironu tīkls (pašreizējā tehnoloģijas attīstības līmenī) parasti izpaužas kā skaitļošanas sistēma, kas noteiktā veidā atbild (novērtē, atpazīst utt.) uz ieejas paraugiem. No matemātikas viedokļa neironu tīkls modelē kādu procesu vai objektu, kuru reprezentē noteiktas formas paraugi.

**Matemātisks modelis** (arī neironu tīkls) ir kāda procesa vai objekta vispārīgs raksturojums, kas nodrošina relatīvi vienkāršu manipulāciju ar mainīgajiem, lai noteiktu šī procesa vai objekta uzvedību dažādās situācijās.

Matemātisko modeļu sastāvā ietilpst trīs šādas lielumu grupas [frolova99]:

- Lielumi $\boldsymbol{Y}=y_1,y_2,...y_m$, kas raksturo oriģinālo procesu vai objektu (atbilst neironu tīkla izejas vērtībām).
- Lielumi $\boldsymbol{X}=x_1,x_2,...x_n$, kas raksturo ārējos apstākļus (atbilst neironu tīkla ieejas vērtībām).
- Oriģinālā procesa vai objekta iekšējo parametru kopa $\boldsymbol{A}$ (atbilst neironu tīkla svariem un citiem iekšējiem parametriem). Daži no šīs kopas elementiem var būt zināmi, bet daži nezināmi.

Šo trīs grupu lielumus no modeļa izmantošanas viedokļa var dalīt divās daļās [frolova99]:

- Ārējo apstākļu raksturojumus $\boldsymbol{X}$ un kopas $\boldsymbol{A}$ zināmos elementus sauc par **eksogēniem parametriem**, un tos nosaka neatkarīgi no paša modeļa.
- Meklējamos lielumus $\boldsymbol{Y}$ un kopas $\boldsymbol{A}$ nezināmos elementus sauc par **endogēniem parametriem**, un tie ir nosakāmi, atrisinot vai izpētot modeli.

Eksogēnie parametri veido matemātiskā modeļa ieeju, bet endogēnie – izeju.

Ņemot vērā to, ka kopas $\boldsymbol{A}$ elementi (atbilst neironu tīkla svariem un citiem iekšējiem parametriem) var būt gan eksogēnie, gan endogēnie parametri, tad par matemātisku modeli ir uzskatāms gan neapmācīts, gan apmācīts neironu tīkls.

## *4.2. Neironu tīklu vispārināšanas spēja*

### 4.2.1. Vispārināšanas spējas nodrošināšana

Neironu tīkla konstruēšanas pamatmērķis ir iegūt tādu tīklu, kas pēc iespējas labāk atpazītu (novērtētu utt.) ieejas paraugus. To nodrošina tādas neironu tīkla īpašības, kā iegaumēšana un vispārināšana. Neironu tīkla uzdevums būtu iegūt līdzsvaru starp šīm divām īpašībām [fausett94]:

- **Iegaumēšana** (*memorization*) ir neironu tīkla spēja precīzi atpazīt paraugus, ar kuriem tas ir ticis apmācīts.
- **Vispārināšana** (*generalization*) ir neironu tīkla spēja saprātīgi reaģēt uz tādiem ieejas paraugiem, kuri ir līdzīgi, bet ne identiski tiem, ar kuriem neironu tīkls ir ticis apmācīts.

Viena no neironu tīklu lielākajām problēmām apmācības laikā ir pārmērīga pielāgošanās.

**Pārmērīga pielāgošanās** (*overfitting*) ir parādība, kad neironu tīkla apmācības rezultātā tas tik ļoti pielāgojas apmācības piemēriem, ka vispārināšanu veic slikti.

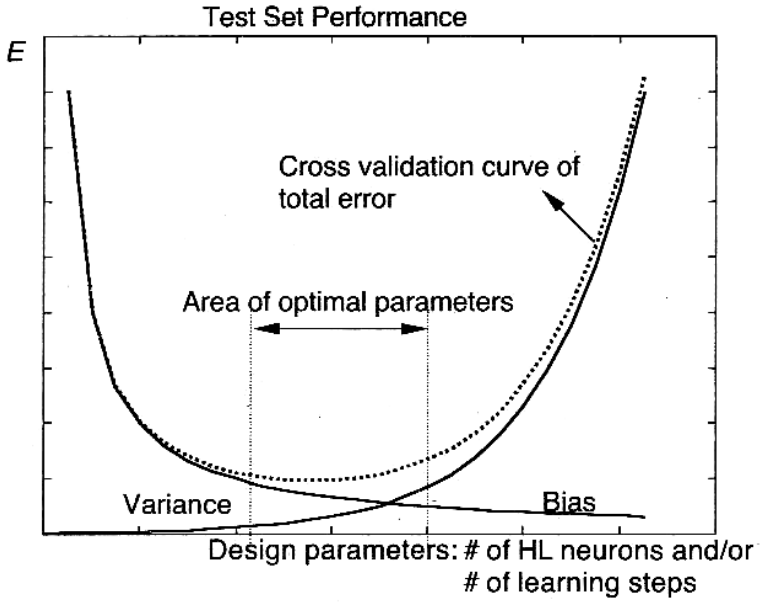
Runājot par neironu tīkla apmācīšanu un vispārināšanas spēju, bieži tiek pieminēts tāds jēdziens kā **tendenciozitātes-dispersijas dilemma** (*bias-variance dilemma*), kas nāk no matemātiskās statistikas. Tendenciozitāte un dispersija ir apmācāmu modeļu divas savstarpēji konkurējošas un pēc būtības pretējas, bet abas negatīvas no vispārināšanas viedokļa, īpašības:

1. **Tendenciozitāte** (*bias*) ir neironu tīkla nespēja veikt mērķa funkcijas aproksimāciju, kuras pamatā ir neironu tīkla nepietiekamā sarežģītība (piemēram, pārāk maz slēpto neironu). Tendenciozitāte būtībā norāda uz neironu tīkla spēju (respektīvi, nespēju) pielāgoties apmācības datiem, nevis uz vispārināšanas līmeni. Tendenciozitāti var uztvert kā aproksimācijas kļūdu [haykin99].
2. **Dispersija** (*variance*) ir neironu tīkla apmācības efektivitātes novirze, mainot apmācības piemērus. Dispersija norāda uz neironu tīkla vispārināšanas spēju, t.i., vai neironu tīkls pielāgojas apmācības piemēriem, neņemot vērā apmācības datu specifiku. Dispersija savā ziņā reprezentē apmācības piemēru neatbilstību mērķa funkcijai un to, cik lielā mērā šī neatbilstība pāriet uz neironu tīklu. Tādējādi dispersiju var uztvert kā novērtējuma kļūdu [haykin99].

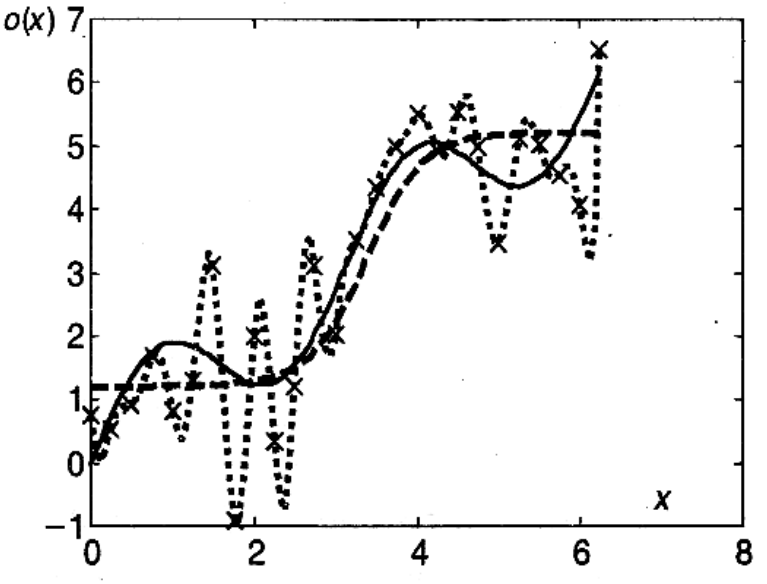
**Aproksimācijas kļūda** (*approximation error*) ir neironu tīkla dotā kļūda uz paraugiem, uz kuriem tas ticis apmācīts.

**Novērtējuma kļūda** (*estimation error*) ir neironu tīkla dotā kļūda uz funkciju, kuru tam vajadzētu modelēt. Praktiski šāda funkcija vispārīgā gadījumā ir nezināma, tādēļ

novērtējuma kļūdu nosaka kā kļūdu uz paraugiem, uz kuriem neironu tīkls nav ticis apmācīts.




**Attēls 25. Tendenciozitātes (bias) un dispersijas (variance) tipiska shēma atkarībai no neironu tīkla konfigurācijas (neironu skaita slēptajā slānī un apmācības procesa soļu skaita) [kecman02]**



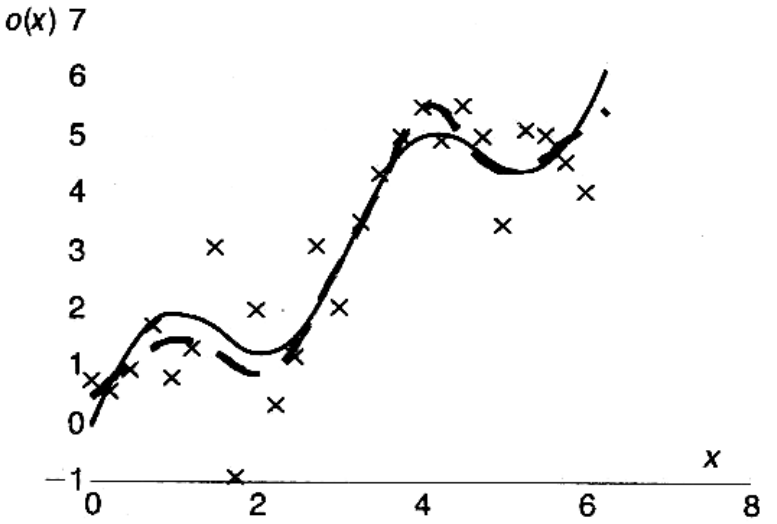

**Attēls 26. Tendenciozitātes un dispersijas pakāpes demonstrācija, aproksimējot līkni (26 apmācības piemēri, apzīmēti ar krustiņiem, trokšņa līmenis 25%). Atbilstošā funkcija *f*(x)=x+sin(2x) (nepārtraukta līnija). Aproksimācija ar tīklu, kurā ir tikai viens neirons slēptajā slānī (raustīta līnija) – augsta tendenciozitāte, zema dispersija. Aproksimācija ar tīklu, kurā ir 26 neironi slēptajā slānī (punktētā līnija) – zema tendenciozitāte, augsta dispersija [kecman02]**

Tā kā abas šīs īpašības ir negatīvas attiecībā pret neironu tīklu, bet vienas īpašības ietekmes samazināšana noved pie otras īpašības ietekmes palielināšanās, būtu jāatrod kompromiss starp abām šīm īpašībām.

Lai nodrošinātu zemāku tendenciozitāti, būtu jāpalielina neironu tīkla (konkrētāk slēpto slāņu) apjoms.

Lai nodrošinātu zemāku dispersiju, bet augstāku vispārināšanās spēju un tādējādi izvairītos no pārmērīgas pielāgošanās (*overfitting*), būtu izmantojamas vairākas metodes, piemēram:

- korekta modeļa izvēle un konfigurēšana (t.sk., pietiekoši mazs neironu tīkla apjoms),
- apmācībai pieejamās paraugu kopas palielināšana,
- apmācības procesa ātrāka pārtraukšana (nodaļa 4.2.3),
- vairāku neironu tīklu kombinēšana.




**Attēls 27. Tendenciozitātes un dispersijas pakāpes saprātīga kompromisa demonstrācija, aproksimējot līkni (26 apmācības piemēri, apzīmēti ar krustiņiem, trokšņa līmenis 25%). Atbilstošā funkcija *f*(x)=x+sin(2x) (nepārtraukta līnija). Aproksimācija ar tīklu, kurā ir 8 neironi slēptajā slānī (raustīta līnija) – sapratīga tendenciozitāte, saprātīga dispersija [kecman02]**

### 4.2.2. Vispārināšanas spējas novērtēšana, izmantojot krustenisko validāciju

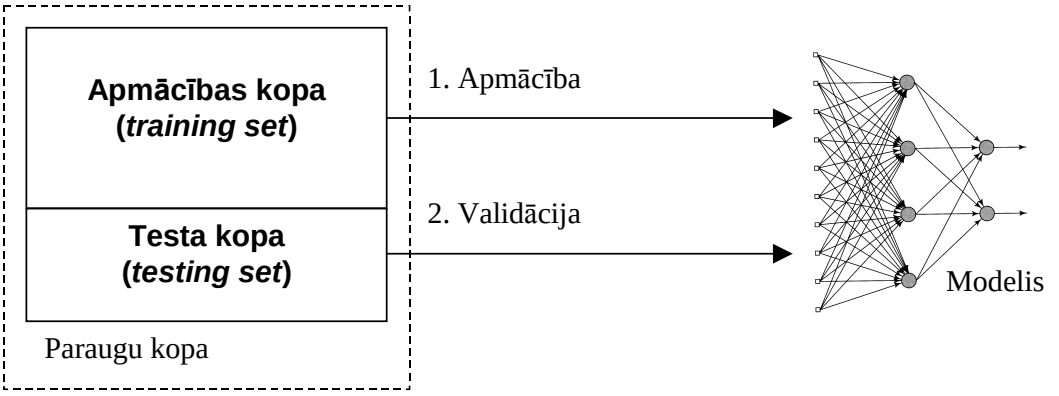
Lai nodrošinātu vispārināšanas spējas uzlabošanu, nepieciešamas metodes vispārināšanas spējas novērtēšanai. Viena no labākajām modeļu novērtēšanas metodēm ir **krusteniskā validācija** (*cross-validation*). Būtībā tā ir nevis novērtēšanas metode, bet gan paraugu dalīšanas metode apakškopās tā, lai viena no apakškopām tiktu izmantota

neironu tīkla apmācībā, bet atlikusī daļa – tā derīguma novērtēšanā (validācijā). Krusteniskās validācijas un citu vispārināšanas spēju novērtēšanas metožu pamatelements ir viena parauga novērtēšanas kļūda (t.i. starpība starp pareizo un neirona tīkla doto rezultātu). Krusteniskajai validācijai ir vairāki paveidi, kas tālāk iztirzāti sīkāk.

#### 4.2.2.1. Metode ar noturēšanu

**Metode ar noturēšanu** (*holdout method*) ir vienkāršākais krusteniskās validācijas paveids, kas pēc būtības nemaz nav krusteniskā validācija, jo nekāda krustošana šeit nenotiek. Paraugi (*data set*) uz labu laimi tiek sadalīti divās kopās – **apmācības kopā** (*training set*) un **testa kopā** (*testing set*). Neironu tīkla apmācība notiek tikai, izmantojot apmācības kopu, bet pēc tam tas tiek pārbaudīts, izmantojot testa kopu. Parasti testa kopā ir ne vairāk kā viena trešdaļa no kopējā paraugu skaita.

Metodes trūkums ir vērtējuma augstā dispersija (*variance*) – vērtējums lielā mērā atkarīgs no tā, kādi paraugu ir izvēlēti apmācības kopai un kādi testa kopai.



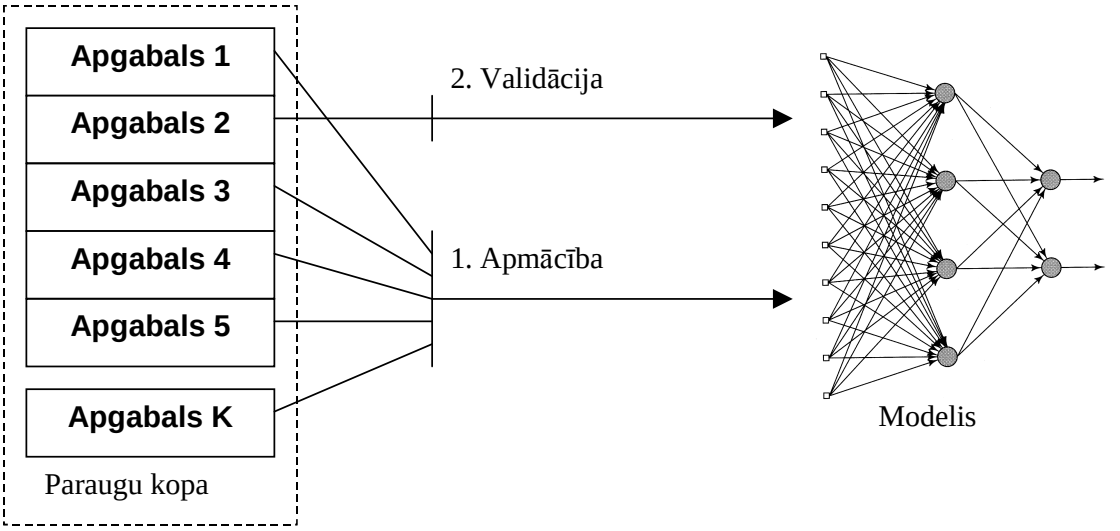

**Attēls 28. Metode ar noturēšanu (*holdout method*)**

#### 4.2.2.2. K-apgabalu krusteniskā validācija

**K-apgabalu krusteniskā validācijā** (*K-fold cross-validation*) paraugu kopa tiek dalīta *K* daļās (apgabalos). Tad *K* reizes tiek veikta novērtēšana pēc metodes ar noturēšanu – viens datu apgabals tiek izvēlēts par testa kopu, bet pārējie *K-1* apgabali – par apmācības kopu. Lai iegūtu kopējo novērtējumu, visi *K* sesiju rezultāti tiek noteiktā veidā kombinēti, piemēram, ņemta to vidējā vērtība.

Izmantojot K-apgabalu validāciju ar krustošanu, katrs paraugs tieši vienu reizi atrodas kādā testa kopā un tieši *K-1* reizi – kādā apmācības kopā. Metodes priekšrocība ir mazāka dispersija nekā metodei ar noturēšanu un tā var vēl tikt samazināta, palielinot *K* vērtību. Metodes trūkums ir lielais nepieciešamais skaitļošanas apjoms – *K* reizes lielāks nekā metodei ar noturēšanu. Vēl viens šīs metodes variants ir tāds, ka sākotnēji

paraugu dalīšana apgabalos nenotiek (apgabali netiek fiksēti), bet apmācības kopas un testa kopas izvēle pēc nejaušības principa notiek pirms katras novērtēšanas sesijas.




**Attēls 29. K-apgabalu krusteniskā validācija**

#### 4.2.2.3. Vienu elementu izslēdzoša krusteniskā validācija

**Vienu elementu izslēdzoša krusteniskā validācija** (*leave-one-out cross-validation*) ir parastās K-apgabalu validācijas galējais variants, kad $K$ ir vienāds ar $N$ – paraugu kopējo skaitu. Tādējādi modeļa novērtējums ir izvērsts $N$ sesiju garumā, un katrā sesijā tieši viens paraugs tiek ievietots testa kopā, bet visi pārējie – apmācības kopā.

### 4.2.3. Apmācības procesa ātrāka pārtraukšana

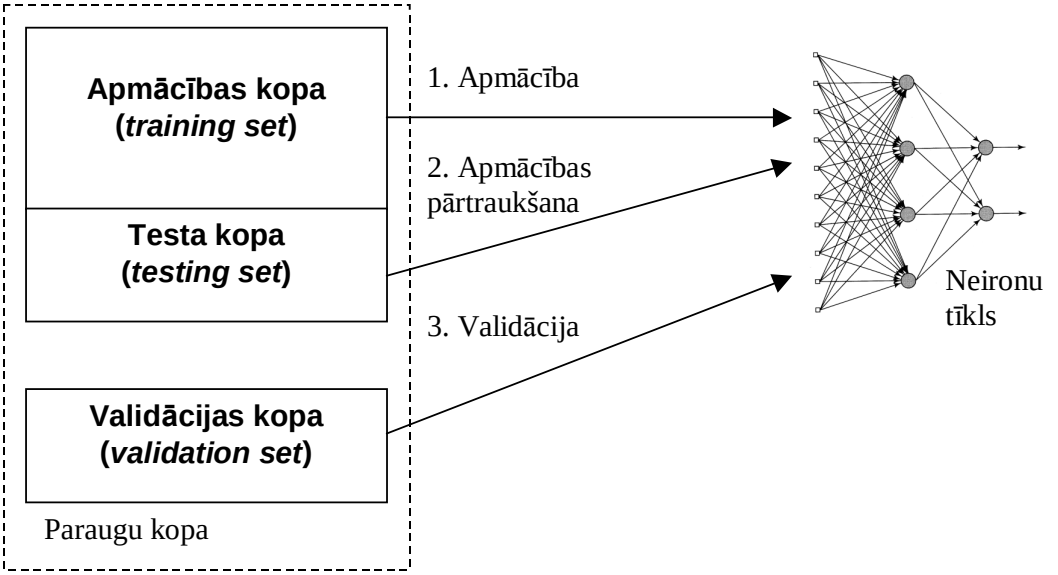
Viena no metodēm neironu tīkla vispārināšanas spējas uzlabošanai ir **apmācības procesa ātrāka pārtraukšana** (*early stopping*), kas nozīmē, ka kontrolēta apmācības process tiek pārtraukts ātrāk nekā neironu aproksimācijas kļūda (jeb kļūda uz ieejas paraugiem) ir samazinājusies zem noteiktas vērtības.

Šim nolūkam no apmācības kopas tiek nodalīta testa kopa, un pēc katras epohas neironu tīkls tie pārbaudīts arī uz testa kopas. Apmācība tiek pārtraukta, kad neironu tīkla kļūda pret testa kopas paraugiem sāk palikt lielāka. Tādējādi aproksimācijas kļūda kalpo kā svaru vērtību izmaiņas virzītājs apmācības procesā, bet vairs ne kā beigšanas kritērijs.

Līdz ar to neironu tīkla apmācībā un validācijā tiek lietotas trīs dažādas paraugu kopas (Attēls 30):

- apmācības kopa – neironu tīkla apmācībai,
- testa kopa – lai noteiktu apmācības procesa beigas,

- validācijas kopa – lai novērtētu iegūto neironu tīklu, izmantojot kādu no modeļu validācijas metodēm (tas pats, kas testa kopa nodaļā 4.2.2).




**Attēls 30. Apmācības procesa ātrāka pārtraukšana**

## *4.3. Heiristikas inteliģentu sistēmu darbībā*

**Heiristika** (*heuristic*) ir metode vai paņēmiens, kas, ekspluatējot konkrētas problēmas specifiskas īpašības, noteiktos gadījumos uzlabo algoritma darbību, tomēr nav teorētiski pamatots un tādējādi nav lietojams vispārīgā gadījumā.

Kaut arī heiristikas ir paņēmieni bez stingra teorētiskā pamatojuma, tomēr tām ir ļoti liela nozīme reālu problēmu risināšanā, it sevišķi pie augstas sarežģītības problēmām, piemēram, *NP-hard*.

### 4.3.1. Heiristikas ģenētisko algoritmu un neironu tīklu darbībā

#### 4.3.1.1. Heiristikas ģenētisko algoritmu darbībā

Tā kā ar ģenētiskajiem algoritmiem parasti risina augstas sarežģītības problēmas, heiristikām šeit bieži ir izšķiroša nozīme, lai nodrošinātu pieņemamu risinājumu iegūšanu saprātīgā laikā. Ņemot vērā to, ka ģenētiskie algoritmi kā tādi pēc būtības ir tikai karkass vai shēma, nevis algoritms, tad jebkura uz ģenētiskajiem algoritmiem bāzēta sistēma praktiski ir veidojama, pievienojot problēmas specifiskas zināšanas, arī heiristikas.

Lai nodrošinātu ĢA darbību saprātīgā laikā, būtiskākā nozīme ir ģenētisko operatoru darbības efektivitātei (gan ātrdarbībai, gan kvalitātei), jo tieši ģenētiskie operatori ir tie, kas nodrošina jaunu (t.sk. labāku, kas ir mūsu galvenā interese) risinājumu ģenerēšanu.

[fernandes99] apraksta vairākas heiristikas ĢA-bāzētā sistēmā stundu saraksta sastādīšanas problēmas risināšanā:

- Speciāla remontēšanas funkcija (*repair function*), kas nodrošina daudz vienkāršāku ģenētisko operatoru uzbūvi un ātrāku izpildi (sīkāks apraksts nodaļā 3.3.2).
- Labāko hromosomu mutācija (*best chromosome mutation*), izveido labāko hromosomu (to, kas izvēlētas elitisma rezultātā) kopijas, kuras pakļauj viena gēna mutācijai.
- Atgriezeniskā remontēšanas funkcija (*backward repair function*) un slikto gēnu mutācija (*bad genes mutation*). Ja parastā remontēšanas funkcijas (kas aprakstīta iepriekš) nedod vajadzīgo rezultātu un joprojām tiek konstatēta resursu pārklāšanās, tiek veikta labāko hromosomu remontēšana pretējā secībā. Tad sliktākā gēna mutācijas operators izveido jaunu hromosomu nākošajai paaudzei, veicot mutāciju kādai no labākajām hromosomām, kur nav bijusi pakļauta izmaiņām atgriezeniskās remontēšanas funkcijas rezultātā.

Savukārt [arous99] piedāvā ĢA un lokālās meklēšanas algoritma kombinēšanu, lai nodrošinātu algoritma efektivitāti.

#### 4.3.1.2. Heiristikas vairākslāņu perceptrona darbībā

Heiristiku izmantošana neironu tīklos parasti saistīta ar apmācības procesa paātrināšanu, jo apmācības procesa ātrdarbība uzskatāma par neironu tīklu kā tehnoloģijas vājajām pusēm. Šim nolūkam tiek lietoti dažādi neironu tīklu modeļu paplašinājumi, kā arī pielietoti dažādi papildus algoritmi konfigurācijas dinamiskai izmaiņai.

- **Inerces faktors** (*momentum*) $\alpha$ nosaka iepriekšējā soļa izmaiņas ietekmi uz pašreizējo svara izmaiņas vērtību (formula (19), salīdzināt ar formulu (7)).

$$\Delta w_{ji}(t) = \eta \delta_j o_i + \alpha \Delta w_{ji}(t-1), \qquad \textbf{(19)}$$

kur $\Delta w_{ji}(t)$ – svara izmaiņa kārtējā solī; $\alpha \Delta w_{ji}(t-1)$ – svara izmaiņas iepriekšējā solī ietekme uz izmaiņu kārtējā solī.

- ***Quickprop* algoritms**. *Quickprop* ir kļūdas atgriezeniskās apmācības metodes (*backpropagation*) variācija, kad kļūdas funkcija tiek tuvināta parabolai ar nolūku paātrināt konverģenci. Šis algoritms (21) balstās uz kļūdas tendences $s(\cdot)$ (20) atšķirību kārtējā un iepriekšējā apmācības solī.

$$s(t) = \frac{dE(t)}{dw_{ji}(t)}, \qquad \textbf{(20)}$$

kur $s(\cdot)$ – kļūdas tendence; $E(\cdot)$ – kļūdas funkcija.

$$\Delta w_{ji}(t) = \frac{s(t)}{s(t-1) - s(t)} \Delta w_{ji}(t-1), \qquad \textbf{(21)}$$

kur $\Delta w_{ji}(t)$ – izmaiņa pašreizējā solī; $\Delta w_{ji}(t-1)$ – izmaiņa iepriekšējā solī.

- ***Rprop* algoritms**. *Rprop* savā ziņā līdzinās *Quickprop*, jo salīdzina kļūdas tendenci iepriekšējā un kārtējā solī, bet šajā gadījumā būtiska ir kļūdas tendences zīme abos soļos (uz vienu pusi, vai dažādām pusēm).
- **Apmācības koeficienta $\eta$ dinamiskas izmaiņas funkcija**, kas nodrošina to, ka sākumā apmācība notiek ātrāk, bet apmācības procesa beigās daudz lēnāk.

## 4.3.2. Uz slāņu topoloģiju balstīts vairākslāņu perceptrona paplašinājums

Šī apakšnodaļa apraksta autora piedāvāto neironu tīkla paplašinājumu.

### 4.3.2.1. Modeļa vispārējs apraksts, ieviešot ieejas slāņa topoloģiju

Rakstā [zuters05b] ir aprakstīts autora piedāvātais vairākslāņu perceptrona paplašinājums, kas nodrošina labāku vispārināšanas spēju ar roku rakstītu paraugu atpazīšanā. Šajā nodaļa dots šīs heiristiskās metodes izklāsts.

Attēls 31 demonstrē cilvēka spēju īpatnības atšķirt dažādus paraugus. Šeit parādīti 4 paraugi ar aptuveni vienādu baltā un melnā proporciju tajos. Ir acīmredzami, ka cilvēks vieglāk atšķir pirmos divus vienu no otra nekā otros divus.


**Attēls 31. Četri paraugi, kas demonstrē cilvēka spēju atšķirt paraugus**

Tas ir tādēļ, ka vienādas krāsas pikseļi ir koncentrējušies kopā. Tādējādi var teikt, ka vieglāks atšķiršanas (atpazīšanas) process ir panākts, papildus izmantojot paraugu topoloģiskās īpatnības.

Autora galvenā ideja, paplašinot vairākslāņu perceptrona (*MLP*) modeli, bija izmantot ieejas signālu savstarpējās attiecības, un, tiem kooperējoties, panākt labākus rezultātus.

Vairākslāņu perceptrons uzskatāms par neironu tīklu etalonu, un *backpropagation* algoritms ir nozīmīgākā apmācības metode neironu tīkliem ar kontrolēto apmācību. MLP darbības un apmācības kodols parādīts formulās (1),(3),(7) (15. lpp.).

Vairākslāņu perceptronam ir pazīstami pietiekoši daudzi paplašinājumi, bet lielākā daļa no tiem saistīta ar apmācības procesa paātrināšanu (nodaļa 4.3.1.2).

Autora piedāvātais vairākslāņu perceptrona modeļa paplašinājums piedāvā vispārināšanas spējas uzlabošanu, iekļaujot modelī topoloģijas specifisku algoritmu pēc līdzības ar Kohonena tīklu. Ar topoloģiju šeit tiek saprasts, ka starp katriem diviem neironiem slānī ir zināma **attāluma funkcija** (*distance function*) *d(·,·)* (22).

$$\forall i, j \exists d(i, j) \tag{22}$$

Ņemot vērā pieejamo informāciju par topoloģiju, starp katriem diviem neironiem vienā slānī var definēt **kaimiņu funkciju** (*neighbourhood factor*) *h(·)* (23) (Attēls 34), kuras tipiska izvēle (tāpat kā Kohonena tīklā) ir Gausa funkcija.

$$h(i, j) = \exp\left( -\frac{d^2(i, j)}{2\sigma^2} \right), \tag{23}$$

kur $\sigma$ – slīpuma koeficients (*effective width*).

Pēc tam, kad ir definēta kaimiņu funkcija, summēšanas funkcijas (1) vietā tiek definētā **paplašinātā summēšanas funkcija** (*extended propagation function*) (24):

$$NET_j = \sum_{i=0}^{n} w_{ji} o_i^* , \tag{24}$$

kur $o_i^*$ – **paplašinātā ieeja** (*extended input*), kas definēta (25):

$$o_i^* = \left\{ \begin{array}{c} \sum_{k=1}^{n} h(i,k) o_k ;\ i \neq 0 \\ 1;\ i = 0 \end{array} \right\}, \tag{25}$$

kur $o_k$ –neirona $j$ ieeja $k$; $i$=0 – attiecas uz pseido-ieeju, kas ienāk uz papildus svaru (*bias*), šī formulas komponente nepieciešama tādēļ, ka formulā (24) summēšana sākas ar 0-to komponenti.

Atbilstošai summēšanas funkcijas paplašinājumam, formulas (7) (23. lpp.) vietā tiek ieviesta **paplašinātā svara izmaiņa** (*extended weight correction rule*) (26):

$$\Delta w_{ji} = \eta \delta_j o_i^* \tag{26}$$

Tā kā paplašinātās ieejas (25) izrēķināšanai potenciāli nepieciešamas visas ieejas vērtības, tad šīs formulas izmantošana tiešā veidā noved pie algoritma ātrdarbības samazināšanā par veselu kārtu. Lai nodrošinātu algoritma ātrdarbību, tiek ierobežots

saskaitāmo skaits, atstājot tikai tos, kas satur pietiekoši lielas vērtības. Formulu (25) šim nolūkam aizstāj formula (27):

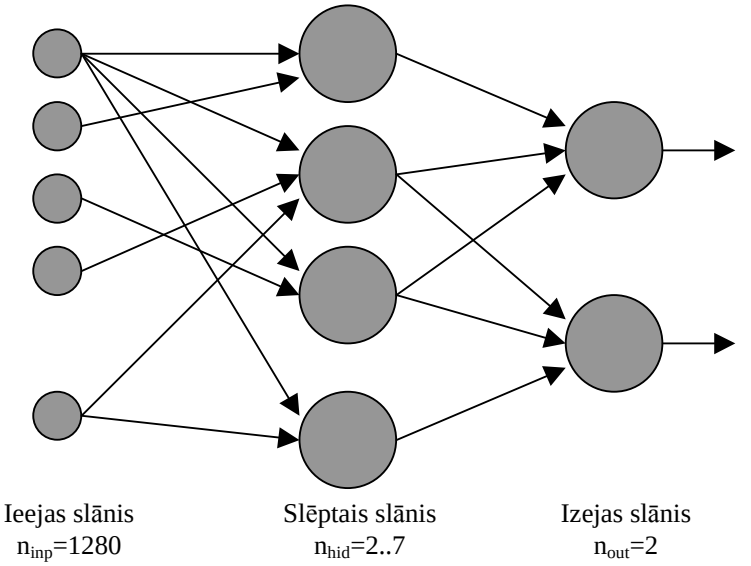
$$o_i^* = \left\{ \begin{array}{c} \sum_{k:\, h(i,k)\geq\Theta} h(i,k)o_k\,;\; i \neq 0 \\ 1;\; i = 0 \end{array} \right\} \qquad \textbf{(27)}$$

Normalizācijas nolūkiem, kaimiņu funkcija (23) tika papildināta ar **normalizācijas faktoru** τ, kas ir konstants fiksētam σ (28):

$$h(i,\, j) = \tau \exp\left( -\frac{d^2(i,\, j)}{2\sigma^2} \right) \qquad \textbf{(28)}$$

#### 4.3.2.2. Eksperimenti ar paplašināto MLP modeli

Eksperimenti tika veikti ar melnbaltiem attēliem (32x40=1280 pikseļi), kuros attēloti ar roku rakstīti cipari (Attēls 33). Eksperimentos tika izmantoti vairākslāņu perceptroni (Attēls 32) ar vienu slēpto slāni, kas tika darbināti, izmantojot nodaļā 4.3.2.1 aprakstītās formulas.




**Attēls 32. Eksperimentos izmantoto neironu tīklu arhitektūra**

Neironu tīkli tika apmācīti paraugu klasificēšanai divās kategorijās.

Tika veikti trīs veidu eksperimenti – 1000 eksperimenti katram veidam:

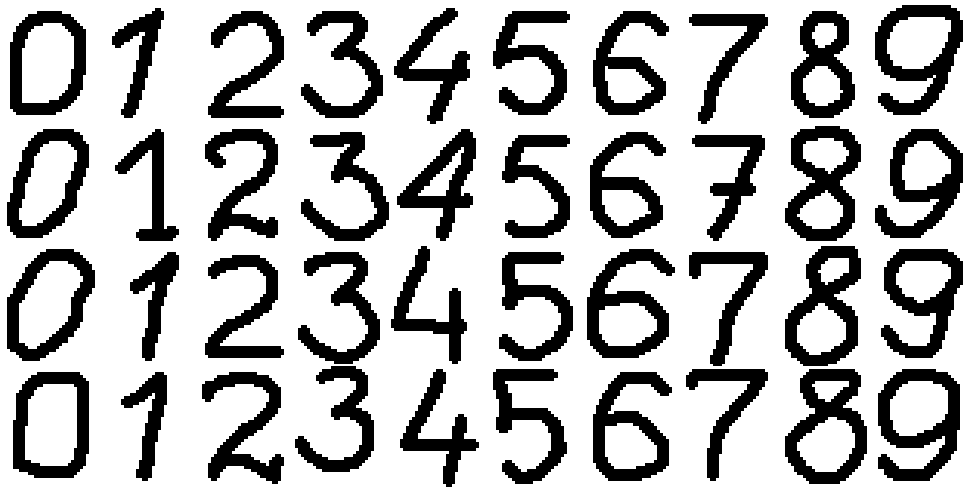
- 1:9 – viens cipars vienā kategorijā, pārējie otrajā kategorijā,
- 3:7,
- 5:5.

Izejas slānī bija divi neironi: pirmais tika apmācīts ar 1 pirmajai kategorijai, bet ar nulli otrajai, bet otrs – otrādi.

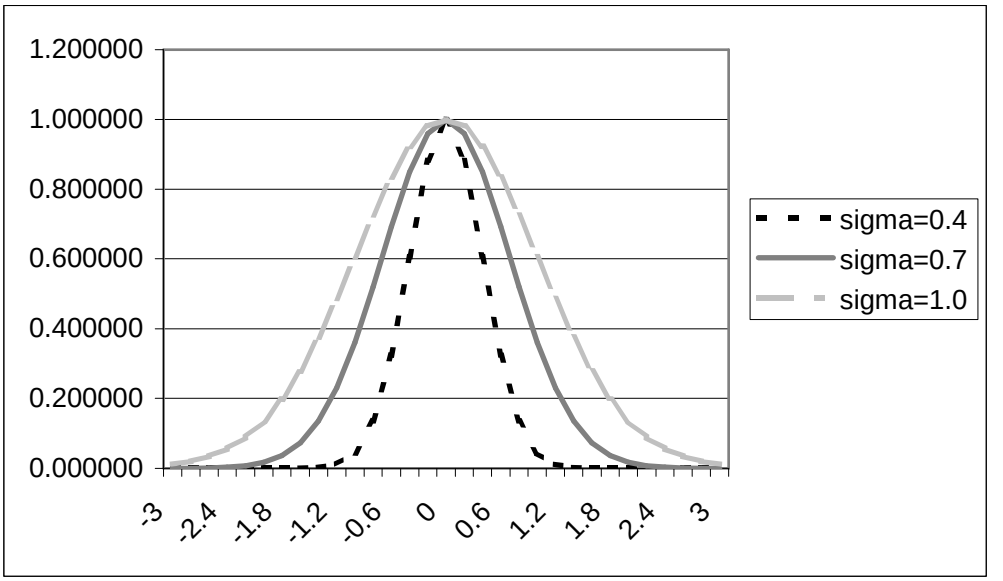
Pirms katra eksperimenta atbilstoši tā tipam (1:9, 3:7 vai 5:5) pēc nejaušības principa tika izvēlēti cipari pirmajai un otrajai kategorijai.

Eksperimentos tika izmantota kaimiņu funkcija ar σ vērtībām {0-klasiskais MLP; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.0}.

Modeļa novērtēšanai atbilstoši pieejamajam paraugu klāstam (Attēls 33) tika izmantota 4 apgabalu krusteniskā validācija (nodaļa 4.2.2.2).



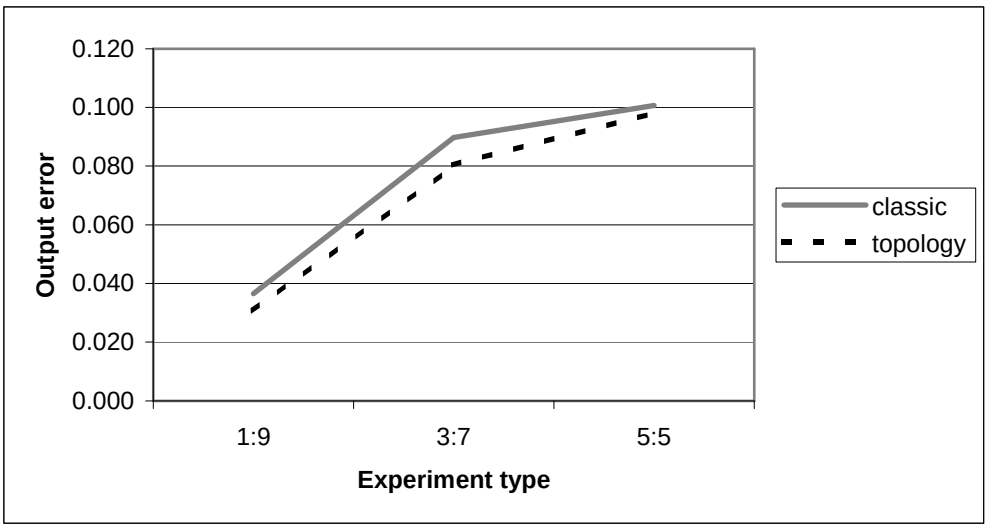
**Attēls 33. Paraugi, kas tika izmantoti apmācībā un testēšanā**



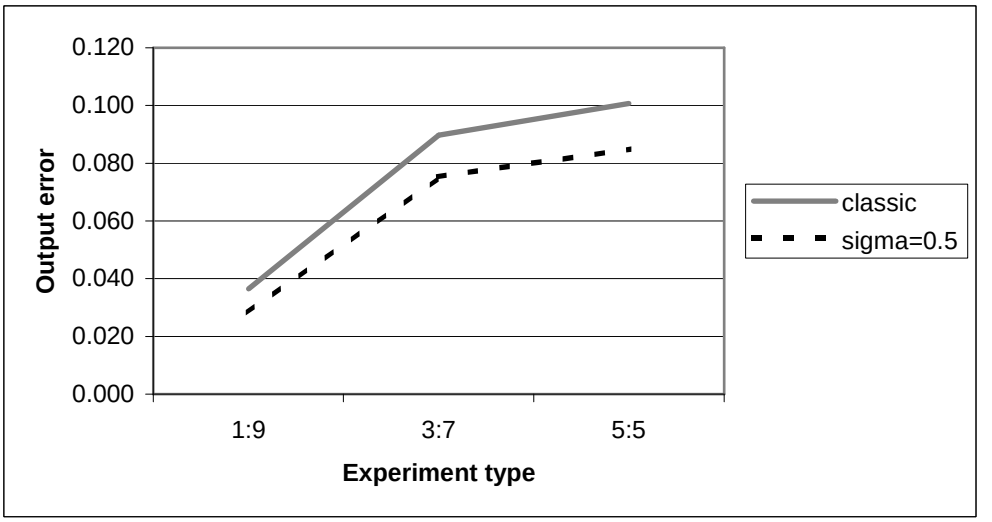

**Attēls 34. Dažas no kaimiņu funkcijām, kas tika izmantotas eksperimentos**

#### 4.3.2.3. Eksperimentu rezultāti

Eksperimentu rezultāti parādīja, ka uz dotajiem testa piemēriem, paplašinātais MLP modelis uzrādīja labākus vispārināšanas rezultātus (t.i. mazāku kļūdu) visās 3 eksperimentu grupās (Attēls 35). Vispārliecinošākie rezultāti tika iegūti eksperimentos ar kaimiņu funkciju, kur σ=0.5 (Attēls 36).




**Attēls 35. Vidējās izejas kļūdas klasiskajam MLP un uz ieejas slāņa topoloģiju balstītajam MLP paplašinājumam (vidēji visām pielietotajām kaimiņu funkcijām)**




**Attēls 36. Vidējās izejas kļūdas klasiskajam MLP un uz ieejas slāņa topoloģiju balstītajam MLP paplašinājumam (ar fiksētu kaimiņu funkciju σ=0.5)**

## *4.4. Nejaušības inteliģentu sistēmu darbībā*

### 4.4.1. Nejaušību izmantošana ģenētiskajos algoritmos un neironu tīklos

**Nejaušībai** (*randomness*) ir liela nozīme gan ģenētisko algoritmu, gan neironu tīklu darbībā. Ģenētiskie algoritmi pēc savas būtības balstās uz nejaušībām, savukārt neironu tīklos ar nejaušībām pārsvarā saistītas atsevišķas darbības, izņēmums ir vienīgi atsevišķi neironu tīklu modeļi.

#### 4.4.1.1. Nejaušības ģenētiskajos algoritmos

Ģenētiskie algoritmi kā skaitļošanas koncepcija, atbilstoši analoģiskajiem dabiskajiem procesiem, ļoti lielā mērā balstās uz nejaušībām, katrā atsevišķā gadījumā pievienojot klāt problēmu specifiskas heiristikas. Nejaušību izmantošana ģenētiskajos algoritmos ir līdzeklis meklēšanas globālā rakstura nodrošināšanā. Nejaušībām ir būtiska nozīme šādās praktiski visās ĢA komponentēs:

- **Sākotnējās populācijas ģenerēšana.** Sākotnējais potenciālo risinājumu kopums tiek ģenerēts, balstoties uz nejaušības principu.
- **Izvēle.** Labākajiem indivīdiem ir lielākas izredzes tikt izraudzītiem nākamajā paaudzē, tomēr pat izvēle notiek pēc nejaušības principa. Vienīgā atkāpe no tā ir elitisms.
- **Vecāku izvēle ģenētiskajiem operatoriem** notiek pēc līdzīgas sistēmas kā izvēle pārejai uz nākamo paaudzi.
- **Ģenētisko operatoru pielietošana.** Uz nejaušību balstās gan pielietošanas noteikšana konkrētai hromosomai, gan hromosomas daļas izvēle, uz kuru attiecas ģenētiskais operators.

#### 4.4.1.2. Nejaušības neironu tīklu darbības nodrošināšanā

Atsevišķi nejaušību pielietojumi, bieži vien komplektā ar heiristikām, sastopami vai ikvienā neironu tīklu modelī, kā arī neironu tīklu ekspluatācijas procesā. Lūk, daži svarīgākie nejaušību pielietošanas gadījumi neironu tīklos:

- **Sākotnējo svaru vērtību ģenerēšana** lielākajā daļā neironu tīklu notiek pēc nejaušības principa, izvēloties vērtības noteiktā intervālā. Izņēmums ir daži neironu tīklu modeļi, piemēram, Hopfīlda tīkls, kur apmācība notiek pēc citiem principiem.
- **Neironu darbināšanas secība** slāņa ietvaros atsevišķos neironu tīklos nenotiek pēc kārtas, bet gan nejaušā secībā, piemēram, jau pieminētajā Hopfīlda modelī.

- **Trokšņu injekcija** apmācības paraugos, lai nodrošinātu labāku neironu tīkla vispārināšanas spēju (sīkāk skatīt nodaļā 4.4.2).
- **Neironu tīklu validācija**, piemēram, apmācības un testa kopas izvēle (sīkāk skatīt nodaļu 4.2.2).

## 4.4.2. Trokšņa izmantošana neironu tīklu apmācībā

### 4.4.2.1. Trokšņa injekcija

**Troksnis** (*noise*) ir nelielas izmaiņas ieejas datos. Trokšņa pamatā ir neprecīzi vai kļūdaini mērījumi, un tas var novest pie sistēmas nekorektas darbības. Neironu tīkli ir samērā robusti pret trokšņiem, un to bieži sauc par vispārināšanas spēju [fausett94].

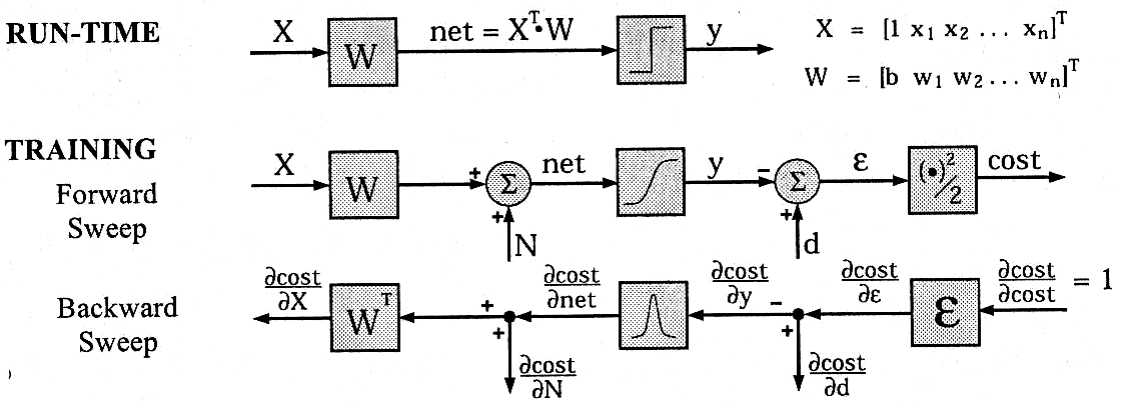
Tomēr apzināta trokšņu pievienošana apmācības piemēros jeb **trokšņa injekcija** (*noise injection*) **ieejas paraugos** daudzos gadījumos nodrošina būtisku vispārināšanas spējas uzlabošanu [bishop95],[seghouane04], lielā mērā ļaujot izvairīties no pārmērīgas pielāgošanās.

Trokšņa pievienošana uzskatāma kā vēl viens no mehānismiem (blakus strukturālajai stabilizācijai un regularizācijai), lai nodrošinātu kontrolētu kompromisu starp tendenciozitāti un dispersiju [bishop95] (sk. arī nodaļu 4.2.1). Vispārīgā gadījumā trokšņa pievienošana nozīmē nejauša vektora $\xi$ pievienošana katram ieejas paraugam x pirms padošanas tīklam kārtējā apmācības solī, tādējādi perceptrona neirona summēšanas funkcija izskatītos kā (29) (salīdzināt ar standarta summēšanas funkciju perceptronam (1)).

$$NET_j = \sum_{i=0}^{n} w_{ji}(x_i + \xi_i) \qquad \textbf{(29)}$$

No heiristiskā viedokļa, pievienojot ieejas paraugiem katru reizi citu nejaušu vektoru, mēs varētu gaidīt, ka troksnis "izsmērēs" datus un neironu tīklam būs grūti pielāgoties katram datu punktam. Tomēr daudzi veikti eksperimenti ir nodemonstrējuši, ka apmācība ar pievienotu troksni var veikt uzlabojumus tīkla vispārināšanas spējā.

[wilson94] piedāvā modificētu *backpropagation* algoritmu ar **trokšņa injicēšanu apmācības laikā pirms aktivizācijas funkcijas** (nevis apmācības piemēros) (Attēls 37).




**Attēls 37. Apmācības algoritms ar trokšņa injicēšanu pirms aktivizācijas funkcijas apmācības procesa "virziena uz priekšu" fāzē [wilson94]**

[hammadi98] parāda, ka apmācība ar trokšņu injicēšanu ieejas paraugos un slēptajos neironos ir līdz pat otrai kārtai statistiski ekvivalentas

#### 4.4.2.2. Apmācības kopas paplašināšana pēc nejaušības principa

[karystinos00] piedāvā **pēc nejaušības principa paplašinātu apmācības paraugu kopu** (*randomly expanded training sets*) izmantošanu – trokšņa injekcijai ieejas paraugos alternatīvu metodi, kas savā ziņā ir idejiski tuva, tomēr pēc būtības daudz savādāka, jo ir saistīta ne tikai ar ieejas, bet arī ar izejas datiem. Idejas pamatā ir lokāli visvairāk entropiskais ieejas un izejas apvienotā varbūtību blīvuma funkcijas novērtējošais aprēķins (*locally most entropic estimate of the true joint input-output probability density function*), no kura pēc nejaušības principa tiek ņemti (ģenerēti) papildus apmācības paraugi.

### 4.4.3. Nejauši ģenerētu paraugu izmantošana neironu tīkla apmācībā

Šī apakšnodaļa apraksta autora izstrādāto neironu tīkla apmācības metodi, kas ir ietverta autora piedāvātajā stundu saraksta sastādīšanas modelī.

#### 4.4.3.1. Problēmas apraksts

Paraugu novērtēšanas problēma var tikt efektīvi risināta, izmantojot vairākslāņu perceptronu. Atliek vienīgi izveidot tādu neironu tīklu, kas aproksimē nezināmu novērtēšanas funkciju $f(\cdot)$ [haykin99]:

$$d = f(x), \tag{30}$$

kur vektors *x* apzīmē ievadu, bet vektors *d* – izvadu.

Šim nolūkam ir dota (pieejama) apmācības piemēru kopa T, kur katrs elements ir pāris:

$$\mathrm{T} = \{(x_i, d_i)\}_{i=1}^{n} \tag{31}$$

Funkcijai *F(·)*, kuru apraksta uz šiem paraugiem apmācīts neirons, būtu jābūt pietiekoši tuvu funkcijai *f(·)* no Eiklīda attāluma viedokļa:

$$\forall x \| F(x) - f(x) \| < \varepsilon, \tag{32}$$

kur $\varepsilon$ – mazs pozitīvs skaitlis.

Šādas problēmas ir tipiskas risināšanai ar kontrolētas apmācības (*supervised*) neironu tīkliem, piemēram, vairākslāņu perceptronu vai RBF tīklu.

Sarežģījumi rodas tad, ja mums nav pieejama tāda apmācības paraugu kopa, kas adekvāti reprezentētu funkciju *f(·)*. Viens variants varētu būt vērtējumu neesamība noteiktiem zināmiem piemēriem. Otrs variants ir, ja apmācībai pieejami ir tikai pozitīvi (ar augstu vērtējumu) paraugi, bet iztrūkst paraugu, kas reprezentētu vidējus vai sliktus risinājumus. Šajā gadījumā neder arī [karystinos00] piedāvātā apmācības piemēru kopas paplašināšana (sk. nodaļu 4.4.2.2), jo nav datu, lai veiktu varbūtību blīvuma funkcijas novērtējumu.

Rakstā [zuters06a] autors piedāvā kontrolētās apmācības procesa organizāciju, kurā tiek pielietoti pilnīgi **nejauši ģenerēti apmācības piemēri**, kas tiek lietoti, lai papildinātu pieejamo apmācības piemēru kopu, kompensējot iztrūkstošos paraugus. Mērķis: aprakstīt un demonstrēt modificētu apmācības metodi, kas ļautu izveidot vērtēt spējīgu (*evaluative*) neironu tīklu, balstoties tikai uz pozitīviem apmācības piemēriem.

Stundu sarakstu novērtēšanas gadījumā, neironu tīkls tiktu apmācīts ar esošiem (tātad, atzītiem par labiem esam) stundu sarakstiem, lai iemācītos novērtēt citus stundu sarakstus.

#### 4.4.3.2. Standarta neironu tīklu kontrolētas apmācības process

Neironu tīkla apmācībai ir izšķiroša nozīme neironu tīkla lietošanā. Apmācības procesam būtu izšķirami divi aspekti:

- Apmācības algoritms (piemēram, (7)). Apmācības algoritmi var ļoti atšķirties dažādiem neironu tīklu modeļiem.
- Apmācības procesa organizācija.

Apmācības process nosaka, kādā veidā apmācības piemēri tiek padoti apmācības algoritmam. Parasti apmācības process tiek organizēts epohās (sk. nodaļu 1.2.3.3). Pseidokods 1 parāda parastu neironu tīkla kontrolētas (*supervised*) apmācības procesu, kāds tas ir, piemēram, vairākslāņu perceptronam.

**Pseidokods 1. Tipisks neironu tīkla kontrolētas apmācības process**

```
PROCEDURE TrainNetworkStandard(NET:NeuralNetwork,TRAINING_SET:
          LabeledTrainingPatterns)
epoch_size:=SizeOf(TRAINING_SET)
REPEAT
    FOR i:=1 TO epoch_size
        {pattern,desired_value}:=GetNextPatternFrom(TRAINING_SET)
        TrainOnSinglePatternSupervised(NET,pattern,desired_value)
    END FOR
UNTIL FinalConditionMet(NET)
```

### 4.4.3.3. Neironu tīkla apmācības procesa modifikācija

Lai nodrošinātu nodaļā 4.4.3.1 nosprausto mērķi – izveidot apmācības metodi, kas ļautu izveidot objektu vērtējošajai izmantojamu neironu tīklu, balstoties tikai uz pozitīviem apmācības piemēriem, autors piedāvā modificētu apmācības procesu, iesaistot apmācībā pēc nejaušības principa ģenerētus apmācības piemērus. Pseidokods 2 parāda piedāvāto apmācības metodi. Īsumā tās ideja ir šāda:

- Dots apmācības piemēru kopums $T$ (***TRAINING_SET***), kas reprezentē tikai vērtējamā apgabala pozitīvos piemērus.
- Atkarībā no nejaušības faktora $\tau$ (***RANDOM_RATE***) tiek palielināts vienas epohas apjoms (***epoch_size***), lai nodrošinātu, ka vidēji katras epohas laikā katrs piemērs no eksistējošās paraugu kopas tiktu padots neironu tīklam.
- Katrā epohas solī atkarībā no nejaušās vērtības (***rnd***) tiek noteikts vai nu parastais režīms, vai nu ģenerēšanas režīms:
  - parastajā režīmā tiek paņemts paraugs (***pattern***) no eksistējošās paraugu kopas un noteikta tā vēlamā novērtēšanas vērtība (***desired_value***) kā 1,
  - ģenerēšanas režīmā paraugs tiek uzģenerēts pēc nejaušības principa, tam piešķirot vēlamo vērtību 0.
- Kad ir zināms paraugs un tā vēlamā vērtība, tiek izsaukts apmācības algoritms dotajam pārim (***TrainOnSinglePatternSupervised***).

Piedāvātā metode savā ziņā līdzinās apmācības paraugu kopas paplašināšanai, kas aprakstīta nodaļā 4.4.2.2, jo metode nosaka ne tikai ieejas paraugu, bet arī vēlamo

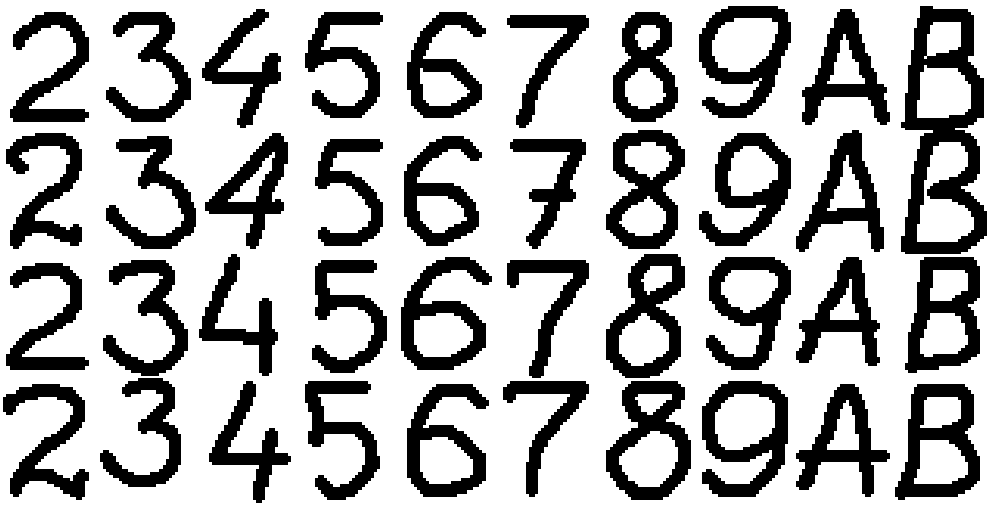
vērtību, tomēr atšķirība ir ļoti būtiska, jo šeit netiek ekspluatēta varbūtību blīvuma funkcija, jo tā nav iegūstama sakarā ar nepieciešamo ieejas paraugu iztrūkumu, kādēļ papildus paraugi tiek ģenerēti pilnīgi pēc nejaušības principa.

**Pseidokods 2. Neironu tīkla apmācības process ar papildus pēc nejaušības principa ģenerētiem apmācības piemēriem**

```
PROCEDURE TrainNetwork(NET:NeuralNetwork,
          TRAINING_SET:ReadyClassTimeTables,RANDOM_RATE:0..1)
epoch_size:=Round(SizeOf(TRAINING_SET)/(1-RANDOM_RATE))
REPEAT
    FOR i:=1 TO epoch_size
        rnd:=GetRandomValueBetween(0,1)
        IF rnd<RANDOM_RATE THEN
            pattern:=GenerateRandomPattern()
            desired_value:=0
        ELSE
            pattern:=ChoosePatternFrom(TRAINING_SET)
            desired_value:=1
        END IF
        TrainOnSinglePatternSupervised(NET,pattern,desired_value)
    END FOR
UNTIL FinalConditionMet(NET)
```

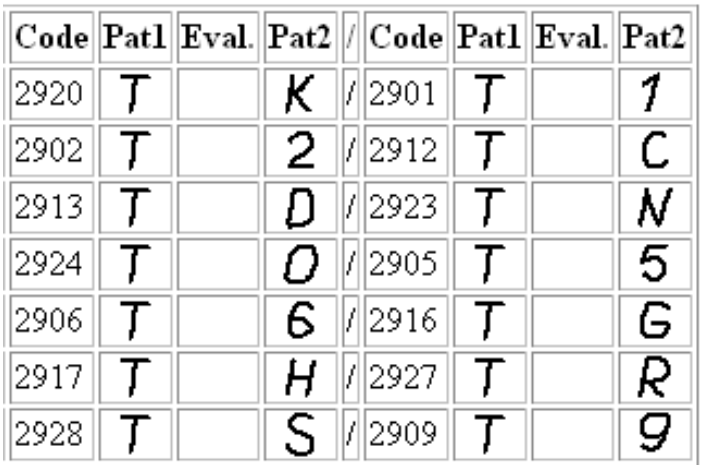
### 4.4.3.4. Eksperimenti piedāvātās apmācības metodes pārbaudei

**Apmācības piemēru kopa.** Apmācības piemēru kopa $T^0$ tika izveidota no rakstītiem simboliem (cipariem un lielajiem burtiem). Kopumā tika izmantoti 36 simbolu veidi (10 cipari un 26 latīņu alfabēta burti), katrs no tiem tika pārstāvēts piemēru kopā ar 4 eksemplāriem (Attēls 38). Apmācības piemēros pārstāvēto simbolu veidu kopu apzīmēsim ar $I^0$.



**Attēls 38. Apmācības piemēru kopas $T^0$ fragments**

**Līdzīguma funkcijas noteikšana ar anketēšanas metodi.** Tika izveidota anketa, kurā parādīti visi iespējamie $I^0$ simbolu pāri (kopā 630 pāri) (Attēls 39). Respondentiem bija jānovērtē simbolu līdzīguma pakāpe starp katra pāra elementiem, kas vēlāk tika normalizēta uz intervālu 0..1, kur vērtība 1 nozīmē – ļoti līdzīgs, bet vērtība 0 – pilnīgi atšķirīgs (piemēram, cipars '0' un burts 'O' ir ļoti līdzīgi, bet cipars '1' un burts 'S' – nē).




**Attēls 39. Anketas paraugs simbolu paraugu novērtēšanai**

Tika aptaujāti 8 respondenti un 204 pāros no 630 vismaz viens no respondentiem bija saskatījis kaut minimālu (no 0 atšķirīgu līdzību). Apkopojot respondentu sniegtās atbildes tika izveidots simbolu **līdzīguma mērs** (*similarity measure*) $g(\cdot,\cdot)$ (33):

$$g : \Gamma^0 \times \Gamma^0 \to \{0..1\} \qquad \textbf{(33)}$$

**Tabula 1. Aptaujas rezultātu fragments.: simbolu pāri, kas tika novērtēti ar vismaz 0.2 no 1**

| Pat1 | Pat2 | Eval. |
|---|---|---|
| 0 | O | 0.98 |
| 6 | G | 0.80 |
| 0 | D | 0.74 |
| O | Q | 0.71 |
| 0 | Q | 0.70 |
| 1 | I | 0.65 |
| 5 | S | 0.64 |
| 8 | B | 0.61 |
| D | O | 0.61 |
| 7 | Z | 0.60 |
| C | G | 0.54 |
| P | R | 0.50 |
| E | F | 0.48 |
| U | V | 0.46 |
| 1 | 7 | 0.44 |
| K | X | 0.44 |
| 8 | S | 0.39 |
| V | Y | 0.36 |
| 3 | 8 | 0.35 |
| 6 | C | 0.33 |

| Pat1 | Pat2 | Eval. |
|---|---|---|
| 1 | J | 0.31 |
| 6 | 9 | 0.31 |
| 7 | T | 0.31 |
| B | R | 0.31 |
| X | Y | 0.31 |
| M | N | 0.29 |
| 6 | S | 0.26 |
| C | O | 0.25 |
| G | Q | 0.25 |
| 1 | T | 0.24 |
| B | E | 0.24 |
| I | J | 0.24 |
| N | V | 0.23 |
| 9 | O | 0.21 |
| D | Q | 0.21 |
| F | T | 0.21 |
| 2 | Z | 0.20 |
| 3 | B | 0.20 |
| H | M | 0.20 |

Izmantojot aptaujas rezultātā iegūto līdzīguma mēru $g(\cdot,\cdot)$, var nodefinēt **līdzīguma funkciju** (*similarity function*) $h(\cdot)$ attiecībā pret fiksētu simbolu kopu $\Gamma$ (34):

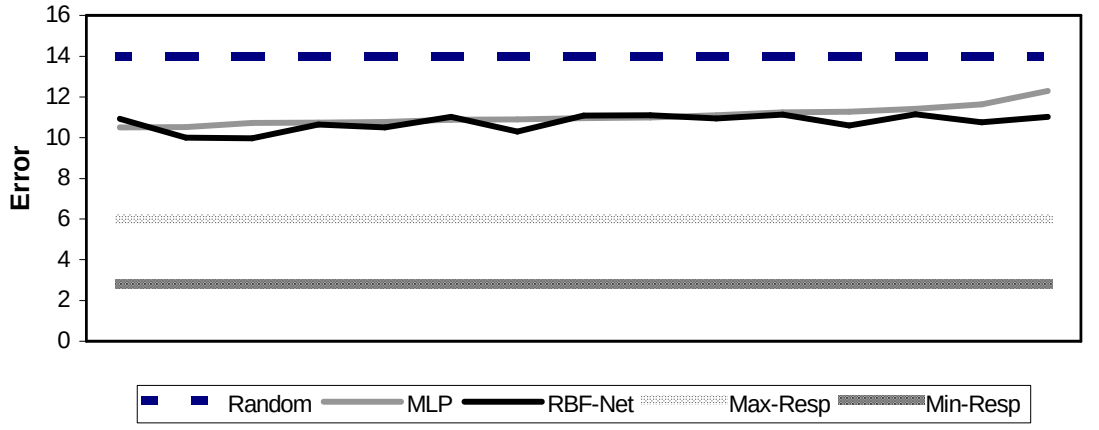
$$h_{\Gamma}(i) = \max_{j \in \Gamma} g(j,i) \qquad \textbf{(34)}$$

Līdzīguma funkcijas $h_{\Gamma}$ būtība ir: kāda līdzīguma pakāpe ir brīvi izvēlētam simbolam $i$ pret fiksētu simbolu kopu $\Gamma$, kas tiek definēts kā līdzība ar to $\Gamma$ elementu, kas ir vislielākā.

**Neironu tīklu konstruēšana un darbināšana.** Tika konstruēti neironu tīkli (vairākslāņu perceptroni un radiālo bāzes funkciju tīkli). Fiksējot noteiktu simbolu kopu $\Gamma$, neironu tīkli tika apmācīti, izmantojot piedāvāto algoritmu (Pseidokods 2), t.i. tikai uz pozitīvajiem piemēriem, kas reprezentē $\Gamma$, un ar dažādiem nejaušības faktoriem $\tau$, ar nolūku iegūt tādus neironu tīklus $\varphi_{\Gamma}(\cdot)$, kas, darbināti uz paraugiem no $T^0$, sniegtu pēc iespējas tuvākus rezultātus ar anketēšanas palīdzību iegūtajai līdzīguma funkcijai $h_{\Gamma}(\cdot)$.

### 4.4.3.5. Eksperimentu rezultāti

Iegūtie dati tika analizēti, izmantojot speciālu virknēšanas (*ranking*) metodoloģiju, kas nozīmē, ka, lai salīdzinātu divas virknes, tika salīdzināti divu virkņu attiecīgo elementu kārtas numuri. Rezultāti parādīja, ka šādi neironu tīkli spēj noteiktā mērā tuvoties vēlamajai funkcijai $h_{\Gamma}(\cdot)$.




**Attēls 40. Eksperimentu rezultātu kopsavilkums simbolu līdzības noteikšanai – kļūdas līmenis**

Apzīmējumu skaidrojums (Attēls 40):

- ***Random*** – vidējais kļūdas līmenis nejauši ģenerētiem paraugiem,
- ***MLP*** – vairākslāņu perceptrons,
- ***RBF-Net*** – radiālo bāzes funkciju tīkls,
- ***Max-Resp*** – lielākā kāda respondenta kļūda,
- ***Min-Resp*** – mazākā kāda respondenta kļūda.

Kā redzams no kopsavilkuma (Attēls 40), rezultāti ir visai tālu no ideāla – krietni sliktāki un tuvāki nejaušu paraugu rezultātiem par vissliktāk atbildējušā respondenta rezultātiem, tomēr zināms efekts ir iegūts, ja rēķinās ar to, ka apmācība tika veikta tikai, izmantojot pozitīvos paraugus.

Eksperimenti tika veikti ar dažādām nejaušības faktora $\tau$ vērtībām (0, 0.2, 0.5, 0.7, 0.9), kur 0 nozīmē, ka netika izmantoti papildus ģenerētie paraugi, tīklu apmācot tikai uz pozitīvajiem piemēriem. Bez tam eksperimenti tika veikti fiksētajām simbolu kopām $\Gamma$ ar dažādu lielumu $c$ (1, 2 un 4). Sekojošā tabula (Tabula 2) parāda rezultātu kopsavilkumu radiālo bāzes funkciju tīkliem. Pēc tās redzams, ka sliktākie rezultāti ir ar nejaušības faktoru $\tau$=0, t.i., bez papildus ģenerēto paraugu izmantošanas, kas apstiprina pēc nejaušības principa ģenerēto apmācības paraugu izmantošanas efektu.

**Tabula 2. Eksperimentu rezultātu kopsavilkuma izvērsums radiālo bāzes funkciju tīkliem**

| Tīkla kļūda | Nejaušības faktors ($\tau$) | Simbolu skaits fiksētajā kopā (c) |
|---|---|---|
| 9.959279 | 0.5 | 4 |
| 9.998273 | 0.9 | 4 |
| 10.30499 | 0.2 | 2 |
| 10.50103 | 0.2 | 4 |
| 10.59219 | 0.9 | 2 |
| 10.63132 | 0.7 | 4 |
| 10.74725 | 0.0 | 2 |
| 10.93429 | 0.5 | 2 |
| 10.93745 | 0.2 | 1 |
| 11.01699 | 0.0 | 4 |
| 11.02072 | 0.5 | 1 |
| 11.09412 | 0.7 | 2 |
| 11.10901 | 0.7 | 1 |
| 11.12824 | 0.9 | 1 |
| 11.14873 | 0.0 | 1 |

Kaut arī nejauši ģenerētu apmācības paraugi apmācības procesā nevar sacensties ar "īstajiem" paraugiem, tomēr eksperimentu rezultāti ir parādījuši, ka to izmantošana dod efektu, ja pilnvērtīga apmācības kopa nav pieejama.

# 5. Ģenētisko algoritmu un neironu tīklu hibrīdais modelis skolu stundu sarakstu sastādīšanai

Rakstā [zuters05a] autors piedāvājis, un tālāk rakstos [zuters06a], [zuters06b], [zuters07] attīstījis ģenētisko algoritmu un neironu tīklu hibrīdo modeli skolu stundu saraksta problēmas risināšanai. Neironu tīkls piedāvātajā modelī tiek lietots kā derīguma funkcijas komponente, tādējādi palīdzot novērtēt stundu saraksta atbilstību prasībām.

Šajā nodaļā sniegtais apraksts ietver divas galvenās autora piedāvātās novitātes:

- Stundu saraksta sastādīšanas modelis, kas ir promocijas darba galvenais rezultāts.
- Modeļa novērtēšanas metodika.

Apraksts strukturēts 3 apakšnodaļās:

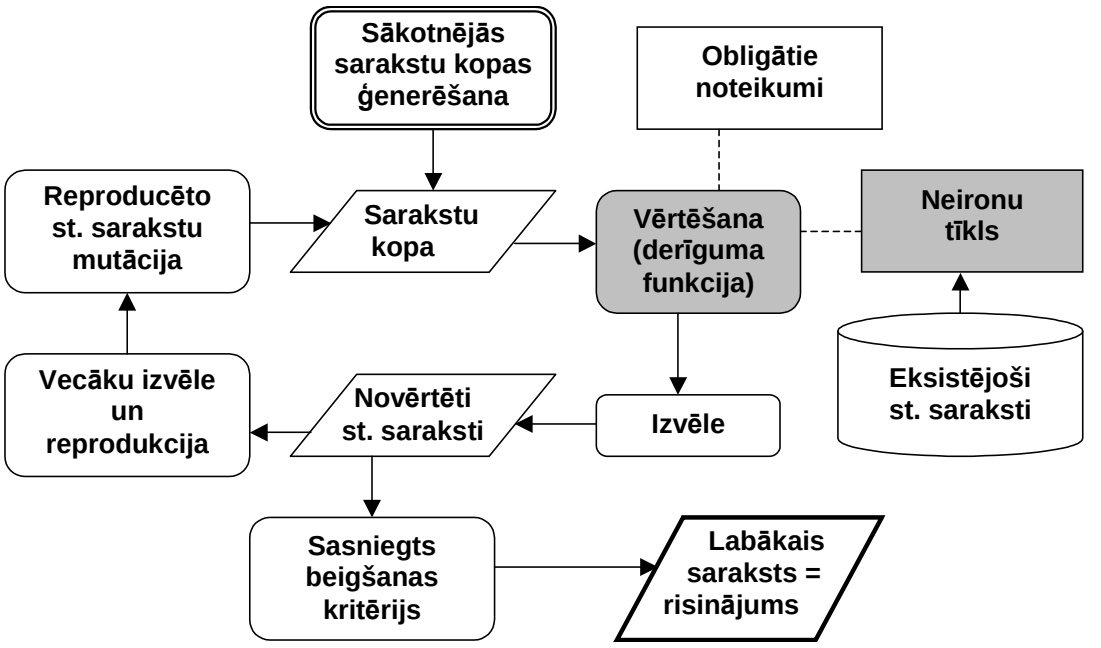
- Nodaļa 5.1 satur koncentrētu piedāvātās idejas – ģenētisko algoritmu un neironu tīklu hibrīdā modeļa – aprakstu.
- Nodaļa 5.2 piedāvāto modeli apraksta detalizēti, izvēršot izklāstu pa tās trijām komponentēm: stundu saraksta sastādīšanas bloks, neironu tīklu bloks un ģenētisko algoritmu bloks.
- Nodaļa 5.3 apraksta autora speciāli izveidoto metodiku modeļa novērtēšanai, kā arī atbilstoši šai metodikai veiktos eksperimentus un to rezultātus, kas apstiprina modeļa darbības efektivitāti.

## *5.1. Piedāvātā modeļa vispārīgs apraksts*

Galvenā ideja neironu tīklu izmantošanai stundu saraksta sastādīšanas problēmas risināšanai ir radusies no novērojuma, ka eksistējoši stundu saraksti ir noderīga informācija jaunu stundu sarakstu sastādīšanai. Ar derīgiem stundu sarakstiem apmācīti neironu tīkli varētu būt noderīgi stundu sarakstu novērtēšanā uz ģenētiskajiem algoritmiem bāzētā stundu sarakstu sastādīšanas sistēmā. Cerības uz iespēju izmantot esošus (arī citu skolu) stundu sarakstus jaunu stundu sarakstu veidošanā balstās uz to, ka dažādos (arī dažādu skolu) stundu sarakstos ir līdzīgs nodarbībās iesaistīto mācību priekšmetu kopums un ka mācību priekšmetu izvietojumam stundu sarakstā varētu būt noteikta loma.

Lai izstrādātu mehānismus stundu sarakstu novērtēšanai, tradicionāli būtu nepieciešami precīzi kritēriji, bet tik sarežģītiem objektiem kā stundu saraksti tos varētu būt grūti iegūt un pat apzināties to esamību. Šādā situācijā neironu tīkli varētu palīdzēt, jo, neesot nepieciešamībai pēc precīziem kritērijiem, varētu atvieglot vērtēšanas mehānisma izveidi.

Autors piedāvā uz ĢA bāzētu skolu stundu sarakstu sastādīšanas modeli, kurā derīguma funkciju papildina uz eksistējošiem stundu sarakstiem apmācīti neironu tīkli, kas apmācīti lietojot autora piedāvāto apmācības metodi ar nejauši ģenerētiem apmācības piemēriem (sk. nodaļu 4.4.3). Atšķirībā no dažādām heiristikām, kas tiek izmantotas ĢA darbības nodrošināšanai (sk. nodaļu 4.3.1.1), piedāvātā metode attiecas tieši uz derīguma funkciju (*fitness function*).




**Attēls 41. Uz ģenētiskajiem algoritmiem bāzēts skolu stundu sarakstu sastādīšanas modelis, kas papildināts ar neironu tīkliem**

## 5.2. Neironu tīklu un ģenētisko algoritmu hibrīdās sistēmas izveidošana un konfigurēšana

Eksperimentu veikšanai autors izveidoja oriģinālu datorsistēmu ar kopējo apjomu vairāki desmiti tūkstošu pirmkoda rindiņu (C++), kas tika izmantota arī citu eksperimentu veikšanai, kas aprakstīti [zuters05b], [zuters06b] un [zuters06a]. Izveidotajai sistēmai ir 3 galvenie loģiskie bloki:

- stundu sarakstu sastādīšanas bloks,
- neironu tīklu bloks,
- ģenētisko algoritmu bloks.

Šī apakšnodaļa (5.2) satur izstrādātās sistēmas aprakstu pa tās loģiskajiem blokiem no uzbūves, izmantoto algoritmu un konfigurācijas viedokļa. Apakšnodaļas uzdevums ir iz-

gaismot kontekstu, kādā ir tikuši veikti tālāk (apakšnodaļā 5.3) aprakstītie eksperimenti, lai parādītu autora piedāvāto ideju efektivitāti.

## 5.2.1. Stundu saraksta sastādīšanas bloks

### 5.2.1.1. Struktūra un darbības princips

Atbilstoši [fernandes99] (3.3.1.2) stundu saraksta sastādīšanas bloks sastāv no 2 veidu datiem:

- Konstantā informācija – kopīga vairākiem stundu sarakstiem, kas ir kandidāti uz gala risinājumu (piemēram, klases, skolotāji, kā arī nodarbību piesaiste noteiktam priekšmetam, klasei, skolotājam un nodarbību garums).
- Dinamiskā informācija – informāciju par laiku, kurā ieplānota nodarbība.

Jau pats stundu saraksta sastādīšanas bloks nodrošina vairāku paralēlu stundu sarakstu veidošanu, ļoti lielā mērā nodrošinot visus datus vienas populācijas veidošanai ģenētisko algoritmu blokā.

Realizētā stundu saraksta sastādīšanas bloka īpašības:

- katrai nodarbībai var būt viens vai divi skolotāji,
- klases var dalīt grupās,
- klases un grupas var apvienot, tādējādi uz vienu nodarbību var attiekties vairāk nekā viena klase.

Stundu saraksta bloks ietver sevī arī stundu saraksta novērtēšanas funkciju $\varphi(\cdot)$ (35):

$$\varphi(\mathrm{T}) = \sum_{i=1}^{c} \alpha_i \varphi_i(\mathrm{T}), \qquad \textbf{(35)}$$

kur $\varphi_i(\cdot)$ – novērtēšanas funkcija atbilstoši kritērijam $i$; $\alpha_i$ – noteikuma $i$ svars; $T$ – stundu saraksts; $c$ – vērtēšanas kritēriju skaits.

Vienam kritērijam atbilstošā novērtēšanas funkcija $\varphi_i(\cdot)$ tiek veidota kā apkopojums no atsevišķu klašu sarakstu vērtējumu apkopojums pēc šī kritērija (36):

$$\varphi_i(\mathrm{T}) = \sum_{k=1}^{g} \psi\left(\varphi_i^*(\mathrm{T}_k)\right), \qquad \textbf{(36)}$$

kur $\varphi_i^*(\cdot)$ – vērtēšanas funkcija vienai klasei atbilstoši kritērijam $i$; $\psi(\cdot)$ – vērtējuma pastiprināšanas pakāpe; $T_k$ – klases k stundu saraksts; $g$ – klašu skaits.

Stundu sarakstu vērtēšana notiek pēc sodīšanas (*penalty*) principa, t.i., labākam sarakstam atbilst mazāka vērtība.

Vērtēšanas funkcija tika realizēta četriem kritērijiem (Tabula 3).

**Tabula 3. Kritēriji stundu sarakstu vērtēšanai**

| Nr. | Kritērijs |
|---|---|
| 1 | Izvietot nodarbības vienmērīgi pa visu nedēļu |
| 2 | Samazināt logus starp stundām skolēniem |
| 3 | Cik iespējams, izvietot nodarbības maiņas sākumā |
| 4 | Līdzsvarot nodarbību izvietojumu, ņemot vērā priekšmetu tipus (piemēram, neieplānot par daudz eksakto priekšmetu pēc kārtas) |

**Stundu sarakstu integritātes nodrošināšanai** stundu saraksta bloks uztur nedēļas plānu katrai klasei, skolotājam un telpai.

#### 5.2.1.2. Plānošanas sistēma

Stundu saraksta datu pamatelements ir **nodarbība**. Nodarbība var būt ieplānota (*scheduled*) vai neieplānota (*unscheduled*). Realizētās plānošanas sistēmas ietvaros nav pieejama telpas piesaiste nodarbībai.

Vienas nodarbības ieplānošana jeb **ieplānošanas operācija** (noteiktas dienas un sākuma perioda piešķiršana nodarbībai) ir plānošanas sistēmas pamatelements. Izveidotajā sistēmā tiek nodrošināts, ka katra ieplānošanas operācija saglabā kopējo stundu saraksta integritāti.

Ieplānošanas operācija (Pseidokods 3) ir veidota izmantošanai no ĢA bloka un ir pilnībā balstīta uz nejaušību. Tā ietver divus galvenos blokus:

- **Nejaušais bloks.** Vairākas reizes pēc nejaušības principa tiek izvēlēta diena un pozīcija nodarbības ieplānošanai pieejamajā periodu apgabalā. Kolīdz tiek atrasta pozīcija, kurā nodarbība ir ievietojama, nepārkāpjot saraksta integritāte, ieplānošana tiek veikta, un ieplānošanas operācija tiek pabeigta.
- **Kompaktais bloks.** Ja nejaušajā blokā nodarbību neizdodas ieplānot, sāk darboties kompaktais bloks, kura laikā tiek pēc kārtas pārstaigātas un pārbaudītas visas iespējamās pozīcijas. Kolīdz tiek atrasta pozīcija, kurā nodarbība ir ievietojama, nepārkāpjot saraksta integritāte, ieplānošana tiek veikta, un ieplānošanas operācija tiek pabeigta.

**Pseidokods 3. Ieplānošanas operācija**

```
PROCEDURE SCHEDULINGOPERATION(T:TimeTable,L:Lesson)
i:=1
REPEAT
    day:=GETRANDOMVALUE(1..5)
    position:= GETRANDOMVALUE(1..MAX_POSITION)
```

```
    IF NotConflicting(T,L,day,position) THEN
        SetTimeToLesson(T,L,day,position)
    END IF
    i:=i+1
UNTIL i>MAX_SCHEDULE_COUNR OR L.scheduled
IF NOT L.scheduled THEN
    FORALL (day,position) IN T.available_positions
        IF NotConflicting(T,L,day,position) THEN
            SetTimeToLesson(T,L,day,position)
        END IF
    END FORALL
END IF
```

Stundu saraksts ir uzskatāms par sastādītu, ja visas nodarbības ir ieplānotas.

Tā kā ieplānošanas operācija nodrošina stundu saraksta integritātes saglabāšanu, tad nav iespējams, ka divas nodarbības būtu ieplānotas, savstarpēji konfliktējot, tai pat laikā teorētiski var gadīties situācija, ka dažas nodarbības var vispār palikt neieplānotas.

### 5.2.2. Neironu tīklu bloks

Neironu tīkla bloks tika izveidots, lai, apmācīts ar gatavajiem stundu sarakstiem, tas varētu tikt izmantots stundu sarakstu vērtēšanai ĢA sistēmas ietvaros. Neironu tīklu bloks veiktajos eksperimentos sastāvēja no viena neironu tīkla, kam ieejā tika padots vienas klases stundu saraksts, bet izejā tika saņemts tā vērtējums.

#### 5.2.2.1. Stundu saraksta kodēšana neironu tīklam

Neironu tīklu un ģenētisko algoritmu hibrīdā modeļa galvenā ideja balstās uz pieņēmumu, ka gatavi stundu saraksti ir derīga informācija jaunu stundu sarakstu sastādīšanā (Attēls 42).

| D | St | 7a | 7b | 7c | 8a |
|---|---|---|---|---|---|
| PIRMDIENA | 1 | | Latviešu val. | Matemātika | Angļu val. |
| | 2 | Ģeogrāfija | Angļu val. | Matemātika | Angļu val. |
| | 3 | Mājturība | Informātika | Latviešu val. | Vēsture |
| | 4 | Mājturība | Ģeogrāfija | Informātika | Latviešu val. |
| | 5 | Latviešu val. | Krievu/Vācu | Vēsture | Fizika |
| | 6 | Latviešu val. | Matemātika | Angļu val. | Ģeogrāfija |
| | 7 | Angļu val. | Vēsture | | Matemātika |
| | 8 | | | | Matemātika |
| | 9 | | | | |
| OTRDIENA | 1 | Matemātika | Sports | | Latviešu val. |
| | 2 | Krievu/Vācu | Ētika | Matemātika | Sports |
| | 3 | Bioloģija | Matemātika | Krievu val. | Mājturība |
| | 4 | Sports | Latviešu val. | Bioloģija | Mājturība |
| | 5 | Mūzika | Krievu/Vācu | Angļu val. | Matemātika |
| | 6 | Latviešu val. | Matemātika | Matemātika | Vācu/Krievu |
| | 7 | Angļu val. | Audzināšana | Latviešu val. | Matemātika |
| | 8 | Audzināšana | | Audzināšna | Audzināšana |
| | 9 | | | | |

**Attēls 42. Gatava stundu saraksta fragments, kas varētu tikt izmantots neironu tīkla apmācībā**

Tomēr gatavajos un pieejamajos (arī citu skolu) stundu sarakstos vienīgā derīgā informācija ir mācību priekšmeti, resp. mācību priekšmetu izvietojums pa dienām un periodiem (mācību priekšmetu kopums ir vienīgais, kas vairāk vai mazāk varētu būt līdzīgs starp dažādu skolu stundu sarakstiem). Ņemot vērā šo ierobežojumu, stundu saraksts vienai klasei $T_k$ var tikt reprezentēts kā mācību priekšmetu virkne (37):

$$\mathrm{T}_k = \left\langle s_{ki} \right\rangle_{i=1}^{p}, \qquad \textbf{(37)}$$

kur $s_{ki}$ – priekšmets klasei $k$ periodā $i$ (var būt tukšs); $p$ – kopējais pieejamo periodu skaits (piemēram, nedēļā).

Tā kā vienai klasei vienā periodā (apvienoto klašu dēļ) var neatbilst viens vienīgs priekšmets, bet var būt vairāki paralēli priekšmeti (Attēls 42, klase 7a, otrdiena, periods 2), stundu saraksta reprezentācija (37) tiek modificēta un paliek nedaudz sarežģītāka (38):

$$\mathrm{T}_k = \left\langle \left( \left\langle s_{ki1}, v_{ki1} \right\rangle, \left\langle s_{ki2}, v_{ki2} \right\rangle, \ldots \right) \right\rangle_{i=1}^{p}, \qquad \textbf{(38)}$$

kur $s_{kij}$ – priekšmets $j$ klasei $k$ periodā $i$; $v_{kij}$ – priekšmeta $j$ vērtība (svars) klasei $k$ periodā $i$.

Latvijas vispārizglītojošajās skolās sastopamo priekšmetu skaits ir ar kārtu 100. Tā kā galvenā interese no stundu saraksta, kuru reprezentē mācību priekšmetu virkne, ir iegūt informāciju par vēlamajām tematiskajām virknēm, un vairāki priekšmeti ir tematiski līdzīgi, tad, lai atvieglotu aprēķinus un lietderīgi izmantotu tematisko saistību starp dažādiem priekšmetiem, tiek ieviests jēdziens "mācību priekšmeta kategorija", kas

apvieno vairākus priekšmetus, un mācību priekšmetu vietā tika izmantotas kategorijas (piemēram, matemātikas vai fizikas vietā lietot: eksaktais priekšmets).

Šim nolūkam tika definēts mācību priekšmetu sadalījums pa kategorijām (autora subjektīva izvēle), turklāt mācību priekšmets šajā sadalījumā var tikt piesaistīts vairāk nekā vienai kategorijai un katrai ar noteiktu svaru (39):

$$s = \langle \sigma_1, v_1 \rangle, \langle \sigma_2, v_2 \rangle, \ldots, \tag{39}$$

kur $s$ – mācību priekšmets; $\sigma_j$ – priekšmetam atbilstošā kategorija; $v_j$ – kategorijas $j$ vērtība (svars) priekšmetā.

Kopā tika noteiktas 12 dažādas priekšmetu kategorijas. Tabula 4 parāda mācību priekšmetu kategorijas un dažus tām atbilstošos priekšmetus.

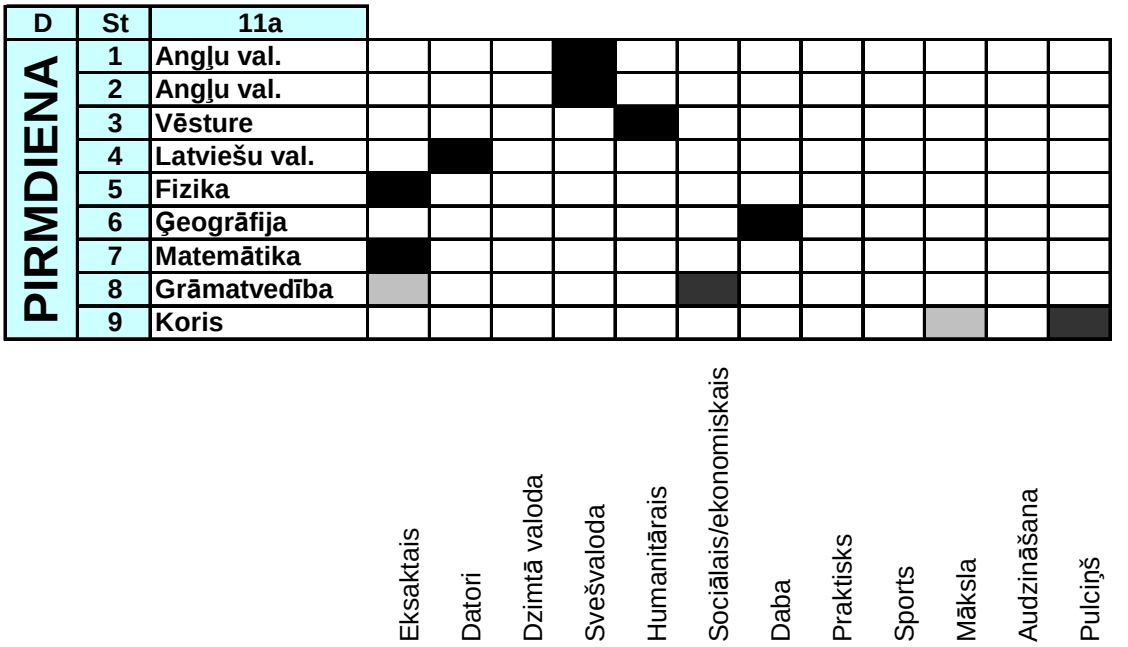
**Tabula 4. Mācību priekšmetu kategorijas**

| Nr. | Kategorija | Piemēri |
|---|---|---|
| 1 | Eksaktais | matemātika, fizika, grāmatvedība (daļēji) |
| 2 | Datori | datorzinības |
| 3 | Dzimtā valoda | latviešu valoda, krievu valoda kā dzimtā |
| 4 | Svešvaloda | angļu valoda, vācu valoda, krievu valoda |
| 5 | Humanitārais | ētika, filozofija, kultūras vēsture |
| 6 | Sociālekonomiskais | ekonomika, civilzinības, grāmatvedība (daļēji) |
| 7 | Daba | dabaszinības, ģeogrāfija |
| 8 | Praktiskais | darbmācība, mājturība |
| 9 | Sports | sports |
| 10 | Māksla | mūzika, vizuālā māksla, koris (daļēji) |
| 11 | Audzināšana | audzināšanas nodarbība |
| 12 | Pulciņš | koris (daļēji), fakultatīvās nodarbības |

Eksistējot jēdzienam "mācību priekšmeta kategorija", stundu sarakstu vienai klasei (38) varēja pārdefinēt kā virkni no mācību priekšmetu kategorijām, precīzāk, virkni no svērtu kategoriju kopām (40):

$$\mathrm{T}_k = \left\langle \left\{ \langle \sigma_{ki1}, v_{ki1} \rangle, \langle \sigma_{ki2}, v_{ki2} \rangle, \ldots \right\} \right\rangle_{i=1}^{p} \tag{40}$$

Tādējādi viens stundu saraksta periods (viena stunda) vienai klasei tika kodēts kā 12 reālu skaitļu kortežs ar kopējo summu 1, kur katrs skaitlis norādīja periodā esošo nodarbību atbilstību kādai mācību priekšmetu tematiskajai kategorijai 0..1. Praktiski lielākajā daļa gadījumu vienu stundu saraksta periodu reprezentēja vienpadsmit vērtības 0 un viena vērtība 1. Vienā dienā maksimālais iespējamais periodu skaits sistēmā tika noteikts 10, bet nedēļā ir 5 dienas, tāpēc **vienas klases vienas nedēļas stundu saraksts tika reprezentēts ar 600 skaitļiem** (5 dienas × 10 periodi dienā × 12 mācību priekšmetu kategorijas) ar katra skaitļa vidējo vērtību 1/12.



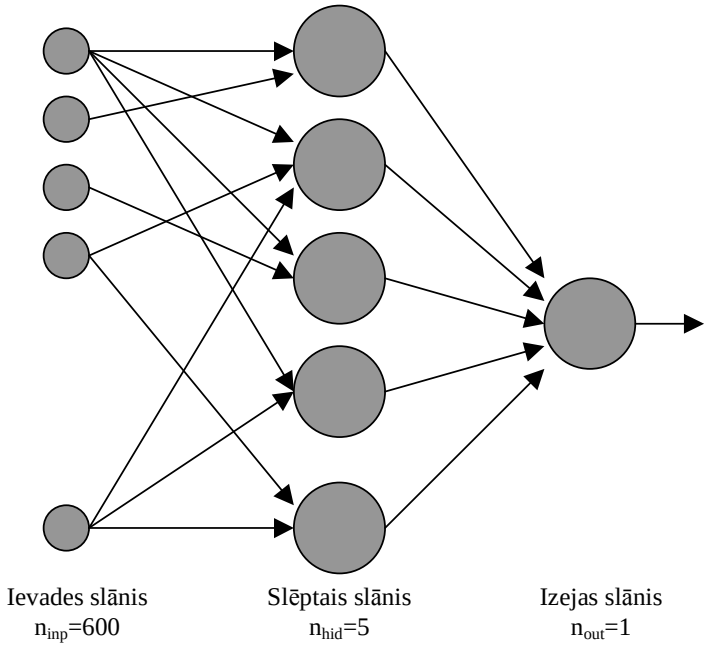

**Attēls 43. Vienas klases vienas dienas stundu saraksta reprezentācijas ar mācību priekšmetu kategorijām vizuālā interpretācija (atbilstoši formulai (40))**

### 5.2.2.2. Neironu tīklu konfigurēšana stundu sarakstu novērtēšanai

Neironu tīkli tika apmācīti, izmantojot iepriekš aprakstīto algoritmu ar nejauši ģenerēto apmācības piemēru izmantošanu (Pseidokods 2). Šajā gadījumā nejauši ģenerēts piemērs bija 600 nejauši ģenerēti skaitļi (0 vai 1) ar vieninieku un nuļļu proporciju 1:11.

Neironu tīklu apmācībai tika lietoti 10 dažādu skolu gatavie stundu saraksti, kas kopā veidoja 208 dažādus klašu stundu sarakstus. Vienas klases vienas nedēļas stundu saraksts bija pamatvienība, ar kuru strādāja neironu tīkli.

Neironu tīkli tika konstruēti un apmācīti vienas klases stundu saraksta novērtēšanai. Neironu tīkli, kas tika lietoti eksperimentos (Attēls 44), bija vairākslāņu perceptroni ar 600 ieejām ar vienu slēpto slāni ar 5 neironiem un vienu izeju. Tādējādi vienā darbināšanas reizē neironu tīklam tika padotas 600 skaitliskas vērtības, kas reprezentēja sastādītu nedēļas stundu sarakstu vienai klasei, bet neironu tīkls izrēķināja vienu skaitli – padotā stundu saraksta vērtējumu.




**Attēls 44. Eksperimentos lietoto neironu tīklu arhitektūra**

## 5.2.3. Ģenētisko algoritmu bloka izveidošana un konfigurēšana

### 5.2.3.1. Hromosomu reprezentācija

Izveidotajā sistēmā stundu saraksti netika reprezentēti kā abstraktas bitu virknes, bet gan kā C++ programmas objekti. Tādējādi bija nepieciešams izstrādāt tādus specializētus ģenētiskos operatorus, kas mācētu strādāt ar stundu sarakstiem objektu formā.

### 5.2.3.2. Ģenētiskie operatori

Ņemot vērā hromosomu reprezentāciju (kā objekts) un ģenētisko operatoru bloka specifiku (ģenētisko operatoru bloks nodrošina integritātes noteikumiem atbilstošu stundu sarakstu ģenerēšanu), mutācija tika realizēta kā vienīgais ģenētiskais operators.

Mutācijas operatora pamatā ir noteikta skaita nodarbību pārvietošana (*MutateLesson*, Pseidokods 4) katras klases stundu sarakstā, kas īsumā notiek šādos soļos (sk. arī nodaļu 5.2.1.2):

- nodarbība tiek izņemta no plāna (*UnscheduleLesson*),
- pēc nejaušības principa tiek izvēlēta nodarbības jaunā vieta (diena un pozīcija),
- no plāna tiek izņemtas visas nodarbības, kas nonāktu konfliktā, nodarbību ieliekot jaunajā vietā (*UnscheduleConflictingLessons*),

- nodarbība tiek ielikta jaunajā vietā (*SetTimeToLesson*),
- tiek izsaukta ieplānošanas operācija (*SchedulingOperation*, Pseidokods 3) visām saraksta neieplānotajām nodarbībām (*T.unscheduled_lessons*).

**Pseidokods 4. Vienas nodarbības pārvietošana mutācijas operatora ietvaros**

```
PROCEDURE MUTATELESSON(T:TimeTable,L:Lesson)
UNSCHEDULELESSON(T,L)
day:=GETRANDOMVALUE(1..5)
position:= GETRANDOMVALUE(1..MAX_POSITION)
UNSCHEDULECONFLICTINGLESSONS(T,L,day,position)
SETTIMETOLESSON(T,L,day,position)
FORALL U IN T.unscheduled_lessons
    SCHEDULINGOPERATION(T,U)
END FORALL
```

### 5.2.3.3. Izvēles funkcija

Izvēles funkcijai tika izmantota ruletes metode (*roulette-wheel selection*) komplektā ar elitismu.

### 5.2.3.4. Sākotnējās populācijas ģenerēšana

Sākotnējās populācijas ģenerēšana notika, katra indivīda (stundu saraksta) katrai nodarbībai piemērojot ieplānošanas operāciju (Pseidokods 3).

### 5.2.3.5. Derīguma funkcija

Derīguma funkcija vispārīgā gadījumā tika veidota, apvienojot četrus tiešā veidā realizētos vērtēšanas kritērijus (Tabula 3) un neironu tīklu, funkciju $\varphi(\cdot)$ (35) paplašinot uz $\varphi^*(\cdot)$ (41):

$$\varphi^*(\mathrm{T}) = \sum_{i=0}^{c} \alpha_i \varphi_i(\mathrm{T}), \qquad \textbf{(41)}$$

kur $\varphi_0$ – stundu sarakstu novērtēšanai paredzētais neironu tīkls; $\alpha_0$ – neironu tīkla svars kopējā vērtējumā.

### 5.2.3.6. Konfigurācijas parametri

Eksperimentos izmantotās ģenētisko algoritmu sistēmas galvenie konfigurācijas parametri:

- Populācijas lielums: 10.
- Elitisma pakāpe: 1.
- Katrā iterācijā 3 indivīdi tika nomainīti, bet 7 pārgāja uz nākošo paaudzi.
- Mutācijas faktors: 20%.
- Neirona tīkla svars $\alpha_0$ kopējā vērtējumā ((41),(43)) bija ļoti neliels un nepārsniedza 0.01. Tik neliela $\alpha_0$ vērtība tika izvēlēta tāpēc, ka eksperimentāli tika noskaidrots, ka lielāka vērtība noved pie sistēmas darbības pasliktināšanās.

## 5.3. Neironu tīklu un ģenētisko algoritmu hibrīdā modeļa novērtēšana

Iespējamais ieguvums no neironu tīkliem, kas tiktu ievietoti derīguma funkcijā, tādējādi atvieglojot vērtēšanas noteikumu definēšanu, būtu aplūkojams no 2 aspektiem:

1. **Aizvietot**, kādu no noteikumiem ar neironu tīklu, tādā veidā atvieglojot noteikumu definēšanu kopumā.
2. Tā kā visu noteikumu definēšana ir grūti paveicama, un tas ir praktiski neiespējams, neironu tīkls varētu **papildināt** jau esošos stingri definētos noteikumus, tādējādi uzlabojot stundu saraksta vērtēšanu.

Neironu tīklu ieguldījumu atbilstoši otrajam aspektam būtu ļoti grūti novērtēt, jo tad būtu nepieciešama grandioza ekspertu vērtējuma nodrošināšana. Tāpēc tālāk piedāvātais modelis tiks pārbaudīts atbilstoši pirmajam aspektam – kā spējīgs aizvietot kādu no iepriekš definētiem noteikumiem. Šim nolūkam ir izveidota speciāla metodika, kas dod iespēju iegūt formālu novērtējumu.

### 5.3.1. Novērtēšanas metodika

Piedāvātais neironu tīklu un ģenētisko algoritmu hibrīdais modelis tika pārbaudīts, izmantojot speciālu, autora izstrādātu, metodiku, kas deva iespēju ar formāliem līdzekļiem parādīt piedāvātā modeļa efektivitāti. Metodikas idejas pamatā ir salīdzināt stundu sarakstus, kas iegūti, izmantojot divu veidu derīguma funkcijas (attiecībā pret katru tieši definēto kritēriju), pirmkārt, tādas, kur kritērijs aizvietots ar neironu tīklu, otrkārt, tādas, kur attiecīgais kritērijs vienkārši izmests no derīguma funkcijas. Modeļa pārbaude notika pēc kārtas attiecībā pret visiem četriem vērtēšanas kritērijiem (Tabula 3).

Piedāvātās modeļa novērtēšanas metodikas ietvaros tika ieviesti 3 jauni jēdzieni:

- **Nominālā derīguma funkcija** $\varphi(\cdot)$ (*the nominal fitness function*) (35) – derīguma funkcija, kas ietver visus 4 tieši realizētos vērtēšanas kritērijus, bet neietver neironu tīkla doto vērtējumu.
- **Reducētā derīguma funkcija** $\varphi^f(\cdot)$ (*the reduced fitness function*) (42) – nominālā derīguma funkcija, no kuras izmests vērtējums pēc kāda no kritērijiem *f*.
- **Mīkstinātā derīguma funkcija** $\varphi^{f+}(\cdot)$ (*the softened fitness function*) (43) – reducētā derīguma funkcija, kur izmestā kritērija vērtējuma *f* vietā ir stājies neironu tīkla vērtējums.

Reducētā derīguma funkcija:

$$\varphi^f(\mathrm{T}) = \sum_{i=1}^{c} \begin{cases} 0; & \text{if } i = f \\ \alpha_i \varphi_i(\mathrm{T}); & \text{otherwise} \end{cases}, \qquad \textbf{(42)}$$

kur *f* – izmetamā vērtējuma kritērija numurs.

Mīkstinātā derīguma funkcija:

$$\varphi^{f+}(\mathrm{T}) = \varphi^f(\mathrm{T}) + \alpha_0 \varphi_0(\mathrm{T}), \qquad \textbf{(43)}$$

kur $\varphi_0(\cdot)$ – stundu sarakstu novērtēšanai paredzētais neironu tīkls; $\alpha_0$ – neironu tīkla svars kopējā vērtējumā.

Visām derīguma funkcijām atbilstošās formulas (35),(42),(43) ir formulas (41) speciālgadījumi.

Attēls 45 vizualizē iepriekš definētos derīguma funkciju veidus un parāda galveno ideju modeļa novērtēšanai, kuru precīzi parāda Pseidokods 5:

- Nominālā derīguma funkcija tiek uzskatīta par kvalitātes kritēriju.
- Tiek darbināta stundu saraksta sastādīšanas sistēma, pārmaiņus izmantojot reducēto un mīkstināto derīguma funkciju (attiecībā uz dažādiem tieši definētajiem vērtēšanas kritērijiem).
- Attiecīgo reducēto un mīkstināto derīguma funkciju sistēmu ģenerētie stundu saraksti tiek salīdzināti, izmantojot nominālo derīguma funkciju.

Piedāvātā modeļa novērtēšanas metode balstās uz nominālo derīguma funkciju kā vērtēšanas kritēriju, kas no vienas puses var likties ne visai objektīvi, tomēr tas ir ērts veids, kā nodrošināt formalizētu modeļa novērtējumu, jo neatkarīgs ekspertu vērtējums prasītu ļoti daudz resursu.

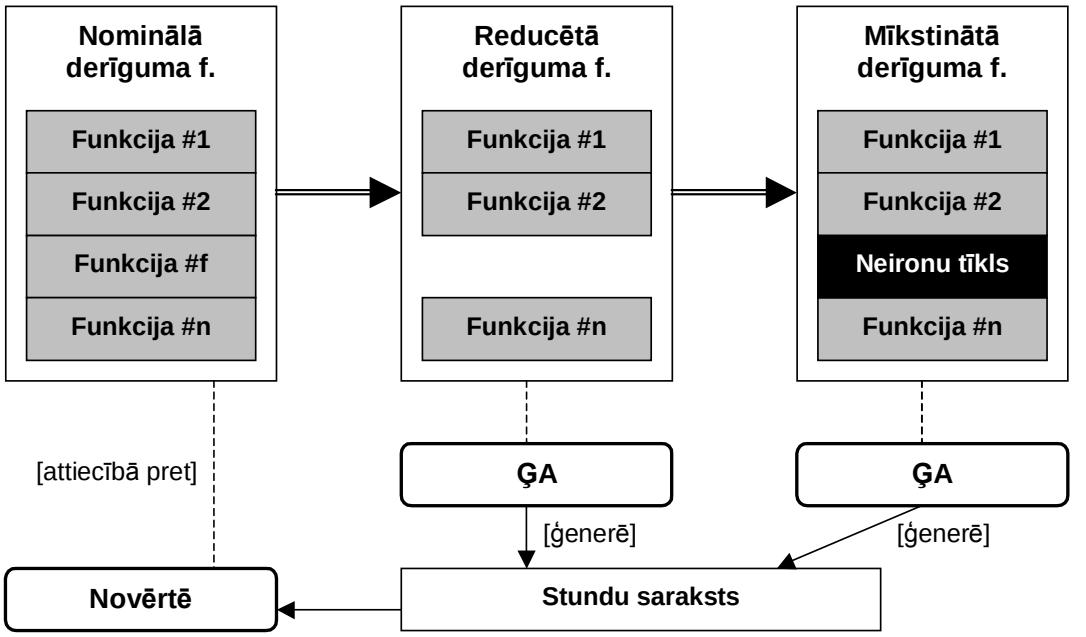
**Pseidokods 5. Eksperimentu veikšanas shēma neironu tīklu un ģenētisko algoritmu hibrīdā modeļa novērtēšanai**

```
PROCEDURE RUNEXPERIMENTS(FILE:FileName,
```

```
                TRAINING_SET:ReadyClassTimeTables,
                RAW_SET:UnscheduledTimetables,
                EVAL_FUNCT_COUNT:Integer)
NET:NeuralNetwork
TT,TTR,TTS:TimeTable
EvalReduced,EvalSoftened:Real
GAConfig:GAConfiguration
NNConfig:NeuralNetworkConfiguration
RANDOM_RATE:0..1
REPEAT
    NNConfig:=GENERATENNCONFIGURATION()
    RANDOM_RATE:=CHOOSEVALUEFROM({0,0.2})
    NET:=CREATENEURALNETWORK(NNConfig)
    TRAINNETWORK(NET,TRAINING_SET,RANDOM_RATE)
    GAConfig:=GENERATEGACONFIGURATION()
    TT:=CHOOSEPATTERNFROM(RAW_SET)
    FOR func:=1 TO EVAL_FUNCT_COUNT
        TTR:=RUNGAWITHREDUCEDFITNESSFUNCTION(TT,GAConfig,func)
        TTS:=RUNGAWITHSOFTENEDFITNESSFUNCTION(TT,GAConfig,func,NET)
        EvalReduced:=EVALUATEWITHNOMINALFITNESSFUNCTION(TTR)
        EvalSoftened:=EVALUATEWITHNOMINALFITNESSFUNCTION(TTS)
        REGISTER(FILE,NNConfig,GAConfig,EvalReduced,EvalSoftened)
    END FOR
UNTIL BROKENBYUSER()
```

**Komentāri pie algoritma** (Pseidokods 5).

- ***RunGAWithReducedFitnessFunction*** – darbina ĢA sistēmu, par derīguma funkciju izmantojot reducēto derīguma funkciju $\varphi^{f}(\cdot)$ (42), atgriež gatavu stundu sarakstu.
- ***RunGAWithReducedFitnessFunction*** – darbina ĢA sistēmu, par derīguma funkciju izmantojot mīkstināto derīguma funkciju $\varphi^{f+}(\cdot)$ (43), atgriež gatavu stundu sarakstu.
- ***Register*** – Pieraksta eksperimentu rezultātus tālākai analīzei.
- ***TRAINING_SET*** – Gatavu stundu sarakstu kopums, kas tika lietots neironu tīkla apmācībai.
- ***RAW_SET*** – dažādu skolu nesastādītu stundu sarakstu kopums. Katrs šāds saraksts sastāv no neieplānotām nodarbībām un būtu pakļaujams stundu saraksta sastādīšanai.




**Attēls 45. Derīguma funkciju veidi neironu tīklu un ģenētisko algoritmu hibrīdā modeļa novērtēšanai (atbilst algoritmam Pseidokods 5)**

## 5.3.2. Eksperimentu rezultāti un to analīze

### 5.3.2.1. Eksperimentu mērķis

Eksperimentu rezultāti bija dažādu piedāvātās sistēmas ģenerēto stundu sarakstu, kas iegūti, darbinot sistēmu, izmantojot reducēto vai mīkstināto derīguma funkciju, vērtējumi.

Eksperimentu rezultātu analīzes būtība ir pārliecināties, vai stundu saraksti, kas ģenerēti, izmantojot mīkstināto derīguma funkciju, ir labāki par tiem, kas ģenerēti, izmantojot reducēto derīguma funkciju, t.i., vai neironu tīkls ir spējīgs aizvietot kādu no tieši definētajiem stundu saraksta vērtējuma kritērijiem.

Eksperimentu papildus uzdevums bija novērtēt autora piedāvāto neironu tīklu kontrolētās apmācības metodi (nodaļa 4.4.3), izmantojot pēc nejaušības principa ģenerētus apmācības piemērus.

### 5.3.2.2. Eksperimentu rezultāti

Eksperimentu galvenais uzdevums bija parādīt, ka neironu tīkls spēj noteiktā mērā aizvietot tieši definētus stundu saraksta vērtēšanas kritērijus.

| | | | |
|---|---|---|---|
| 10a | 1 | 0 | |
| 10a | 1 | 1 | [42472] Tovanceva L. Kultūras vēsture |

| | | | |
|---|---|---|---|
| 10a | 1 | 2 | [42352] Vesele S. Individuālā vai grupu darba vadība |
| 10a | 1 | 3 | [42371] Malkovska O. Matemātika |
| 10a | 1 | 4 | [42450] Hmeļkova T. Krievu valoda (dzimtā) |
| 10a | 1 | 5 | [42503] Vesele S. Sports |
| 10a | 1 | 6 | [42578] Baikova A. Vācu valoda |
| 10a | 1 | 7 | [42432] Boldāne I. Lietišķā informātika |
| 10a | 1 | 8 | [42579] Baikova A. Vācu valoda |
| 10a | 1 | 9 | |
| 10a | 1 | 10 | |
| 10a | 2 | 0 | |
| 10a | 2 | 1 | [42296] Tovanceva L. Kultūras vēsture |
| 10a | 2 | 2 | [42471] [Vakance] . Vācu valoda |
| 10a | 2 | 3 | [42412] Hmeļkova T. Literatūra (cittautu māc.val.) |
| 10a | 2 | 4 | [42541] Puzorenoka K. Dabaszinības |
| 10a | 2 | 5 | [42428] Tovanceva L. Biznesa ekonom.pamati |
| 10a | 2 | 6 | [42416] Supe A. Mājturība |
| 10a | 2 | 7 | [42592] Malkovska O. Matemātika |
| 10a | 2 | 8 | [42526] Laimiņa L. Angļu valoda |
| 10a | 2 | 9 | [42460] Tovanceva L. Individuālā vai grupu darba vadība |
| 10a | 2 | 10 | |

**Attēls 46. Ar neironu tīklu un ģenētisko algoritmu hibrīdo sistēmu iegūta stundu saraksta paraugs**

Eksperimentu rezultātu kopsavilkums attēlots sekojošajā tabulā (Tabula 5), bet vizualizēts diagrammā (Attēls 47). Eksperimentu rezultāti parāda neironu tīklu spēju aizvietot katru no 4 realizētajiem vērtēšanas kritērijiem (Tabula 3).

Bez tam tika iegūti rezultāti pievienojot neironu tīklu vērtējumu papildus visiem 4 kritērijiem (rinda "*N*"), tātad, bez kāda kritērija izmešanas. Šis rezultāts tika iegūts nevis modeļa novērtēšanai, bet gan lai novērtētu pašu metodiku vai, precīzāk, apstiprinātu tās pareizību (nodaļa 5.3.1).

Melnās joslas diagrammā (Attēls 47) un 2. kolonna tabulā (Tabula 5) ar nosaukumu "ANN=1" apzīmē iegūto stundu sarakstu kvalitāti, lietojot neironu tīklus, t.i., izmantojot mīkstināto derīguma funkciju ĢA sistēmas darbināšanā. Attiecīgi gaišās joslās diagrammā un 3. kolonna tabulā ar nosaukumu "ANN=0" apzīmē iegūto stundu sarakstu kvalitāti, nelietojot neironu tīklus, t.i., izmantojot reducēto derīguma funkciju ĢA sistēmas darbināšanā. Tā kā derīguma funkcija darbojas kā sodīšanas funkcija (*penalty function*), tad mazākās vērtējuma vērtības un attiecīgi īsākās joslas diagrammā nozīmē labāku vērtējumu.

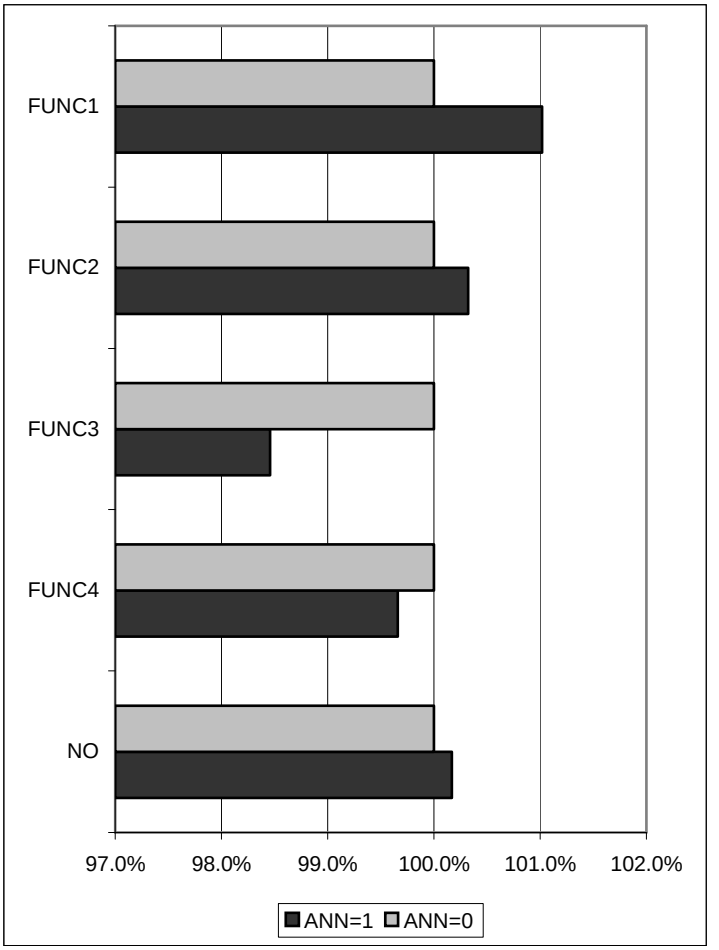
Eksperimentu rezultāti parāda, ka šādi apmācīts neironu tīkls spēj noteiktā mērā aizvietot kritēriju nr. 3 (šim kritērijam melnā josla ir īsāka nekā gaišā). Eksperimentu rezultātu precizējums kritērijam nr. 3 parādīts diagrammā (Attēls 48).

Tā kā neironu tīkls reprezentē noteiktus kritērijus stundu sarakstu vērtēšanai, tad ir visai loģiski, ka tas neuzrāda pozitīvos rezultātus visu tieši definēto kritēriju aizvietošanā.

Negatīvs neironu tīklu pielietošanas rezultāts gadījumā bez kritēriju izmešanas ("*N*") ir pats par sevi saprotams, jo par salīdzinošo funkciju (ANN=0) tika izmantota nominālā derīguma funkcija. Tā kā pati nominālā derīguma funkcija jau tiek lietota kā etalons vērtēšanai, tad jebkurš blakusefekts pie šīs funkcijas neglābjami novedīs pie rezultāta pasliktināšanās vai vismaz nedos uzlabojumu. Gadījumam bez kāda kritērija izmešanas ("*N*") nav nekādas praktiskas nozīmes no optimizācijas viedokļa, tas uzskatāms tikai par papildus ilustrāciju modeļa novērtēšanai pielietotajai metodei.

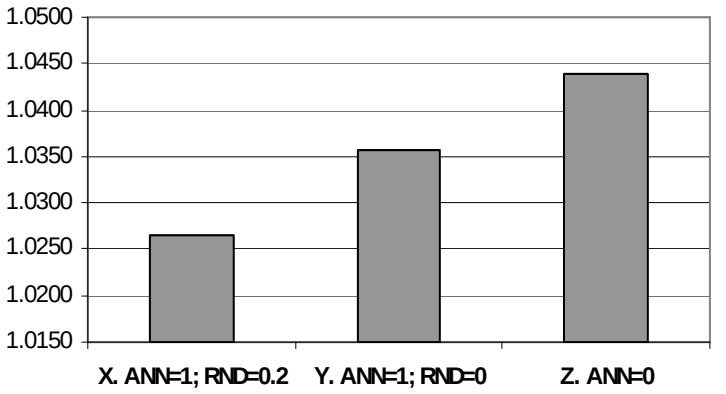
**Tabula 5. Eksperimentu rezultātu kopsavilkums**

| Aizvietojamā funkcija (kritērijs) | Izmantojot mīkstināto derīguma funkciju $\varphi^{f+}$ (ANNS=1) | Izmantojot reducēto derīguma funkciju $\varphi^{f}$ (ANNS=0) |
|---|---|---|
| 1 | 0.97863347 | 0.96876746 |
| 2 | 1.01990608 | 1.01663669 |
| 3 | 1.02775411 | 1.04385954 |
| 4 | 0.95626831 | 0.95952815 |
| N | 0.96297805 | 0.96134982 |




**Attēls 47. Eksperimentu rezultātu kopsavilkums (atbilst Tabula 5 datiem), normalizējot tos attiecībā pret stundu sarakstu vērtējumiem, kas ģenerēti, izmantojot reducēto derīguma funkciju**

Eksperimentu rezultāti attiecībā pret kritēriju nr. 3 (Tabula 5) tika aplūkoti precīzāk, ņemot vērā arī pēc nejaušības principa ģenerēto apmācības piemēru izmantošanas nozīmi un tādējādi apstiprinot autora piedāvātās neironu tīklu apmācības metodes (nodaļa 4.4.3) noderīgumu stundu sarakstu novērtēšanas gadījumā. Attēls 48 parāda tabulas (Tabula 5) rindas, kas atbilst kritērijam nr. 3, izvērsumu, izšķirot nejaušu apmācības piemēru ģenerēšanas faktora apmēru apmācības procesā.




**Attēls 48. Eksperimentu rezultātu kopsavilkums stundu sarakstu vērtēšanas kritērijam nr. 3**

Attēlā (Attēls 48) parādītie rezultāti nosaka 3 rezultātu grupas, kuras nosauksim kā X, Y un Z:

- **Grupa X** (vērtība 1.0265). Derīguma funkcijā tika izmantots neironu tīkls (mīkstinātā derīguma funkcija (43)), un nejaušības faktoram tika noteikta vērtība 0.2 – tātad apmācība, izmantojot nejauši ģenerētus apmācības piemērus.
- **Grupa Y** (vērtība 1.0356). Derīguma funkcijā tika izmantots neironu tīkls (mīkstinātā derīguma funkcija (43)), bet nejaušības faktoram tika noteikta vērtība 0 – tātad apmācība notika tikai ar pozitīviem apmācības paraugiem, neizmantojot nejauši ģenerētus apmācības piemērus.
- **Grupa Z** (vērtība 1.0439). Derīguma funkcijā neironu tīkls netiek izmantots (reducētā derīguma funkcija (42)).

Grupu X un Y labākie rezultāti nekā grupai Z norāda uz to, ka neironu tīkla risinājums ir lietojams, ka daļēji aizvietotu stundu saraksta vērtēšanas kritēriju nr. 3.

Tā kā grupa X uzrāda labākus rezultātus nekā grupa Y, var teikt, ka pēc nejaušības principa ģenerēto apmācības piemēru izmantošana neironu tīklu apmācībai ir izmantojama šajā gadījumā.

#### 5.3.2.3. Eksperimentu rezultātu analīze

Eksperimentu rezultāti ļauj secināt:

- Neironu tīkls ir lietojams, lai aizvietotu novērtēšanas kritēriju nr. 3.
- Pēc nejaušības principa ģenerēto apmācības piemēru izmantošanas metode ir lietojama gadījumā, apmācot neironu tīklus ar stundu sarakstiem.

Tai pat laikā eksperimentu gaitā tika sastaptas šādas problēmas:

- Nelielais neironu tīkla pielietojuma efekts uz rezultātiem izskaidrojams ar nelielo neironu tīkla komponentes svaru kopējā vērtējumā $\alpha_0$=0.01. Eksperimentāli tika secināts, ka lielākas vērtības derīguma funkciju strauji izārda un padara nelietojamu. Tas nozīmē, ka neirona tīklu pielietojums stundu sarakstu vērtēšanā pēc aprakstītās metodes ir pietiekoši nestabils.
- Neironu tīkli tik apmācīti uz stundu sarakstiem, kas tika reprezentēti kā atsevišķu klašu stundu sarakstu kopumi, t.i., tika pieņemts, ka dažādu līmeņu klašu stundu saraksti ir veidoti pēc vienādiem principiem. Diemžēl apmācības paraugu kopums bija pārāk mazs, lai nodrošinātu dažādu klašu līmeņu diferencēšanu (kopā apmācībai bija pieejami 208 klašu stundu saraksti, t.sk., tikai 9 saraksti pirmajām klasēm).

### 5.3.3. Secinājumi par eksperimentu rezultātiem

Eksperimentu rezultāti ir parādījuši, ka neironu tīkli ir spējīgi aizvietot tieši definētus vērtēšanas kritērijus, turklāt veicot to apmācību uz nepilnīgas apmācības kopas, kas satur tikai pozitīvus piemērus.

Neironu tīklu veiksmīga izmantošana līdzīgu problēmu risināšanā prasa pamatīgu problēmas specifisku izpēti. Autora veiktais darbs eksperimentu sagatavošanā veikšanā parādīja, ka neironu tīklu pielāgošana piedāvātā modeļa ietvaros tik specifiskai problēmai kā stundu sarakstu novērtēšana prasa lielu laika resursu ieguldījumu, bet tas ir arī saprotams, jo stundu sarakstu sastādīšana ir saistīta ar liela datu apjoma apstrādi.

Lai uzlabotu aprakstīto stundu saraksta sastādīšanas modeli, būtu vēlams paveikt šādas darbības:

- Būtu jādiferencē dažādu līmeņu klases (pirmās, otrās utt.), kā tas jau ir ticis piedāvāts [zuters06b]. Tam būtu nepieciešama daudz plašāka pieejamā neironu tīklu apmācības kopa.
- Būtu jāveic eksperimenti, lai nodrošinātu optimālāku stundu saraksta reprezentāciju apstrādei ar neironu tīkliem.

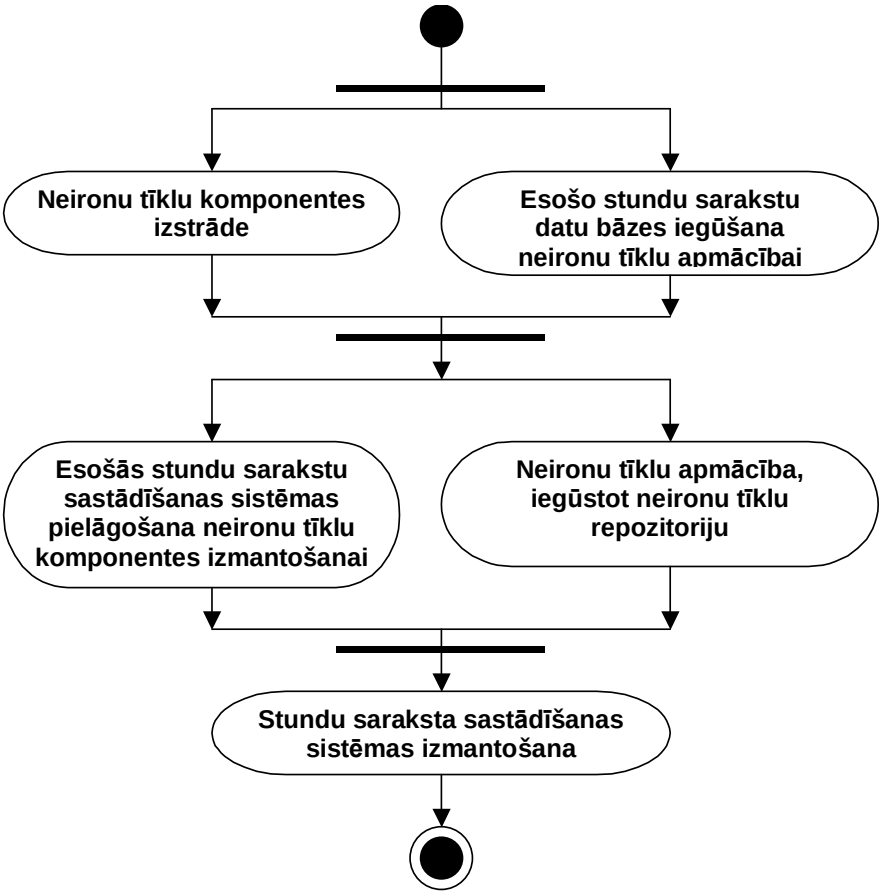
Kaut arī autors pēc veiktajiem eksperimentiem vēl nav piedāvājis gatavu, precīzi noteiktu instrukciju stundu sarakstu sastādīšanai, izmantojot piedāvāto modeli, tomēr eksperimenti ir parādījuši, ka neironu tīkls ir kaut kādā mērā spējīgs risināt pat tik neordinārās problēmas kā skolu stundu sarakstu novērtēšana.

## *5.4. Piedāvātā modeļa praktiskas pielietošanas iespējas*

Attēls 41 parāda, ka no stundu saraksta sastādīšanas procesa viedokļa piedāvātais modelis skar tikai stundu saraksta novērtēšanu (respektīvi, derīguma funkciju ģenētisko algo-

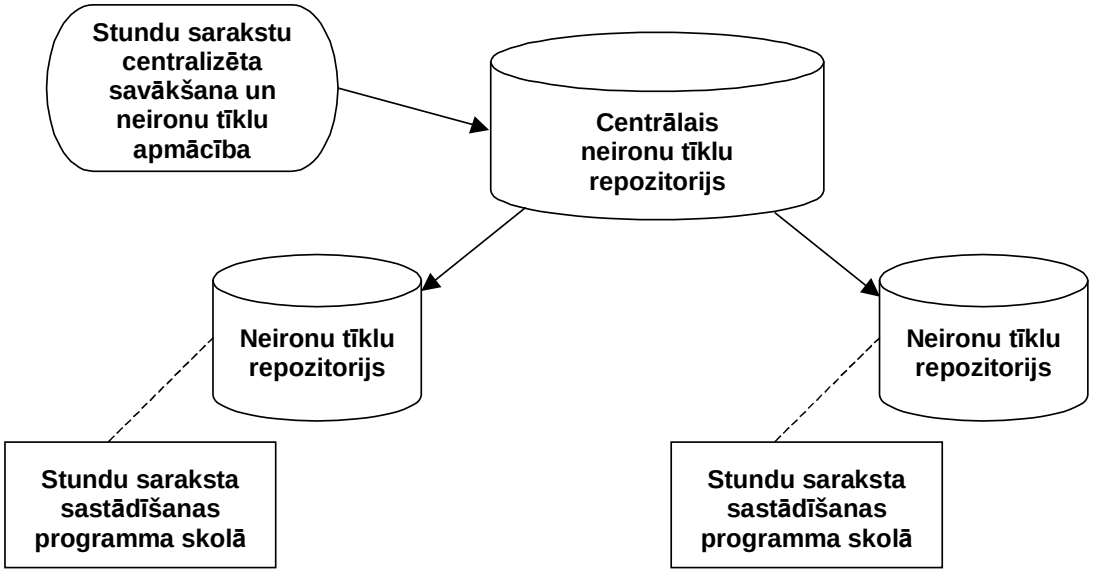
ritmu infrastruktūrā). Tādējādi var uzskatīt, ka neironu tīklu bloks ir pietiekoši neatkarīgs, un tā pievienošana kādai esošai, uz ģenētiskajiem algoritmiem balstītai, stundu saraksta sastādīšanas sistēmai neprasītu būtisku šīs sistēmas pārveidi vai pielāgošanu. Bez tam neironu tīklu bloks netraucētu esošās sistēmas darbam, jo to varētu viegli atslēgt, kā tas arī ir ticis darīts eksperimentos (koeficients $\alpha_0$ formulā (41)).

Otrs, ne mazāk svarīgs, uzdevums ir apjomīga esošo stundu sarakstu kopuma savākšana un neironu tīklu apmācīšana uz šiem stundu sarakstiem, kas būtu jāveic centralizēti. Kaut arī stundu saraksta sastādīšanas sistēmai pie dotās modeļa izstrādes pakāpes pietiek ar vienu neironu tīklu, varētu iegūt vairākus neironu tīklus kā dažādus variantus, lai pēc tam varētu eksperimentāli uz vietām skolās noskaidrot, kurš no tiem ir labāks, tādējādi nodrošinot sistēmas attīstību. Piedāvātā modeļa iespējamā realizācijas shēma redzama attēlā (Attēls 49).




**Attēls 49. Piedāvātā modeļa realizācijas shēma**

Apmācīto neironu tīklu repozitorijs varētu tikt izplatīts pa skolām līdzīgi kā antivīrusu programmas datu bāze (Attēls 50).




**Attēls 50. Piedāvātā modeļa praktiskas pielietošanas shēma**

# Noslēgums

Promocijas darbā autors piedāvājis un parādījis neironu tīklu pielietojumu relatīvi neierastā lomā – kā derīguma funkcijas komponenti uz ģenētiskajiem algoritmiem bāzētā stundu saraksta sastādīšanas modelī.

Pētījumu gaitā autors ieguldījis ļoti lielu eksperimentālu darbu, kas saistīts ar apjomīgas oriģinālprogrammatūras izveidi.

Autors savos pētījumos ir parādījis, ka neironu tīkli ir spējīgi aizvietot tieši definētus vērtēšanas kritērijus un noteiktā līmenī vērtēt dažādus objektus – arī gadījumos, ja tiek lietota nepilnīga un tendencioza apmācības kopa. Tai pašā laikā pētījumu rezultāti vēl nedod gatavu recepti, kā autora piedāvātais modelis būtu praktiski pielietojams reālu stundu sarakstu plānošanas sistēmu izstrādē.

Lielākais darba ieguldījums no stundu saraksta sastādīšanas viedokļa ir **mehānisma izstrāde, kas dod iespēju iegūt stundu sarakstu vērtēšanas kritērijus, neveicot pašu kritēriju formalizēšanu**. Ar šāda mehānisma pielietošanu tiek automatizētas tādas cilvēkam veicamās darbības, kuras ir uzskatāmas par radošām.

Tomēr par veikto eksperimentu un attiecīgi arī šī darba galveno ieguldījumu nebūtu uzskatāms piedāvātais stundu saraksta sastādīšanas modelis kā tāds, bet gan dažādas piedāvātās **neironu tīklu uzbūves, darbināšanas, apmācības un pielietošanas metodes**. Līdz ar to neironu tīklu izmantošanu stundu sarakstu vērtēšanai varētu uzskatīt par ilustrāciju atziņai, ka ar neironu tīkliem var darīt “pat šādas” – ļoti neordinārās un netipiskas lietas.

Izmantojot neironu tīklu īpašību iegūt savu darboties spēju apmācības rezultātā no paraugiem, kā arī neironu tīklu relatīvi lielo noturību pret neprecīziem datiem, autora piedāvāto ideju pielietošana dažādu citu problēmu risināšanā varētu nodrošināt lielāku ieguvumu nekā neironu tīklu iesaistīšana aprakstītajā stundu saraksta sastādīšanas modelī.

Autora mērķis ir turpināt strādāt pie neironu tīklu paradigmā sakņotām informācijas apstrādes metodēm, kas orientētas uz problēmām, kurām ne tikai nav pieejami precīzi kritēriji lēmumu pieņemšanai, bet arī kuru reprezentējošie paraugi tiešā veidā neveido pilnu ainu par problēmas raksturu, tādējādi neļaujot veikt apmācību ar klasiskajām kontrolētās apmācības metodēm.

# Darbā lietoto jēdzienu definīcijas un saīsinājumu skaidrojumi

## *Darbā lietoto jēdzienu definīcijas*

**Aktivitātes stāvoklis** (*activation state*) ir vērtība, kas tiek izrēķināta ar ***aktivizācijas funkciju*** un kas piedalās ***izejas*** vērtības rēķināšanā, bet bieži vien tiek lietota par pašu izejas vērtību.

**Aktivizācijas funkcija** (*activation function*) ir neirona komponente – funkcija, kura, izmantojot summēšanas funkcijas vērtību NET, izrēķina t.s. aktivitātes stāvokli, kas bieži ir arī neirona izejas vērtība. Aktivizācijas funkcijai ir ļoti liela ietekme uz neirona un visa neironu tīkla skaitļošanas spēju. Aktivizācijas funkcija var būt lineāra, vai nelineāra.

**Apmācība** (*learning*; *training*) ir process, kurā neironu tīkls iegūst spēju veikt noteiktu uzdevumu. Apmācība notiek pēc noteikta algoritma, kas ir svarīgākā raksturiezīme, kas atšķir dažādus neironu tīklu modeļus.

**Apmācība ar konkurenci** (*competitive learning*) ir neironu tīklu ***apmācības*** princips, kurā apmācības apjoms katrā solī ir atkarīgs no citu šī paša ***slāņa neironu*** dotās izejas. Tādējādi slāņa neironi konkurē savā starpā. Tiek izmantots ***Kohonena tīkla*** apmācībā.

**Apmācība ar skolotāju** (*training with a teacher*) ir ***apmācības algoritmu*** kategorija, kad apmācība tiek veikta uz piemēriem, kas ir paraugu pāri formā (paraugs, pareizā atbilde). "Simboliskais" skolotājs ir tas, kurš novērtē, cik pareizi tīkls atbildējis uz doto paraugu. Šī darba ietvaros šis termins tiek lietots kā sinonīms terminam **kontrolētā apmācība**.

**Apmācība bez skolotāja** (*training without a teacher*) ir **apmācības algoritmu** kategorija, kuru apmācības paraugi (pretēji apmācībai ar skolotāju) nesatur pareizās atbildes, līdz ar ko apmācība ir zināmā mērā pašorganizējošs process.

**Apmācības algoritms** (*learning algorithm*; *training algorithm*) ir algoritms pēc kura notiek atsevišķa neirona vai visa neironu tīkla apmācība. Apmācības algoritms ir viena no svarīgākajām neironu tīkla komponentēm.

**Apmācības kopa** (*training set*) ir neironu tīklam padodamo paraugu kopa, kas tiek izmantota neironu tīkla apmācībā. Lai neironu tīkls iegūtu spēju risināt noteiktu problēmu, apmācības kopai ir jābūt problēmu reprezentējošai.

**Aproksimācijas kļūda** (*approximation error*) ir neironu tīkla dotā kļūda uz paraugiem, uz kuriem tas ticis apmācīts. Sk. ***tendenciozitāte***, ***iegaumēšana***, ***novērtējuma kļūda***.

**Derīguma funkcija** (*fitness function*) ir metode ***indivīda*** novērtēšanai ***ģenētiskā algoritma*** ietvaros. Derīguma funkcijas noteiktais vērtējums ir pamatā indivīdu izvēlei, kas nodrošina ĢA darbību.

**Dispersija** (*variance*) (apmācāmu modeļu, t.sk. neironu tīklu kontekstā) ir neironu tīkla apmācības efektivitātes novirze, mainot apmācības piemērus. Dispersija norāda uz neironu tīkla ***vispārināšanas*** spēju, t.i., vai neironu tīkls pielāgojas apmācības piemēriem, neņemot vērā apmācības datu specifiku. Dispersija savā ziņā reprezentē apmācības piemēru neatbilstību mērķa funkcijai un to, cik lielā mērā šī neatbilstība pāriet uz neironu tīklu. Tādējādi dispersiju var uztvert kā ***novērtējuma kļūdu*** (*estimation error*).

**Divvirzienu asociatīvā atmiņa** (*bidirectional associative memory*) ir neironu tīklu modelis, Hopfīlda modeļa paplašinājums, kas realizē heteroasociāciju.

**Elitisms** (*elitist selection*) ir ***indivīdu izvēles*** princips nākamajai paaudzei ***ģenētiskajā algoritmā***, kas nozīmē to, ka labākajiem (derīgākajiem) indivīdiem tiek garantēta to pārkopēšana nākamajā paaudzē.

**Epoha** (*epoch*) ir ***apmācības*** procesa daļa, kurā neironu tīkls tieši vienu reizi tiek apmācīts uz visiem ***apmācības piemēriem***.

**Ģenētiskie algoritmi** (*genetic algorithms*, *GA*) ir problēmu risināšanas stratēģija, kas balstās bioloģiskās evolūcijas principiem (paaudžu nomaiņa, konkurence starp populācijas indivīdiem). Ģenētiskie algoritmi ir metožu kopums, lai optimizācijas un meklēšanas problēmām atrastu pareizus vai tuvinātus risinājumus.

**Ģenētiskie operatori** (*genetic operators*) ir ***ģenētiskā algoritma*** mehānismi, kas nodrošina jaunu (savādāku) indivīdu ģenerēšanu uz viena vai vairāku esošo indivīdu bāzes.

**Heiristika** (*heuristic*) ir specifiska metode vai paņēmiens, kas noteiktos gadījumos uzlabo algoritma darbību, tomēr nav teorētiski pamatots un tādējādi nav lietojams vispārīgā gadījumā.

**Hopfīlda tīkls** (*Hopfield network*) ir neironu tīklu modelis, kas realizē autoasociāciju. Klasiskajā variantā darbojas ar diskrētām vērtībām. Īpašs ar to, ka ***apmācība*** notiek vienā solī, bet atpazīšanas process ir iteratīvs.

**Hromosomas** (*chromosomes*) – ***ģenētiskā algoritma*** potenciālo risinājumu (***indivīdu***) abstraktās reprezentācijas, kas tiek izmantotas aprēķinos.

**Ieeja** (*input*) ir ***neironā*** ienākošais signāls (vērtība), kas ir vai nu ārējais neironu tīklā ienākošais signāls, vai nu cita neirona izejas signāls. Neironam var būt vairākas ieejas.

**Ieejas slānis** (*input layer*) ir neironu ***slānis***, kurš tieši saistīts ar apkārtējo vidi, no kurienes tiek uzstādītas ***ieejas*** vērtības.

**Iegaumēšana** (*memorization*) ir neironu tīkla spēja precīzi atpazīt paraugus, ar kuriem tas ir ticis apmācīts.

**Ierobežojumu izpildīšanās problēma** (*constraint satisfaction problem, CSP*) ir problēma, kuras atrisināšana prasa noteiktu iepriekš uzstādītu noteikumu (ierobežojumu) izpildīšanu.

**Indivīds** (*individual*) ir viens no potenciālajiem problēmas risinājumiem ***ģenētiskā algoritma*** ietvaros.

**Izeja** (*output*) ir ***neirona*** darbināšanas rezultātā izskaitļotā vērtība, kas ir vai nu visa neironu tīkla izejošais signāls, vai arī kalpo kā ieejas signāls citos neironos. Neironam var būt tikai viena izeja. Atbilst aksonam bioloģiskajā neironā.

**Izejas funkcija** (*output function*). Neirona komponente – funkcija, kura, izmantojot aktivitātes funkcijas doto vērtību, izrēķina neirona izejas vērtību. Parasti izejas funkcija atsevišķi netiek izšķirta, bet ir inkorporēta aktivitātes funkcijā.

**Izejas slānis** (*output layer*) ir ***neironu slānis***, kas tieši saistīts ar apkārtējo vidi, kur tiek aizvadītas ***izejas*** vērtības. Izejas slānis ir slānis, kura neironu izejas ir arī visa neironu tīkla izejas.

**Izplatīšanās funkcija** – sk. ***summēšanas funkcija***.

**Izvēle** (*selection*) ir ***ģenētiskā algoritma*** mehānisms, kas nosaka, kuri ***indivīdi*** pāries nākamajā paaudzē, bet kuri nē.

**Kohonena tīkls** (*Kohonen network*) ir pašorganizējošs neironu tīkla modelis, kas izmanto ***apmācību ar konkurenci*** un ***bez skolotāja***. Tiek saukts arī par Pašorganizējošo karti (*self-organizing map, SOM*). Realizē paraugu klāsterizāciju.

**Konekcionisms** ir skaitļošanas sistēmu uzbūves princips, kad sistēma sastāv no relatīvi liela daudzuma savstarpēji savienotu elementu. Apmācība un konekcionisms ir divas īpašības, kas piemīt jebkuram neironu tīklam.

**Kontrolētā apmācība** (*supervised learning*) ir neironu tīklu apmācības paradigma, kad ***apmācības*** procesā bez ieejas paraugiem neironu tīkla svaru izmaiņu ietekmē arī citas vērtības vai mehānismi. Tipiskākais kontrolētas apmācības paveids ir ***apmācība ar skolotāju***, un šī darba ietveros tie šie jēdzieni tiek lietoti kā sinonīmi.

**Krusteniskā validācija** (*cross-validation*) modeļa novērtēšanas metode, precīzāk – paraugu dalīšanas metode apakškopās tā, lai viena no apakškopām tiktu

izmantota neironu tīkla apmācībā, bet atlikusī daļa – tā derīguma novērtēšanā (validācijā).

**Krustošana** (*crossover*) ir ***ģenētiskais operators***, kurā iesaistīts vairāk nekā viens sākotnējais ***indivīds***, kuru fragmenti veido pēcnācēju.

**Lokālais gradients** (*local gradient*) ir jēdziens vairākslāņu perceptrona apmācības algoritmā ar kļūdu atgriezeniskās izplatīšanās metodi (*backpropagation*). Tas ir lielums, kas reprezentē vēlamo svara apmācības tendenci, balstoties uz neironu tīkla kļūdu.

**Mīkstinātā derīguma funkcija** (*softened fitness function*) ***reducētā derīguma funkcija***, kur izmestā kritērija vērtējuma vietā ir stājies neironu tīkla vērtējums. Kopā ar ***nominālo derīguma funkciju*** un ***reducēto derīguma funkciju*** ietilpst metodikā, kas izstrādāta, lai novērtētu autora piedāvāto ģenētisko algoritmu un neironu tīklu hibrīdo modeli.

**Mutācija** (*mutation*) ir vienkāršākais ***ģenētiskais operators***, kas nosaka viena vai vairāku bitu (elementu) izmaiņu ***hromosomā***.

**Neirons** (*neuron*). 1. Neironu tīkla pamatvienība. Vienkāršs skaitļošanas elements (*processing unit*), no kādiem sastāv neironu tīkls. Neironam var būt vairākas ieejas, bet tikai viena izeja. Neironi neironu tīklā parasti tiek grupēti slāņos. 2. Nervu šūna.

**Neironu tīkli** (*neural networks*), precīzāk, mākslīgie neironu tīkli (*artificial neural networks*) ir skaitļošanas paradigma, kas savā attīstībā iedvesmojusies no cilvēka nervu sistēmas un smadzeņu struktūras un darbības izpētē gūtajām idejām. Mūsdienu neironu tīklus raksturo divas galvenās īpašības: (1) apmācība kā informācijas ieguves veids, (2) konekcionisms, kas nosaka, ka neironu tīkls sastāv no liela daudzuma elementu.

**Neironu tīkls** (*neural network*) ir mašīna, kas ir veidota, lai modelētu smadzeņu darbību noteikta uzdevuma sasniegšanai vai problēmas risināšanai.

**Nobīde** (jeb papildus svars) (*bias*) ir papildus vērtība ***perceptrona neironā***, kas līdzīgi svaram piedalās ***summēšanas funkcijā*** un iegūst vērtību ***apmācības*** ceļā. Nepieciešama situācijās, kas pie nulles ieejām, neironam jāizdod nenulles vērtība.

**Nominālā derīguma funkcija** (*nominal fitness function*) ir ***derīguma funkcija***, kas ietilpst autora izveidotajā stundu saraksta sastādīšanas modelī un ietver visus 4 tieši realizētos vērtēšanas kritērijus, bet neietver neironu tīkla doto vērtējumu. Kopā ar ***reducēto derīguma funkciju*** un ***mīkstināto derīguma funkciju*** ietilpst metodikā, kas izstrādāta, lai novērtētu autora piedāvāto ģenētisko algoritmu un neironu tīklu hibrīdo modeli.

**Novērtējuma kļūda** (*estimation error*) ir neironu tīkla dotā kļūda uz funkciju, kuru tam vajadzētu modelēt. Šāda funkcija vispārīgā gadījumā ir nezināma, tādēļ novērtējuma kļūdu nosaka kā kļūdu uz paraugiem, uz kuriem neironu tīkls nav ticis apmācīts. Sk. ***dispersija***, ***vispārināšana***, ***aproksimācijas kļūda***.

**Pārmērīga pielāgošanās** (*overfitting*) ir parādība, kad neironu tīkla apmācības rezultātā tas tik ļoti pielāgojas apmācības piemēriem, ka vispārināšanu veic slikti.

**Pašorganizējošā karte** (*self-organizing map, SOM*) – sk. Kohonena tīkls.

**Pašorganizējošie neironu tīkli** (*self-organizing networks*) ir neironu tīklu modeļu kategorija, kas izmanto ***apmācību bez skolotāja***.

**Populācija** (*population*) ir optimizācijas problēmas potenciālo risinājumu (***indivīdu***) reprezentāciju kopums ***ģenētiskā algoritma*** ietvaros.

**Radiālo bāzes funkciju tīkls** jeb **RBF tīkls** (*radial basis function network, RBF network*) ir neironu tīklu modelis, kurš sastāv no viena specifiska (RBF) slāņa un viena perceptrona slāņa. RBF slānis izmanto Gausa funkciju par aktivizācijas funkciju, kuras dēļ slānis un tīkls kopumā ieguvis šo nosaukumu. RBF slāņa apmācība ir ļoti īpatnēja, iespējama vairākos variantos, kas nav definēts paša modeļa ietvaros

**Reducētā derīguma funkcija** (*reduced fitness function*) ir ***nominālā derīguma funkcija***, no kuras izmests vērtējums pēc kāda no kritērijiem. Kopā ar ***nominālo derīguma funkciju*** un ***mīkstināto derīguma funkciju*** ietilpst metodikā, kas izstrādāta, lai novērtētu autora piedāvāto ģenētisko algoritmu un neironu tīklu hibrīdo modeli.

**Slānis** (*layer*) ir neironu tīkla struktūras vienība – ***neironu*** grupa. Slāņi daudzos neironu tīklu modeļos izvietoti virknē viens aiz otra. Slānis var būt ieejas, slēptais vai izejas.

**Slēptais slānis** (*hidden layer*) ir ***neironu slānis***, kas ne no vienas puses (***ieeja***, ***izeja***) nav saistīts ar apkārtējo vidi. Dažu neironu tīklu modeļos (piemēram, perceptronam) slēpto slāņu esamībai ir izšķiroša nozīme spējā risināt noteiktas sarežģītības problēmas.

**Sodīšanas funkcija** (*penalty function*) ir objektus novērtējošu funkciju veids, kad augstākam novērtējumam atbilst zemāka funkcijas vērtība un attiecīgi zemākām vērtējumam – lielāka vērtība.

**Summēšanas funkcija** (jeb izplatīšanās funkcija) (*propagation function*) ir neirona komponente – funkcija, kura, kombinējot ieejas signālus un svarus, izrēķina vienu vērtību, kas tālāk piedalās neirona izejas vērtības izskaitļošanā. Summēšanas funkcijas izrēķināto vērtību parasti apzīmē ar burtu virkni

NET. Vistipiskākā summēšanas funkcija ir ieejas signālu un attiecīgo svaru reizinājumu summa.

**Svari** (*weights*). Skaitliskas vērtības, kas piedalās neironu tīkla darbināšanā, kombinējoties ar ieejas signāliem. Svari veido katra neirona un arī visa neironu tīkla galveno atmiņas daļu, kas nodrošina to, ka neironu tīkls funkcionē tā, kā tas funkcionē un kuru vērtības tiek uzstādītas apmācības procesa ceļā.

**Tendenciozitāte** (*bias*) ir neironu tīkla nespēja veikt mērķa funkcijas aproksimāciju, kuras pamatā ir neironu tīkla nepietiekamā sarežģītība (piemēram, pārāk maz slēpto neironu). Tendenciozitāte būtībā norāda uz neironu tīkla spēju (respektīvi, nespēju) pielāgoties apmācības datiem, nevis uz ***vispārināšanas*** līmeni. Tendenciozitāti var uztvert kā ***aproksimācijas kļūdu*** (*approximation error*).

**Tendenciozitātes-dispersijas dilemma** (*bias-variance dilemma*) ir parādība apmācāmos modeļos, kad jāatrod kompromiss starp divām negatīvām īpašībām – ***tendenciozitāti*** un ***dispersiju***, un tādējādi attiecīgi starp ***aproksimācijas kļūdu*** un ***novērtējuma kļūdu***.

**Testa kopa** (*testing*) ir neironu tīklam padodamo paraugu kopa, kas tiek izmantota neironu tīkla darbības pareizības un kvalitātes pārbaudē. Bieži – tas pats, kas ***validācijas kopa***.

**Troksnis** (*noise*) ir nelielas izmaiņas ieejas datos. Trokšņa pamatā ir neprecīzi vai kļūdaini mērījumi, un tas var novest pie sistēmas nekorektas darbības. Neironu tīkli ir samērā robusti pret trokšņiem, un to bieži sauc par vispārināšanas spēju

**Trokšņu injekcija** (*noise injection*) ir trokšņa pievienošana ieejas datiem vai starprezultātiem neironu tīkla apmācības laikā.

**Vairākslāņu perceptrons** (*multilayer percpetron*) ir viens no populārākajiem neironu tīklu modeļiem, kas izmanto ***apmācības*** metodi ***ar skolotāju***. Tiek lietots klasifikācijas problēmu risināšanā.

**Validācijas kopa** (*validation set*) ir neironu tīklam padodamo paraugu kopa, kas tiek izmantota neironu tīkla darbības pareizības un kvalitātes pārbaudē pēc tam, kad neironu tīkls ir apmācīts. Daudzos gadījumos tiek saukta arī par ***testa kopu***.

**Vispārināšana** (*generalization*) ir neironu tīkla spēja saprātīgi reaģēt uz tādiem ***ieejas paraugiem***, kuri ir līdzīgi, bet ne identiski tiem, ar kuriem neironu tīkls ir ticis apmācīts.

## *Darbā lietoto saīsinājumu skaidrojumi*

**BAM** (=*bidirectional associative memory*) – divvirzienu asociatīvā atmiņa

**CSP** (=*constraint satisfaction problem*) – ierobežojumu izpildīšanas problēma

**GA** (=*genetic algorithm*s) – ģenētiskie algoritmi

**ĢA** – ģenētiskie algoritmi

**MLP** (=*multi-layer perceptron*) – vairākslāņu perceptrons

**NET** – sk. summēšanas funkcija

**RBF** (=*radial basis functio*n) – radiālā bāzes funkcija

**SOM** (=*self organizing map*) – pašorganizējošā karte

**SSP** – stundu saraksta sastādīšanas problēma

# Atsauces

## *Citu autoru darbi*

[alifantis01] Alifantis T., Robinson S. Using Simulation and Neural Networks to Develop a Scheduling Advisor. Proceedings of the 2001 Winter Simulation Conference, 2001, pp. 954–958

[arous99] Arous N., Ellouze N., Abdallah S. Evolutionary Potential Timetables Optimization by Means of Genetic and Greedy Algorithms. Proceedings of the 1999 International Conference on Information Intelligence and Systems (ICIIS), 1999

[azzini06] Azzini A., Tettamanzi A. A Neural Evolutionary Approach to Financial Modeling. GECCO'06, July 8-12, Seattle, Washington, USA, pp. 2716-2723

[bishop95] Bishop C. M. Training with Noise is Equivalent to Tikhonov Regularization. Neural Computation 7 No. 1, 1995, pp. 108-116

[colorni98] Colorni A., Dorigo M. Maniezzo V. Metaheuristics for High School Timetabling. Computational Optimization and Application, vol. 9, 1998, pp. 275-298

[fausett94] Fausett L. Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Prentice-Hall, Inc, 1994

[fernandes99] Fernandes C., Caldeira J. P., Melicio F., Rosa A. High School Weekly Timetabling by Evolutionary Algorithms. Proceedings of the 1999 ACM symposium on Applied computing, 1999, pp. 344-350

[frolova99] Frolova L. Ekonomisko procesu matemātiskā modelēšana: Teorija un prakse. Rīgā: Biznesa augstskola Turība, 1999

[hammadi98] Hammadi N. C., Ito H. Improving the Performance of Feedforward Neural Networks by Noise Injection into Hidden Neurons. Journal of Intelligent and Robotic Systems, vol. 21, 1998, pp. 103-115

[haykin99] Haykin S. Neural Networks: a Comprehensive Foundation. 2nd ed. Prentice-Hall, Inc, 1999

[hebb49] Hebb D. The Organization of Behaviour. Wiley Publication, New York City, 1949

[karystinos00] Karystinos G. A., Pados D. A. On Overfitting, Generalization, and Randomly Expanded Training Sets. IEEE Transactions on Neural Networks, vol. 11, no. 5, september 2000

[kecman02] Kecman V. Learning and Soft Computing; Support Vector Machines, Neural Networks, and Fuzzy Logic Models. Pearson Education, 2001

[kim05] Kim J.-H., Choi S.-S., Moon B.-R. Normalization for Neural Network in Genetic Search. GECCO'05, June 25-29, 2005, Washington, D.C., USA, pp. 1581-1582

[kohonen84] Kohonen T. Self-Organization and Associative Memory. Springer Series of Information Science, Springer-Verlag, 1984

[luger02] Luger G. F. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Fourth Edition. Addison-Wesley, 2002

[mcculloch] McCulloch W.S., Pitts W. A Logical Calculus of the Ideas Immenent in Nervous Activity. Bulletin of Mathematical Biophysics 5, 1943

[minsky69] Minsky M., Papert S. Perceptrons. MIT Press, Cambridge, MA., 1969

[osowski02] Осовский С. Нейронные сети для обработки информации: пер. с польского. М.: Финансы и статистика, 2002, 344 с.

[pardoe05] Pardoe D., Ryoo M., Mikkulainen R. Evolving Neural Network Ensembles for Control Problems. GECCO'05, June 25-29, 2005, Washington, D.C., USA, pp. 1379-1384

[rosenblatt62] Rosenblatt F. A Comparison of Several Perceptron Models. Spartan Books, Washington D.C., 1962

[rummelhart84] Rummelhart D.E., McClelland J.L. Parallel Distributed Processing: Explorations of the Microstructure of Cognition, vol. 1. Foundations, The MIT Press, 1986

[russell02] Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 2nd ed. Prentice-Hall, Inc., 2002

[santos05] Santos H. G., Ochi L. S., Souza M. A Tabu Search Heuristic with Efficient Diversification Strategies for the Class/Teacher Timetabling Problem. ACM Journal of Experimental Algorithmics, vol. 10, article No. 2.9, 2005, pp. 1-16

[scherer97] Scherer A. Neuronale Netze: Grundlagen und Anwendungen. Vieweg, 1997

[seghouane04] Seghouane A.-K., Moudden Y., Fleury G. Regularizing the Effect of Input Noise Injection in Feedforward Neural Network Training. Neural Computation & Application, vol. 13, 2004, pp. 248-254

[singh00] Singh S. Noisy Time-Series Prediction Using Pattern Recognition Techniques. Computational Intelligence, vol. 16, no. 4, 2000

[tam03] Tam V., Ting D. Combining the Min-Conflicts and Look-Forward Heuristics to Effectively Solve A Set of Hard University Timetabling Problems. Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03), 2003

[widrow] Widrow B., Hoff M.E. Adaptive Switching Circuits. IRE WESCON Convention Record, Institute of Radio Engineers, New York, 1960, pp. 126-134

[wilson94] Wilson E. Backpropagation Learning for Systems with Discrete-Valued Functions. Proceedings of the World Congress on Neural Networks. San Diego, California, 1994

## *Autora publikācijas*

[zuters05a] Zuters J. An Adaptable Computational Model for Scheduling Training Sessions. Annual Proceedings of Vidzeme University College "ICTE in Regional Development" (presented at the 6th Conference on the Baltic Studies in Europe, June 17-19, 2005, Valmiera, Latvia), 2005, pp. 110-113

[zuters05b] Zuters J. An Extension of Multi-Layer Perceptron Based on Layer-Topology. Proceedings of the 5th International Enformatika Conference'05. Prague, Czech Republic, August 26-28, 2005, pp. 178-181

[zuters06a] Zuters J. A Supervised Learning Process Using Randomly Generated Training Patterns to Obtain an Evaluative Neural Network. WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 4, Volume 3, April 2006, ISSN 1790-0832, pp. 825-831

[zuters06b] Zuters J. An Ensemble of Neural Networks as Part of a GA-based Model to Solve the School Timetabling Problem. Local proceedings of the 7th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2006), Vilnius, Lithuania, July 3-6, 2006, pp. 175-182

[zuters07] Zuters J. Neural Networks to Enrich Fitness Function in a GA-based School Timetabling Model. WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 2, Volume 4, February 2007, ISSN 1790-0832, pp. 346-353