VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
STUDIJŲ PROGRAMA: INFORMATIKA

Pamaininio darbo tvarkaraščio sudarymas naudojant giliuosius neuroninius tinklus
Shift sheduling using deep neural networks

Baigiamasis magistro darbas

Atliko: Lukas Žvilauskas
VU el. p.: lukas.zvilauskas@mif.stud.vu.lt

Vadovas: asist. dr. Linas Petkevičius
Recenzentas: vyres. m. d. dr. Virginijus Marcinkevičius

Vilnius
2022

Pamaininio darbo tvarkaraščio sudarymas naudojant giliuosius neuroninius tinklus

Santrauka

Darbe nagrinėjamas pamaininio darbo tvarkaraščių sudarymas naudojant generatyviniais besivaržančiais tinklais grįstą modelį kaip alternatyvą optimizavimo algoritmams. Šiame darbe yra pateikta pamaininio darbo tvarkaraščio sudarymo uždavinio formuluotė, pasiūlytas generatyvinis modelis tvarkaraščių sudarymui, šis modelis realizuotas praktiškai ir su modeliu atlikti eksperimentai nustačius įvairius tvarkaraščio apribojimus. Taip pat atlikta realizuoto generatyvinio modelio ir pasirinkto optimizavimo įrankio lyginamoji analizė.

Raktiniai žodžiai : Pamaininio darbo tvarkaraščiai, generatyviniai besivaržantys tinklai, gilieji neuroniniai tinklai, optimizavimo algoritmai

Shift sheduling using deep neural networks

Abstract

This thesis will analyze shift scheduling using a model based on generative adversarial networks as an alternative to optimisation algorithms. In this thesis a shift scheduling problem is formulated, a generative model based on generative adversarial networks for shift scheduling problems is proposed, this model was practically implemented and experiments were performed with it by adding various schedule constraints. A comparative analysis of the proposed generative model and selected optimisation tool was also performed.

Keywords : Shift scheduling, generative adversarial networks, deep neural networks, optimization algorithms

Turinys

Įvadas

Sprendžiama problema yra pamaininio darbo tvarkaraščių sudarymas. Pamaininio darbo tvarkaraščiai gali turėti dideli kiekį taisyklių ir apribojimų, pvz.: darbuotojai negali dirbti kelias pamainas iš eilės, turi išdirbti tiek pat valandų per savaitę. Tvarkaraščių sudarymo problema yra matematinė optimizavimo problema ir jai spręsti yra naudojami optimizavimo algoritmai [1, 2]. Dirbtiniai neuroniniai tinklai gali būti panaudoti kaip alternatyva optimizavimo algoritmams. Apmokytas dirbtinis neuroninis tinklas galėtų iš karto pateikti tvarkaraščio išdėstymą neatliekant sudėtingo optimizavimo procedūros bei pateikti skirtingų apribojimus tenkinančių sprendimų alternatyvų. Dirbtiniai neuroniniai tinklai buvo sėkmingai pritaikyti optimizavimo uždaviniams spręsti jau 1985 metais [3], kai naudojant *Hopfield* tinklą buvo pasiūlyta spręsti keliaujančio pirklio uždavinį. Tvarkaraščių sudarymo uždaviniui spręsti planuojama panaudoti generatyviniais besivaržančiais tinklais grįstą modelį, kuris galėtų sugeneruoti taisykles atitinkančius tvarkaraščius.

Generatyviniai besivaržantys tinklai (*GAN*) yra dirbtinių neuroninių tinklų tipas sukurtas Ian Goodfellow 2014 metais [4] ir skirtas naujų duomenų generavimui. *GAN* sudaro du besivaržantys neuroniniai tinklai: generatorius ir diskriminatorius. Generatorius generuoja naujus duomenis iš triukšmo vektoriaus, o diskriminatorius klasifikuoja generatoriaus sugeneruotus duomenis bei duomenis iš duomenų rinkinio į tikrus ir netikrus. Taip yra apmokomas *GAN* neuroninis tinklas, kol diskriminatorius nebegali atskirti realių ir sugeneruotų duomenų. Šie neuroniniai tinklai yra plačiausiai naudojami paveikslėlių generavimui, pvz.: žmonių veidų generavimui [5], tikroviškų fotografijų generavimui [6], paveikslėlių generavimui iš teksto [7]. *GAN* taip pat galima panaudoti ir kitokio formato duomenims generuoti. 2017 metais buvo pasiūlytas *SeqGAN*, nuoseklių duomenų, kaip teksto ar muzikos generavimui [8]. Tačiau sprendimų *GAN* ar kitokio tipo neuroninius tinklus panaudoti pamaininio darbo tvarkaraščių generavimui iki šiol nebuvo pasiūlyta. Pritaikant *GAN* grįstą modelį tvarkaraščių generavimui, generatorius iš triukšmo galėtų sugeneruoti matricą sudarytą iš laiko momentų ir darbuotojų, o diskriminatorius pagal numatytus apribojimus paskaičiuoti klaidą. Neuroninis tinklas taip būtų apmokomas, kol bus pasiekiami tinkami naudoti rezultatai.

Darbo tikslas

Sukurti generatyviniais besivaržančiais tinklais grįstą modelį gebantį generuoti pamaininio darbo tvarkaraščius esant dideliam skaičiui apribojimų.

Uždaviniai

- Atlikti mokslinės literatūros analizę, identifikuoti matematinius pamaininių darbo tvarkaraščių sudarymo modelius.
- Identifikuoti ir formalizuoti taisykles bei apribojimus tvarkaraščių sudaryme, realizuoti jas kaip papildomas tikslo funkcijas.
- Sukurti *GAN* grįstą generatyvinį neuroninio tinklo modelį pamaininio darbo tvarkaraščių sudarymui.
- Realizuoti pasiūlytą modelį bei atlikti lyginamąją tvarkaraščių sudarymo analizę.
- Pademonstruoti, kaip pasiūlytas modelis veikia realiuose praktiniuose uždaviniuose.

1 Teoriniai pagrindai

1.1 Tvarkaraščių sudarymo uždaviniai

Tvarkaraštis yra nepakeičiamas laiko planavimas įrankis, todėl tvarkaraščių sudarymas yra visada aktuali problema, kuriai spręsti vis dar ieškoma naujų ir geresnių būdų. Tvarkaraščių sudarymo uždaviniai yra aktualūs įvairiose srityse: gamybos, inžinerijos, transporto, logistikoje, vadyboje, ekonomikoje. Tvarkaraščių sudarymas yra sudėtingas matematinis optimizavimo uždavinys. Šio uždavinio tikslas yra surasti optimalų tvarkaraštį, kuris atitinka numatytus reikalavimus ir apribojimus. Tvarkaraščių sudarymo uždavinių yra įvairių. Tvarkaraščių sudarymo uždavinius būtų galima suskirstyti pagal taikymo sritį:

- **Gamybos tvarkaraščiai**. Sudarant šiuos tvarkaraščius yra siekiama paskirstyti resursus ir personalą taip, kad būtų galima pagaminti produkciją sumažinant laiką bei kainą. Vienas iš šio tipo uždavinių plačiai aprašytas literatūroje yra *job-shop problem* (JSP) [9, 10]. Šio uždavinio modelis yra sudarytas iš mašinų ir darbų kurie atliekami mašinomis. Kiekvienas darbas yra sudarytas iš kelių operacijų, kurios turi būti atliktos tam tikra tvarka bei naudojant tam tikrą mašiną, kiekviena iš operacijų turi savo vykdymo laiką. Šio uždavinio tikslas yra minimizuoti bendrą visų darbų įvykdymo laiką. Tiek mašinoms, tiek darbams yra taikomi apribojimai:

  – nauja operacija negali būti pradėta kol nėra baigta prieš tai buvusi to darbo operacija,

  – kiekviena mašina vienu metu gali atlikti tik vieną operaciją,

  – pradėta operacija privalo būti atlikta iki galo.

  Taip pat literatūroje yra nagrinėjimas šio uždavinio variantas *flow shop* [11], kai operacijų skaičius sutampa su mašinų skaičiumi ir kiekvieno darbo atlikimo tvarka yra vienoda.

- **Mokymo įstaigų tvarkaraščiai**. Sudarant mokymų įstaigų pamokų ar paskaitų tvarkaraščius yra ieškomas optimalus sprendimas atsižvelgiant į laiko apribojimus, patalpų apribojimus, mokinių ir mokymo įstaigos darbuotojų interesus. Vienas iš šio tipo uždavinių yra universiteto paskaitų tvarkaraščio sudarymas [12, 13, 14]. Šiuo uždaviniu siekiama turint tam tikrą kiekį studentų, auditorijų, dėstytojų bei laiko periodų paskirstyti paskaitas. Kiekvienas dėstytojas taip pat turi savo interesų i kuriuos irgi

reikia atsižvelgti. Uždavinio tikslas yra sumažinti naudojamų auditorijų skaičių atsižvelgiant į dėstytojų interesus ir taikomus apribojimus. Galimi uždavinio apribojimai:

  - kiekviena paskaita privalo būti dėstoma,
  - dėstytojas gali dėstyti tik vieną paskaitą vienu metu,
  - vienoje auditorijoje gali būti dėstoma viena paskaita,
  - auditorijoje turi būti pakankamai vietų atitinkamai grupei,
  - dėstytojai dėsto tomis dienomis, kuriomis gali,
  - tam tikrą paskaitą gali dėstyti tik dėstytojas turintis, atitinkamą kvalifikaciją.

- **Transporto tvarkaraščiai**. Sudarant transporto tvarkaraščius yra siekiama sumažinti sąnaudas bei laiką atsižvelgiant į keleivių interesus. Vienas iš šio tipo uždavinių yra automobilių paskirstymo uždavinys (vehicle scheduling problem) [15, 16]. Šis uždavinys yra taikomas viešajam transportui ir juo siekiama turint grafiką paskirstyti autobusus. Grafike yra nurodyti maršrutai, stotelių išvykimo bei atvykimo laikai. Šio uždavinio tikslas yra sumažinti naudojamų autobusų kiekį ir jų operacines išlaidas. Uždaviniui taikomi apribojimai:

  - autobusai turi būti suskirstyti taip, kad būtų padengtas visas grafikas,
  - kiekviena maršruto kelionė (nuo stotelės iki stotelės) yra priskirta vienam autobusui,
  - vienam autobusui vienu metu negali būti priskirta daugiau nei viena kelionė.

- **Darbo tvarkaraščiai**. Sudarant darbo tvarkaraščius yra siekiama surasti optimalų sprendimą atitinkantį numatytus apribojimus bei darbuotojų interesus. Vienas iš šio tipo uždavinių dažnai aptinkamas literatūroje yra *nurse scheduling problem* (NCP) [17, 18]. Sprendžiant šį uždavinį yra siekiama paskirstyti slaugytojams pamainas bei atostogas atsižvelgiant į gydymo įstaigos reikalavimus ir pačių slaugytojų interesus. Slaugytojų darbo diena įprastai yra sudaryta iš 3 pamainų: rytinės, dieninės ir naktinės. Šiam uždaviniui yra taikomi dviejų rūšių apribojimai: sunkūs, kurių laikytis yra privaloma, kad tvarkaraštis būtų laikomas teisingu bei lengvi, kurių laikytis neprivaloma, bet daugiau jų atitinkantis tvarkaraštis yra laikomas geresniu. Galimi sunkūs apribojimai:

  - per nustatytą kiekį dienų turi būti dirbama ne mažiau nei nustatytas kiekis pamainų (pvz.: per 5 dienas ne mažiau 2 pamainų),

– kiekvienas slaugytojas gali dirbti tik vieną pamainą per dieną,

– jei slaugytojas dirbo naktinę pamainą jis negali dirbti rytinės pamainos kitą dieną.

Galimi lengvi apribojimai:

– kiekvienas slaugytojas negali turėti daugiau nei 3 laisvų dienų iš eilės,

– kiekvienas slaugytojas negali dirbti daugiau nei 7 dienas iš eilės.

1.2 Tvarkaraščių optimizavimo algoritmai

Tvarkaraščių sudarymo uždaviniams spręsti yra naudojami kombinatorinio optimizavimo algoritmai. Kombinatoriniai optimizavimo algoritmai yra skirstomi į tiksliuosius, euristinius ir metaeuristinius. Tiksliejі algoritmai gali rasti optimalų uždavinio sprendinį, bet didėjant uždavinio apimčiai eksponentiškai didėja ir jo vykdymo laikas. Tiksliųjų algoritmų pavyzdžiai: šakų ir ribų (branch and bound), šakų ir rėžių (branch and cut), kertančiųjų plokštumų (cutting planes). Euristiniai algoritmai yra naudojami kai reikia rasti gerą sprendimą per priimtiną skaičiavimo laiką nors jis ir nebus optimalus. Euristiniai algoritmai gali būti pritaikomi tik tam tikriems konkretiems uždaviniams su tam tikrais apribojimais. Euristinių algoritmų pavyzdžiai: artimiausio kaimyno, lokaliosios paieškos, įtraukimo algoritmas. Metaeuristiniai algoritmai yra aukšto lygio nurodymų rinkiniai, kurie aprašo tam tikrą uždavinių sprendimo idėją. Šie algoritmai per priimtiną laiką gali rasti sprendimą artimą optimaliam. Metaeuristiniai algoritmai yra bendro naudojimo ir gali būti pritaikyti įvairiems skirtingiems uždaviniams spręsti, keli metaeuristiniai algoritmai gali būti apjungti. Metaeuristinių algoritmų pavyzdžiai: genetiniai algoritmai, modeliuojamasis atkaitinimas (simulated annealing), paieška su apribojimais (tabu search), skruzdžių kolonijos optimizavimas (ant colony optimization), dalelių spiečiaus optimizavimas (particle swarm optimization), memetiniai algoritmai. Pagrindiniai algoritmai naudojami tvarkaraščių sudarymo uždaviniams yra šakų ir ribų, šakų ir rėžių, modeliuojamojo atkaitinimo, dalelių spiečiaus optimizavimas, skruzdžių kolonijos optimizavimas ir genetiniai algoritmai.

Šakų ir ribų algoritmas. Šis algoritmas priklausantis tiksliųjų algoritmų grupei yra naudojamas gamybos tvarkaraščio sudarymo uždavinio *job-shop sheduling problem* sprendimui [9]. Algoritmo idėja yra suskaidyti galimų sprendimų aibę į poaibius ir patikrinti ar juose gali būti ieškomas sprendinys. Kiekvienas aibės elementas yra galimas sprendinys. Algoritmo vykdymas prasideda sprendinių išvardijimu - paieškos medžio konstravimu, kurio viršūnės yra sprendinių poaibiai. Konstruojant medį yra nagrinėjami tik galimų sprendi-

nių poaibiai, jei sprendiniai negali duoti optimalaus ar galimo sprendimo jie yra atmetami. Tam yra panaudojami viršutinis ir apatiniai rėžiai. Šie rėžiai padeda eliminuoti didelę dalį sprendinių taip sumažinant algoritmo vykdymo laiką. Svarbus algoritmo komponentas yra medžio tyrinėjimo strategija, kuri lemia kokia viršūnė bus pasirinkta šakojimui. Pagrindinės medžio tyrinėjimo strategijos: *depth-first* (DFS), *breadth-first* (BrFS) ir *best-first* (BFS). DFS strategija yra paremta medžio gyliu, kai yra šakojama viršūnė ir visi jos vaikai kol bus gaunamas sprendinys. BrFS strategija yra paremta medžio pločiu, kai yra šakojamos visos viršūnės kiekviename medžio lygyje prieš pereinant į kitą lygį. BFS strategija yra paremta geriausia viršūnė, kiekviename medžio lygyje yra šakojama viršūnė turinti mažiausia apatinį rėžį. Algoritmo vykdymo žingsniai:

1. Pagal tam tikrą formulę yra apskaičiuojamas medžio viršutinis rėžis arba yra priskiriama begalybė

2. Yra parenkama medžio viršūnė ir ji yra šakojama

3. Medis yra šakojamas pagal pasirinktą strategiją, prieš tai yra apskaičiuojami apatiniai viršūnių rėžiai ir lyginami su viršutiniu medžio rėžiu, viršūnės su didesnių apatiniu rėžiu yra atmetamos

4. Radus geresnį sprendinį už dabartinį yra atnaujinamas medžio viršutinis rėžis ir šakojamos likusios viršūnės su mažesniu apatiniu rėžiu

Modeliuojamojo atkaitinimo algoritmas. Šis algoritmas priklausantis metaeuristininių algoritmų grupei yra naudojamas mokymo įstaigų tvarkaraščio sudarymo uždavinio *universiteto paskaitų tvarkaraščio sudarymas* sprendimui [13]. Modeliuojamo atkaitinimo algoritmo idėja kilo iš metalurgijos proceso - atkaitinimo. Atkaitinimas yra procesas kai metalas yra kaitinimas ir vėsinamas, kad būtų pakeistos jo fizikinės savybės. Modeliuojamame atkaitinime yra naudojamas temperatūros kintamasis šiam procesui simuliuoti. Iš pradžių yra nustatoma didelė temperatūra, o algoritmo vykdymo metu ji lėtai yra mažinama. Kol temperatūra yra didelė algoritmas galu priimti sprendinius kurie yra prastesni nei dabartinis sprendinys su tam tikra tikimybę. Kai temperatūra yra mažinama taip pat mažėja ir tikimybė priimti prastesnius sprendinius. Dėl šio "vėsinimo" proceso modeliuojamojo atkaitinimo algoritmas yra efektyvus globalaus optimumo radimui, nes lokalūs sprendimai yra atmetami algoritmo vykdymo pradžioje. Pagrindiniai šio algoritmo komponentai yra temperatūra, tikslo funkcija ir kaimyninio sprendinio paieška. Sprendžiant uždavinius yra svarbu nustatyti tinkamą pradinę temperatūrą ir jos mažinimo strategiją. Tikslo funkcija

yra naudojama įvertinti sprendinio kokybei, o kaimyninis sprendimo paieška reikalinga rasti sprendinį artimą esamajam. Pagrindiniai algoritmo vykdymo žingsniai:

1. Yra nustatoma pradinė temperatūra ir pradinis sprendinys
2. Vykdomas ciklas kol temperatūra sumažėja iki minimalios nustatytos temperatūros, su kiekviena temperatūros reikšme yra įvykdomas nustatytas kiekis iteracijų
3. Yra parenkamas kaimyninis sprendimas šiek tiek pakeitus dabartinį sprendimą
4. Yra nusprendžiama ar pereiti prie šio kaimyninio sprendimo palyginant jį su dabartiniu sprendiniu
5. Jei yra įvykdytos visos numatytos iteracijos yra sumažinama temperatūra ir toliau vykdomas ciklas

Detalesnis algoritmo vykdymas yra aprašytas algoritmo pseudokode (1 pav.)

Dalelių spiečiaus optimizavimas. Šis algoritmas priklausantis metaeuristininių algoritmų grupei yra naudojamas mokymo įstaigų tvarkaraščio sudarymo uždavinio *universiteto paskaitų tvarkaraščio sudarymas* sprendimui [14]. Dalelių spiečiaus optimizavimo algoritmas yra grįstas gyvūnų grupių sumanumu (swarm intelligence), kai gyvūnai priklausantys tam tikrai grupei kaip paukščiai ar žuvys pasidalina žinoma informacija tarpusavyje. Algoritme spiečius yra sudarytas iš dalelių, kurios yra potencialūs optimizavimo uždavinio sprendiniai. Kiekviena dalelė keičia savo pozicija kelių dimensijų paieškos srityje atsižvelgiant į savo asmeninę geriausią poziciją ir viso spiečiaus geriausią poziciją. Toks dalelių pozicijos keitimas padeda pasiekti optimalią poziciją. Dalelių pozicijos optimalumas yra apskaičiuojamas naudojant tikslo funkciją. Vykdant algoritmą yra sukuriama atsitiktinė dalelių populiacija, kurios dydis priklauso nuo sprendžiamo uždavinio. Kiekviena dalelė turi 3 parametrus: poziciją, greitį ir tikslo funkcijos reikšmę. Algoritmo vykdymo žingsniai:

1. Atsitiktinai yra sugeneruojamos dalelės bei nustatomi pradiniai jų pozicijos ir greičio vektoriai.
2. Apskaičiuojamos visų dalelių tikslo funkcijos reikšmės ir yra išsaugoma geriausia visų dalelių reikšmė
3. Vykdomas ciklas tam tikrą iteracijų skaičių
4. Yra suskaičiuojama kiekvienos dalelės nauja greičio reikšmė

5. Yra suskaičiuojama kiekvienos dalelės nauja pozicijos reikšmė

6. Yra apskaičiuojamos kiekvienos dalelės tikslo funkcijos reikšmė ir atnaujinama geriausia dalelės pozicija ir geriausia visų dalelių pozicija

```
s0 – pradinis sprendinys
s' – geriausias sprendinys
sd – dabartinis sprendinys
sb – naujas sprendinys
kmax – maksimalus iteracijų skaičius
k – iteracijų skaičius
t0 – pradinė temperatūra
tmin – minimali temperatūra
t – dabartinė temperatūra
k = 0;
s' = s0;
sd = s0;
t = t0;
kol (t >= tmin) {
  kol(k <= kmax) {
   k = k + 1;
   sn = kaimyninisSprendinys(sd);
   delta = tiksloFunkcija(sn) – tiksloFunkcija(sd);
   jei(delta <= 0) {
     sd = sn;
     jei(tiksloFunkcija(sn) < tiksloFunkcija(s')) {
        s' = sn;
     }
   }
   jei (delta > 0 ir exp(-delta/t) >= rand[0,1]) {
     sd = sn;
   }
  }
  t = temperaturosMazinimas();
  k = 0;
}
```

1 pav.: Modeliuojamojo atkaitinimo algoritmo pseudokodas

Skruzdžių kolonijos optimizavimas. Šis algoritmas priklausantis metaeuristininių algoritmų grupei yra naudojamas gamybos tvarkaraščio sudarymo uždavinio *flow-shop sheduling problem* [11] ir darbo tvarkaraščio sudarymo uždavinio *nurse scheduling problem* [18] sprendimui. Skruzdžių kolonijos optimizavimo algoritmas yra grįstas skruzdžių kolonijų elgsena ieškant maisto. Skruzdės naudoja naudoja tam tikrus cheminius junginius - feromonus, perduoti informacijai apie trumpiausią kelią iki maisto šaltinio. Kai skruzdės pradeda ieškoti

maisto jos atsitiktinai apieško tam tikro plotą, o radusios maisto jį parneša į lizdą ir savo kelyje palieka feromono pėdsakus. Paliekamo feromono kiekis priklauso nuo maisto kokybės ir kiekio. Kai skruzdės ieško kelio jos renkasi kelią kuriame yra didelė feromono koncentracija, taip kitos skruzdės gali pasiekti jau atrastus maisto šaltinius. Skruzdžių kolonijos algoritmas yra įgyvendinamas sudarant aibę sprendinio komponentų, kurie yra naudojami sprendiniui suformuoti ir aibę feromono reikšmių, kuri vadinama feromonų modeliu. Feromonų modelis - tai parametrizuotas tikimybinis modelis ir vienas iš svarbiausių algoritmų komponentų. Feromonų modelis yra naudojamas tikimybiškai sugeneruoti sprendžiamo optimizavimo uždavinio sprendinius suformuojant juos iš sprendinio komponentų. Šis algoritmas siekia iteraciniu būdu rasti uždavinio sprendinį atliekant du pagrindinius veiksmus: sugeneruoti galimus sprendinius naudojant feromonų modelį ir pakeisti feromonų modelio reikšmes taip, kad būtų pagerinimas generavimas ateityje. Feromonų atnaujinimu siekiama koncentruoti paieška paieškos srities regionuose kuriuose yra geresni sprendiniai. Pagrindiniai algoritmo vykdymo žingsniai:

1. Suformuojamos sprendinių komponentų ir feromonų aibės
2. Vykdomas ciklas tam tikrą iteracijų skaičių
3. Naudojant feromonų modelį yra sukonstruojami galima sprendiniai
4. Feromonų modelio reikšmės yra atnaujinamos atsižvelgiant į sukontruotus sprendinius

Genetinis algoritmas. Šis algoritmas priklausantis metaeuristininių algoritmų grupei yra naudojamas gamybos tvarkaraščio sudarymo uždavinio *job-shop sheduling problem* [10] ir darbo tvarkaraščio sudarymo uždavinio *nurse scheduling problem* [17] sprendimui. Genetinis algoritmas yra paremtas evoliucijos idėjomis. Šis algoritmas atitinka natūralios atrankos procesą kur labiausiai prisitaikę individai yra atrenkami reprodukcijai. Natūralioji atranka prasideda labiausiai prisitaikiusių individų atrinkimų. Jie susilaukia palikuonių, kurie yra dar labiau prisitaikę nei jų tėvai ir turi didesnę tikimybę išgyventi. Šis procesas kartojasi ir jo pabaigoje yra gaunama karta labiausiai prisitaikiusių individų. Pagrindinės algoritmo sąvokos: pradinė populiacija, tikslo funkcija, atranka, rekombinacija ir mutacija. Pradinė populiacija tai atsitiktinai sukurta aibė kurios kiekvienas elementas yra individas arba uždavinio sprendinys. Kiekvieno populiacijos individo prisitaikymas yra įvertinamas naudojant tikslo funkciją. Atranka - tai procesas kurio metu yra atrenkami labiausiai prisitaikę individai - tėvai, kurie bus naudojami palikuoniams sukurti. Rekombinacija yra naudojama

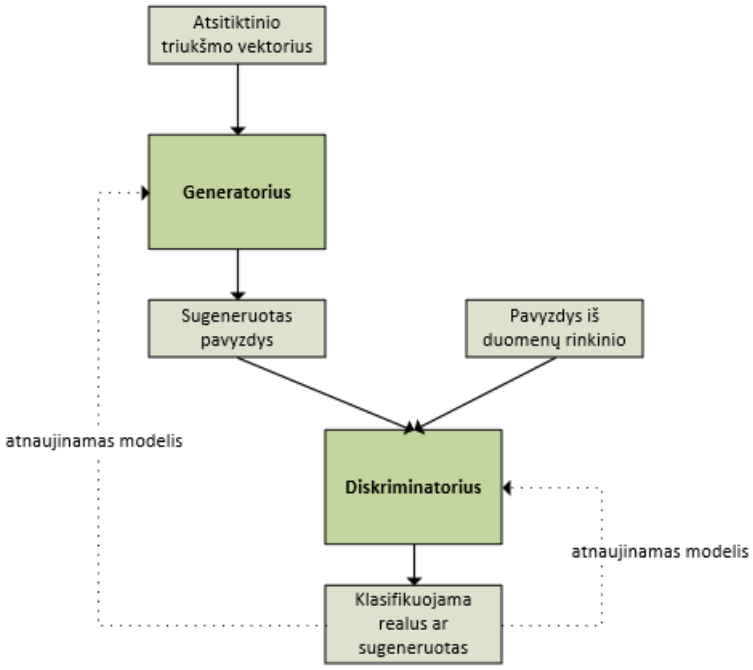
individų palikuonių sukūrimui pagal tam tikrą metodą pvz.: tėvinės chromosomos padalijamos į dvi dalis ir sujungiamos. Mutacija yra procesas kai atsitiktinai pakeičiami vienas ar keli individo genai. Algoritmo vykdymo žingsniai:

1. Atsitiktinai sukuriama pradinė individų (sprendinių) populiacija
2. Pradinė populiacija yra įvertinama pagal tikslo funkciją
3. Vykdomas ciklas nustatytą kiekį iteracijų arba iki tol kol per kelias iteracijas nėra gaunama geresnių sprendinių
4. Pasirenkamas nustatytas kiekis labiausiai prisitaikiusių individų porų iš populiacijos
5. Yra atliekama rekombinacija ir sukuriami tėvų palikuonys
6. Atsitiktinai atliekama palikuonių genų mutaciją
7. Palikuonys yra pridedami prie populiacijos
8. Nustatytas kiekis blogiausiai prisitaikiusių individų yra pašalinamas iš populiacijos

1.3 Generatyviniai besivaržantys tinklai

Generatyviniai besivaržantys tinklai (GAN) yra giliųjų neuroninių tinklų architektūra sukurta 2014 metais. Šie neuroniai tinklai yra skirti naujų duomenų generavimui pagal duomenų rinkinio duomenis. Šių neuroninių tinklų architektūra yra sudaryta iš dviejų neuroninių tinklų: generatoriaus ir diskriminatoriaus. Generatorius iš triukšmo vektoriaus generuoja naujus duomenis, o diskriminatorius klasifikuoja generatoriaus sugeneruotus duomenis ir duomenis iš duomenų rinkinio į tikrus ir sugeneruotus. Šie 2 neuroniniai tinklai vienas su kitu varžosi: generatoriaus tikslas yra sugeneruoti tokius duomenis, kuriuos diskriminatorius palaikytų tikrais, o diskriminatoriaus tikslas yra atskirti realius duomenis nuo sugeneruotų. Mokant GAN, kai generatoriaus sugeneruoti duomenis yra klasifikuojami kaip tikri yra keičiami diskriminatoriaus parametrai, jei ne - yra keičiami generatoriaus parametrai. Baigus GAN mokymą diskriminatorius nebėra naudojamas, o generatorius yra naudojamas naujų duomenų generavimui. GAN architektūra yra pavaizduota 2 paveikslėlyje.

Šie neuroniniai tinklai yra plačiai naudojami įvairių vaizdų generavimo uždavinių sprendimui. Sprendžiant šiuos uždavinius GAN gali pasiekti geresnių rezultatų nei alternatyvūs generatyviniai modeliai kaip variaciniai autoenkoderiai (angl. *variational autoencoders*). Uždavinių pavyzdžiai:



2 pav.: GAN architektūra

- naujų žmonių veidų generavimas [5],
- animacinių veikėjų generavimas [19],
- vaizdų vertimas į kitus vaizdus (angl. *Image-to-Image Translation*) [20],
- vaizdų iš teksto generavimas [7],
- fotografijų iš paveikslėlių generavimas [21],
- fotografijų redagavimas (pvz.: žmogaus bruožų keitimas, sendinimas) [22, 23],
- vaizdų suliejimas į vieną [24],
- vaizdų rezoliucijos padidinimas [25],
- 3D objektų generavimas [26].

GAN taip gali būti pritaikomi ir kitokių duomenų generavimui. Literatūroje yra nagrinėjami sprendimai *GAN* panaudoti nuoseklių ir diskrečių duomenų generavimui. Tokių uždavinių pavydžiai:

- eilėraščių generavimas [8, 27],
- politinių kalbų generavimas [8],

- muzikos generavimas [8, 28],
- kalbos modeliavimas sakinyje (angl. *Sentence-Level Language Modeling*) [27],
- elektroninių sveikatos duomenų generavimas [29].

2 Literatūros analizė

2.1 Optimizavimo algoritmų palyginimas

Visi kombinatorinio optimizavimo algoritmai turi privalumų ir trūkumų. Pagrindinis tiksliųjų algoritmų privalumas yra optimalaus sprendinio radimas, o trūkumas - eksponentiškai didėjantis algoritmo vykdymo laikas didėjant sprendžiamo uždavinio apimčiai. Dėl šių priežasčių tikslieji algoritmai labiau tinka nedidelės apimties uždaviniams spręsti. Euristinių algoritmų pagrindinis privalumas yra gero sprendinio radimas per priimtiną laiką, o trūkumas yra tai, kad šie algoritmai priklauso nuo konkretaus uždavinio. Dėl priimtino vykdymo laiko šie algoritmai labiau tinka praktiniam naudojimui nei tikslieji algoritmai. Pagrindiniai metaeuristinių algoritmų privalumai yra gero sprendinio radimas per priimtiną laiką ir universalumas pritaikant algoritmą skirtingiems uždaviniams, o trūkumai priklauso nuo konkretaus algoritmo. Bendru atveju metaeuristiniai algoritmai yra pranašesni už euristinius algoritmus, tačiau gali būti atvejų, kai sprendžiant sudėtingą uždavinį reikia tam uždaviniui specifinio euristinio algoritmo. Pagrindiniai konkrečių optimizavimo algoritmų privalumai ir trūkumai yra pateikti 1 lentelėje.

Optimizavimo algoritmas	**Privalumai**	**Trūkumai**
Šakų ir rėžių algoritmas	• gali rasti optimalų sprendinį • gali rasti sprendinį greičiau nei šakų ir ribų algoritmas	• vykdymo laikas auga eksponentiškai priklausoma nuo uždavinio sudėtingumo

Šakų ir ribų algoritmas	• gali rasti optimalų sprendinį • atmetamos medžio viršūnės kuriose negali būti optimalaus sprendinio	• vykdymo laikas auga eksponentiškai priklausoma nuo uždavinio sudėtingumo • yra tikrinamos visos medžio viršūnės kuriose gali būti potencialus sprendinys nors optimalus sprendinys jau gali būti rastas
Modeliuojamasis atkaitinimas	• lengvai įgyvendinamas sudėtingiems uždaviniams • gali rasti gerą ar net optimalų sprendinį	• algoritmas negali žinoti ar buvo rastas optimalus sprendinys • vykdymo laikas didėja norint pasiekti geresnį sprendinį
Skruzdžių kolonijos optimizavimas	• gali labai greitai rasti gerą sprendinį • konvergavimas yra garantuotas	• konvergavimo laikas nėra aiškus • tikimybių paskirstymas gali pasikeisti kiekvienoje iteracijoje

Dalelių spiečiaus optimizavimas	• gali būti pritaikomas didelės apimties uždaviniams • gali greitai konverguoti • lengvai įgyvendinamas • neturi daug parametrų	• gali būti sudėtinga nustatyti pradinius parametrus • sprendinio paieška gali įstrigti lokalaus minimumo taške ypač kai sprendžiamas sudėtingas uždavinys
Genetinis algoritmas	• gali būti lengvai pritaikomas skirtingiems uždaviniams • gali sėkmingai rasti globalų optimumą • gali būti paralelizuotas	• negarantuojamas sprendimo optimalumas • vykdymo laikas gali būti ilgas

1 lentelė: Optimizavimo algoritmų privalumai ir trūkumai

2.2 GAN panaudojimas diskrečių duomenų generavimui

GAN panaudojimas diskrečių duomenų generavimui yra aktuali problema nagrinėjama literatūroje. Pagrindinė problema su kuria susiduriama bandant GAN panaudoti diskretiems duomenims generuoti yra generatoriaus parametrų atnaujinimas, kuris atliekamas apskaičiavus tikslo funkcijos reikšmę. Problema yra tame, kad parametrų reikšmių pakoregavimas

gali būti per mažas, kad padarytų įtaką duomenų generavimui, nes diskrečių duomenų aibė yra baigtinė ir toks pakeitimas gali neturėti atitikmens šioje aibėje.

2016 metais buvo pasiūlytas SeqGAN tinklas nuoseklių duomenų generavimui [8]. Šis sprendimas siekia išspręsti dvi problemas kurios aktualios nuoseklių duomenų generavimui naudojant GAN: 1) generatoriaus parametrų atnaujinimas pagal tikslo funkcijos rezultatus gali būti nepakankamas pakeisti generuojamus duomenis 2) GAN gali įvertinti tik visą sugeneruotų nuoseklių duomenų seką, bet negali įvertinti sugeneruotos sekos dalies. Šioms problemoms spręsti buvo panaudotas skatinamasis mokymas (reinforcement learning). Generatorius yra naudojamas kaip skatinamojo mokymo agentas, būsena yra šiuo metu sugeneruotos sekos dalys, o veiksmas yra kita generuojama sekas dalis. Generatorius yra modeliuojamas kaip stochastinė parametrizuota strategija, strategijos gradientui yra naudojama Monte Carlo paieška. Paskata yra apskaičiuojama diskriminatoriaus pagal visą sugeneruotą seką ir yra perduodama generatoriui naudojant Monte Carlo paiešką. Generatoriui yra naudojamas rekurentinis neuroninio tinklo modelis (RNN), o diskriminatoriui konvoliucinis neuroninio tinklo modelis (CNN). SeqGAN buvo sėkmingai pritaikytas nuoseklių duomenų generavimo uždavinių sprendimui: eilėraščių generavimui, politinių kalbų generavimui ir muzikos generavimui.

2017 metais buvo pasiūlytas medGAN tinklas elektroninių sveikatos duomenų (electronic health record) - pacientų įrašų generavimui [29]. Šiuo sprendimu siekiama sugeneruoti didelės apimties diskrečius kintamuosius, tiek binarinius, tiek skaičiuojamus kaip paciento diagnozes ar medikamentų kodus. Šiame sprendime generatoriaus parametrų atnaujinimo problemai spręsti buvo panaudotas *autoencoder* neuroninis tinklas. Autoenkoderis yra naudojamas paversti diskrečius duomenis tolydžiais ir atkurti pradinius duomenis. Autoenkoderis šiuo atveju yra apmokamas naudojant realų duomenų rinkinį prieš apmokant GAN. Tada apmokant GAN generatorius generuoja tolydžius duomenis ir jie yra paverčiami į diskrečius naudojant autoenkoderio dekoderio dalį ir paduodami diskriminatoriui kartu su realiais duomenimis. Taip yra išsprendžiama generatoriaus parametrų atnaujinimo problema, nes generatorius nesiekia generuoti diskrečių duomenų. Šis GAN modelis buvo sėkmingai panaudotas sveikatos duomenų generavimui, kurie prilygsta realiems duomenis.

2.3 Tvarkaraščių apribojimų formavimas

Šakų ir ribų algoritmas. Taikant šį algoritmą *job-shop sheduling problem* uždavinio [9] sprendimui yra įvertinami apribojimai:

- mašina vienu metu gali vykdyti tik vieną operacija,
- operacija negali būti nutraukta,
- darbas gali turėti ne daugiau nei nustatytą kiekį operacijų,
- kiekvieno darbo operacijos turi tam tikrą eiliškumą.

Šakų ir rėžių algoritmas. Taikant šį algoritmą *vehicle sheduling problem* uždavinio [16] sprendimui yra įvertinami apribojimai:

- kiekviena kelionė (trip) gali būti priskirta tik vienai transporto priemonei,
- kiekviena transporto priemonė turi pradėti ir baigti savo darbą toje pačioje stotyje,
- kiekvienai transporto priemonei yra priskirta aibė kelionių, kur gretimos kelionės yra įmanomos (eina viena po kitos bendroje visų kelionių aibėje),
- kiekvienoje stotyje gali būti tam tikras maksimalus kiekis transporto priemonių.

Modeliuojamojo atkaitinimo algoritmas. Naudojant šį algoritmą universiteto paskaitų tvarkaraščio sudarymui [13] tvarkaraščio apribojimai yra įvertinami skaičiuojant tikslo funkcijos reikšmę. Šiame uždavinyje yra įvertinami 6 apribojimai:

- konkreti paskaita negali būti priskirta tam tikru laiku,
- tas pats dėstytojas negali būti priskirtas 2 paskaitoms vienu metu,
- ta pati grupė negali būti priskirta 2 užsiėmimams vienu metu,
- kelios to pačio dalyko paskaitos negali būti padalintos į dvi dienas,
- paskaitos negali būti priskirtos laiku, kuris nepatogus dėstytojui,
- turi būti išvengiama paskaitų priskyrimų metu, kai vyksta kitų paskaitų kartojimas/taisymasis.

Kiekvienas apribojimas yra įvertinimas pagal atitinkamą formulę, kuri apskaičiuoja baudą, o tikslo funkcijos reikšmė yra gaunama sudėjus visų apribojimų formulių reikšmes.

Dalelių spiečiaus optimizavimas. Naudojant šį algoritmą universiteto paskaitų tvarkaraščio sudarymui [14] skaičiuojant tikslo funkciją yra įvertinami sunkūs ir lengvi tvarkaraščio apribojimai. Taikomi sunkūs apribojimai:

- studentui tam tikru laiku gali būti priskirta tik viena paskaita,

- dėstytojas tuo pačiu laiku gali dėstyti tik vieną paskaitą,
- paskaitos negali būti priskirtos pertraukų metu,
- kelios iš eilės vykstančios to pačio dalyko paskaitos negali turėti pertraukos,
- paskaitos gali vykti tik atitinkamose auditorijose.

Taikomi lengvi apribojimai:

- kiek įmanoma turi būti laikomasi dėstytojų interesų dėstyti tam tikru laiku,
- turi būti vengiama tam pačiam dėstytojui priskirti kelias paskaitas iš eilės.

Skruzdžių kolonijos optimizavimas. Šis algoritmas yra naudojamas *nurse-scheduling problem* uždavinio variantui spręsti kai slaugytojus reikia paskirstyti keliems ligoninės departamentams [18]. Skaičiuojant tikslo funkciją yra įvertinami įvairūs apribojimai:

- maksimalus darbų valandų kiekis per dieną ir savaitę,
- maksimalus darbų dienų kiekis per savaitę,
- maksimalus naktinių pamainų kiekis per savaitę ir mėnesį,
- maksimalus naktinių pamainų kiekis iš eilės,
- minimalus poilsio laikas iš eilės per savaitę,
- kiekvienoje pamainoje gali dirbti tik slaugytojas su atitinkamu kvalifikacijos lygiu,
- turi būti atsižvelgiama į slaugytojų interesus dirbti tam tikruose ligoninės departamentuose,
- turi būti atsižvelgiama į slaugytojų interesus dirbti tam tikromis dienomis ir valandomis,
- turi būti atsižvelgiama į slaugytojų interesus dirbti pozicijoje kuri reikalauja tam tikro kvalifikacijos lygio,
- visos pamainos turi būti padengtos.

Genetinis algoritmas. Naudojant šį algoritmą gamybos tvarkaraščio uždaviniui *job-shop scheduling problem* [10] spręsti skaičiuojant tikslo funkciją yra įvertinami šie apribojimai:

- bet kuriuo metu yra pasiekiamas ne daugiau nei nustatytas kiekis kiekvieno tipo resursų (mašinų),
- kiekvieno darbo operacijos turi nustatytą eiliškumą,
- gali būti nustatytas darbų pradžios ir pabaigos laikas,
- tam tikru laiku tam tikri resursai gali būti nepasiekiami,
- kai kurie resursai gali būti riboti.

Sprendžiant *nurse-scheduling problem* uždavinį yra taikomas netiesioginis genetinis algoritmas [17]. Tokiu atveju genetinis algoritmas yra naudojamas išspręsti problemą be apribojimų - surikiuoti slaugytojus, o apribojimai yra įvertinami naudojant dekoderį, kuris paverčia genetinio algoritmo rezultatą į tinkamą tvarkaraštį. Uždavinyje yra įvertinami apribojimai:

- kiekvienas slaugytojas pagal sutartį dirba tik naktines, tik dienines arba dienines ir naktines pamainas,
- kiekvienas slaugytojas dirba nustatytą kiekį pamainų per savaitę,
- pamainoje turi dirbti nustatytas kiekis slaugytojų su tam tikru kvalifikacijos lygiu, aukštesnės kvalifikacijos slaugytojai gali pakeisti žemesnės kvalifikacijos slaugytojus,
- turi būti padengtos visos pamainos.

2.4 Tvarkaraščių sudarymo įrankiai

OR-Tools. Tai yra atviro kodo įrankis sukurtas "Google" kompanijos ir skirtas optimizavimo uždaviniams spręsti [30]. Šis įrankis yra pritaikytas įvairių tipų optimizavimo uždavinių sprendimui: priskyrimo uždaviniams, paskirstymo uždaviniams, dėžių pakavimo uždaviniams, tvarkaraščių sudarymo uždaviniams ir daugeliui kitų. Jis yra parašytas *C++* programavimo kalba, bet yra palaikomos ir *C#*, *Python* bei *Java* kalbos. Šis įrankis turi 2 pagrindinius komponentus: *MPSolver*, skirtą tiesinio programavimo uždavinių sprendimui ir *CP-SAT solver*, skirtą sveikųjų skaičių programavimui (angl. integer programming).

Choco Solver. Tai atviro kodo *Java* kalbos biblioteka, kuri yra skirta apribojimų programavimui (angl. constraint programming) [31]. Šis įrankis gali būti pritaikytas įvairiems optimizavimo uždaviniams: keliaujančio pirklio uždaviniui, matematiniam šachmatų uždaviniui, lėktuvo nusileidimo uždaviniui ir kitiems uždaviniams. Šio įrankio pagrindinis

komponentas yra modelis, kuris nusako sprendžiamą problemą ir yra sudaromas iš kintamųjų bei apribojimų. Uždavinio sprendimui yra naudojama paieška ir apribojimų propagavimas.

OptaPlanner. Tai yra atviro kodo optimizavimo įrankis parašytas Java kalba [32]. Šis įrankis yra skirtas įvairiems optimizavimo uždaviniams: transporto priemonių paskirstymo uždaviniams, darbuotojų tvarkaraščių sudarymo uždaviniams, konferencijų planavimo uždaviniams, užduočių paskirstymo uždaviniams, nuoseklios gamybos uždaviniams ir daugeliui kitų. *OptaPlanner* pritaiko objektinio programavimo principus optimizavimo uždavinių sprendimui. *OptaPlanner* nuo kitų įrankių skiriasi tuo, kad suskaičiuoja balą (angl. score) rastam sprendiniui ir taip gali rasti gerą, bet ne optimalų sprendinį. Šis įrankis taip pat palaiko kietus ir minkštus apribojimus.

3 Tvarkaraščių sudarymo uždavinys

3.1 Matematinis uždavinio formulavimas

Užduotis yra sugeneruoti pamaininio darbo tvarkaraštį pagal formulę $\hat{S} = f_\theta(z)$, kur f_θ yra neuroninis tinklas su nežinomu kiekiu parametrų $\theta \in R^{d_0}$ ir $z \in R^{d_z}$ yra triukšmo vektorius. Grafikas $S_{T\times P}$ yra matrica, kur T - laiko momentai (eilutės) ir P - darbuotojai (stulpeliai). Kiekvienas prognozuojamos matricos elementas yra tikimybė darbuotojui dirbti tuo laiko momentu, kuri klasifikuojama, į dirbs/nedirbs.

Sugeneravus prognozę (matricą) yra skaičiuojama klaida $\mathcal{L}$, įvertinus visus įtrauktus k apribojimus $\mathcal{L}_{cr,k}$:

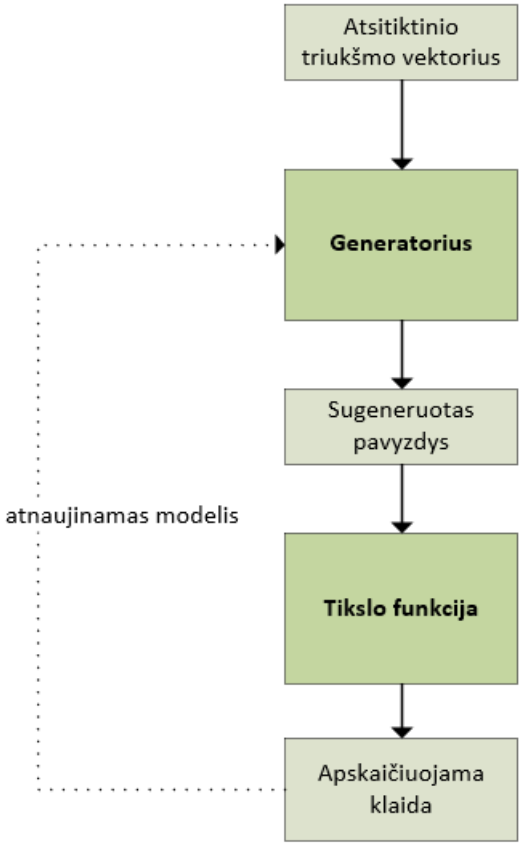
$$\mathcal{L} = \alpha_1\mathcal{L}_{cr,1} + .. + \alpha_k\mathcal{L}_{cr,k}$$

čia α_k yra apribojimo įtaka galutinėje tikslo funkcijoje.

3.2 Tvarkaraščių generavimas naudojant generatyvinį modelį

Tvarkaraščių sudarymo uždaviniui spręsti yra siūlomas kitoks nei įprastas *GAN* modelis (3 pav.). Šiuo atveju nebus naudojamas duomenų rinkinys, nes planuojama generuoti tvarkaraščius pagal įvairius apribojimus, o tokių duomenų (juo labiau optimalių tvarkaraščių) rinkinių neturime. Vietoje atskiro neuroninio tinklo - diskriminatoriaus, kuris klasifikuotų duomenis bus naudojama tikslo funkcija, kuri įvertins kaip sugeneruotas tvarkaraštis atitinka nustatytus apribojimus. Tvarkaraščių generavimui bus naudojamas tik atsitiktinai sugeneruotas triukšmas. Apmokytas neuroninis tinklas, jam padavus triukšmo vektorių, su-

generuos tikimybių matricą iš darbuotojų ir laiko momentų, kurios elementus suapvalinus iki sveiko skaičius bus gaunamas reikiamas tvarkaraštis.



3 pav.: *GAN* modelis tvarkaraščių generavimui

4 Praktinė dalis

4.1 Siūlomas generatyvinis modelis

Siūlomas GAN grįstas modelis yra sudarytas iš vieno neuroninio tinklo (generatoriaus) ir tikslo funkcijos (diskriminatoriaus atitikmens). Šis neuroninio tinklo modelis buvo realizuotas naudojant `Python` programavimo kalbos biblioteką `PyTorch`. Neuroninio tinklo architektūra yra sudaryta iš 3 neuronų sluoksnių, pirmų 2 dimensija yra 32 × 32, paskutinio 32 × (darbuotojų kiekis × valandų kiekis). Po pirmo ir antro sluoksnio yra naudojama $ReLU$ funkcija. Aktyvacijai yra naudojama sigmoidinė funkcija, o išeities dimensija - duomenų rinkinio dydis × (darbuotojų kiekis × valandų kiekis). Šis modelis yra apmokamas vykdant tam tikrą kiekį iteracijų, kuriose yra atsitiktinai sugeneruojamas triukšmo vektorius, paduodamas neuroniniam tinklui, gautam rezultatui paskaičiuojama tikslo funkcija ir tada

atliekamas propagavimas. Atsitiktinis triukšmo vektorius yra generuojamas taikant atsitiktinių skaičių generavimą iš tolygaus intervalo, jo dimensija - duomenų rinkinio dydis × neuronų kiekis (šiuo atveju 32). Priklausomai nuo darbuotojų kiekio, nustačius 24 valandas, modelis turi nuo 3696 nežinomų parametrų, kai nustatyti 2 darbuotojai iki 27456 nežinomų parametrų, kai nustatyti 32 darbuotojai (2 lentelė). Modelio architektūra nustačius duomenų rinkinio dydį 512, 24 valandas ir 4 darbuotojus yra pavaizduota 4 paveikslėlyje

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param
==========================================================================
Generator                                [512, 24, 4]              --
├─Linear: 1-1                            [512, 32]                 1,056
├─ReLU: 1-2                              [512, 32]                 --
├─Linear: 1-3                            [512, 32]                 1,056
├─ReLU: 1-4                              [512, 32]                 --
├─Linear: 1-5                            [512, 96]                 3,168
├─Sigmoid: 1-6                           [512, 24, 4]              --
==========================================================================
```

4 pav.: Siūlomo generatyvinio modelio architektūra

Darbuotojų kiekis	Nežinomi modelio parametrai
2	3696
4	5280
8	8448
16	14784
32	27456

2 lentelė: Modelio nežinomų parametrų kiekis

4.2 Pradiniai eksperimentai

Naudojant sudarytą neuroninio tinklo modelį buvo atliekami eksperimentai norint realizuoti konkretų tvarkaraščio sudarymo uždavinį įvertinant apribojimus dėl darbuotojų dirbamų valandų, pamainų paskirstymo ir kitus. Prieš pradedant atlikti eksperimentus buvo nustatyti šie techniniai modelio parametrai:

- optimizatorius *Adam*,
- mokymo žingsnis 0.001 (angl. learning rate),

- 1000 iteracijų,
- 512 duomenų rinkinio dydis (angl. batch size).

Visuose atliktuose eksperimentuose buvo nustatyta 1 diena (24 valandos). Vieną valandą tvarkaraštyje atitinka viena eilutė, bet eilutę būtų galima laikyti ir 30 ar 15 minučių, jei tam būtų poreikis.

4.2.1 Pirmas eksperimentas

Šiam eksperimentui buvo nustatyti 3 darbuotojai. Buvo nustatyti 2 apribojimai iš kurių sudaryta tikslo funkcija:

- Vienu metu (tą pačią valandą) gali dirbti vienas darbuotojas.
- Visi darbuotojai dirba po lygiai valandų.

Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} + \alpha_2 \mathcal{L}_{cr,2} \tag{1}$$

$$\mathcal{L}_{cr,1} = \sum_{t=1}^{T} \left(\left(\sum_{p=1}^{P} \hat{S}_{t,p} \right) - 1 \right)^2 \tag{2}$$

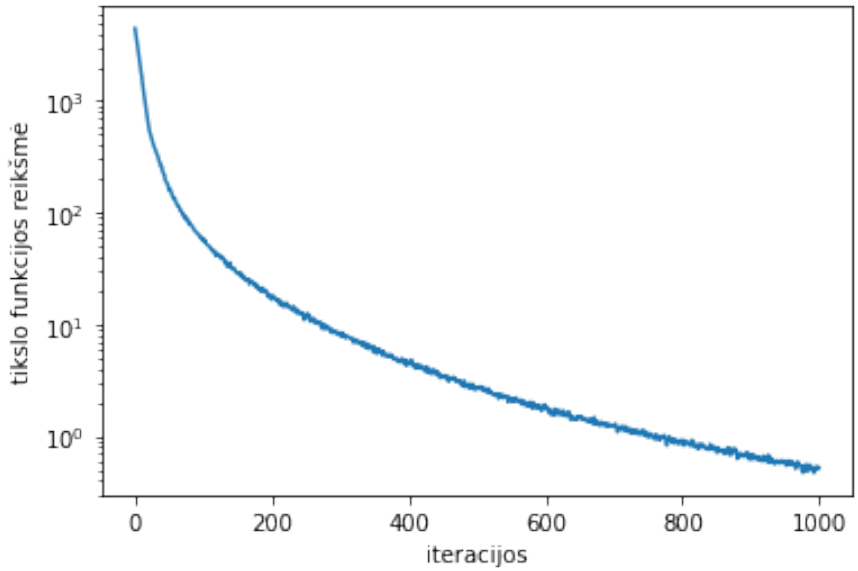
$$\mathcal{L}_{cr,2} = \sum_{p=1}^{P} \sum_{p'=1, p \neq p'}^{P} \left(\left(\sum_{t=1}^{T} \hat{S}_{t,p} \right) - \sum_{t=1}^{T} \hat{S}_{t,p'} \right)^2 \tag{3}$$

Čia $\alpha_k = 1, k = 1, 2, T =$ valandų skaičius, $P =$ darbuotojų skaičius, $\hat{S} =$ matrica (tvarkaraštis).

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (5 pav.). Apmokius neuroninį tinklą buvo sugeneruota 10240 tvarkaraščių ir kiekvienam iš šių tvarkaraščių buvo apskaičiuota kiekvieno apribojimo reikšmė. Visi iš šių tvarkaraščių atitiko abu apribojimus, tai yra $\mathcal{L}_{cr,1} = 0$ ir $\mathcal{L}_{cr,2} = 0$ (3 lentelė).

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240
$\mathcal{L}_{cr,2}$	10240 iš 10240

3 lentelė: Pirmo eksperimento rezultatai



5 pav.: Pirmo eksperimento modelio mokymo rezultatai

4.2.2 Antras eksperimentas

Šiam eksperimentui buvo nustatyti 4 darbuotojai. Eksperimento tikslas padaryti, kad skirtingai nei pirmame eksperimente keli darbuotojai galėtų dirbti vienu metu ir darbuotojai būtų suskirstyti į pamainas. Pirmo eksperimento apribojimai buvo pašalinti ir pridėti 2 nauji apribojimai:

- Vienas darbuotojas dirba 8 valandas.
- Darbuotojas visą savo darbo laiką dirba vienoje iš dviejų 12 valandų pamainų.

Formaliai tikslo funkcijos reikšmė būtų aprašoma:

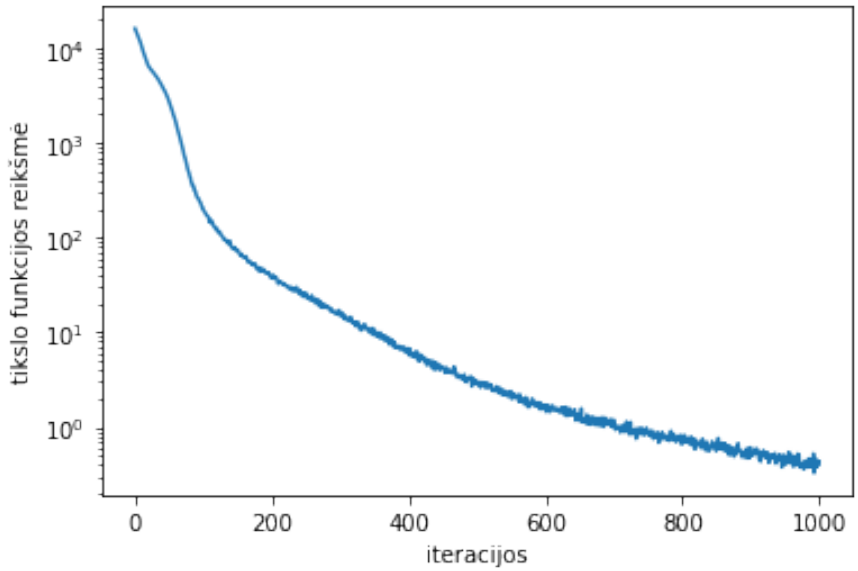
$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} + \alpha_2 \mathcal{L}_{cr,2} \tag{4}$$

$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} \left(\left(\sum_{t=1}^{T} \hat{S}_{t,p} \right) - 8 \right)^2 \tag{5}$$

$$\mathcal{L}_{cr,2} = \sum_{p=1}^{P} \left(\min \left(\sum_{t=1}^{12} \hat{S}_{t,p}, \sum_{t=13}^{24} \hat{S}_{t,p} \right) \right) \tag{6}$$

Čia $\alpha_k = 1, k = 1, 2$.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (6 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, visi iš šių tvarkaraščių atitiko abu apribojimus (4 lentelė).



6 pav.: Antro eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240
$\mathcal{L}_{cr,2}$	10240 iš 10240

4 lentelė: Antro eksperimento rezultatai

4.2.3 Trečias eksperimentas

Šis eksperimentas buvo atliekamas antro eksperimento sąlygomis, bet buvo pridėta naujų apribojimų. Eksperimento tikslas apsirašyti apribojimus dėl to kiek darbuotojų gali dirbti vienu metu. Buvo pridėti 2 papildomi apribojimai:

- Vienu metu dirba ne daugiau darbuotojų nei skaičius gaunamas pagal formulę:

$$k_t = P - 2, t = 1, ..., T$$

- Kiekvieną valandą dirba bent vienas darbuotojas.

Formaliai tikslo funkcijos reikšmė būtų aprašoma:

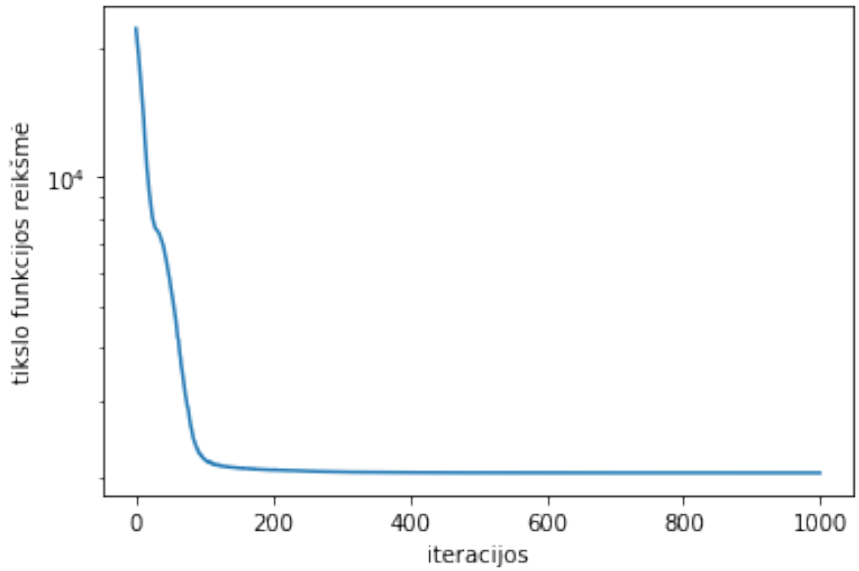
$$\mathcal{L} = \alpha_1\mathcal{L}_{cr,1} + \alpha_2\mathcal{L}_{cr,2} + \alpha_3\mathcal{L}_{cr,3} + \alpha_4\mathcal{L}_{cr,4} \quad (7)$$

$$\mathcal{L}_{cr,3} = \sum_{t=1}^{T} ReLU\left(\left(\sum_{p=1}^{P}\hat{S}_{t,p}\right) - k_t\right) \quad (8)$$

$$\mathcal{L}_{cr,4} = \sum_{t=1}^{T} ReLU\left(1 - \left(\sum_{p=1}^{P}\hat{S}_{t,p}\right)\right) \quad (9)$$

Čia $\alpha_k = 1, k = 1, 2, 3, 4$.

Atlikus eksperimentą pastebėta, kad neretai tikslo funkcija nėra išmokstama ir užstringa lokaliajame minimume (7 pav.). Pabandžius iš tikslo funkcijos pašalinti po vieną apribojimą ir palikti kitus 3, paaiškėjo, kad nėra išmokstamas apribojimas $\mathcal{L}_{cr,4}$. Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, visi sugeneruoti tvarkaraščiai tenkino tik $\mathcal{L}_{cr,2}$ ir $\mathcal{L}_{cr,3}$ apribojimus (5 lentelė).



7 pav.: Trečio eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	20 iš 10240
$\mathcal{L}_{cr,2}$	10240 iš 10240
$\mathcal{L}_{cr,3}$	10240 iš 10240
$\mathcal{L}_{cr,4}$	303 iš 10240

5 lentelė: Trečio eksperimento rezultatai

4.2.4 Ketvirtas eksperimentas

Šis eksperimentas buvo atliekamas trečio eksperimento sąlygomis. Eksperimento tikslas pabandyti išspręsti problemą, kad nėra išmokstamas apribojimas $\mathcal{L}_{cr,4}$ dėl bent vieno dirbančio darbuotojo per valandą. Buvo pabandyti įvairūs problemos sprendimo būdai:

- Keisti mokymo žingsnį.
- Pridėti mokymo žingsnio keitimo algoritmą, kuris automatiškai koreguos mokymo žingsnį.

- Padauginti apribojimą iš svorio.

Pabandžius padidinti mokymo žingsnį iki 0.01 išliko ta pati problema, o sumažinus jį iki 0.0001 $\mathcal{L}_{cr,4}$ apribojimą pavyko išmokti, bet vis dar nebuvo išmokstamas $\mathcal{L}_{cr,1}$ apribojimas (6 lentelė).

Mokymo žingsnis	Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
0.01	$\mathcal{L}_{cr,1}$	0 iš 10240
0.01	$\mathcal{L}_{cr,2}$	10240 iš 10240
0.01	$\mathcal{L}_{cr,3}$	10240 iš 10240
0.01	$\mathcal{L}_{cr,4}$	0 iš 10240
0.0001	$\mathcal{L}_{cr,1}$	6377 iš 10240
0.0001	$\mathcal{L}_{cr,2}$	10240 iš 10240
0.0001	$\mathcal{L}_{cr,3}$	10240 iš 10240
0.0001	$\mathcal{L}_{cr,4}$	10234 iš 10240

6 lentelė: Ketvirto eksperimento rezultatai keičiant mokymo žingsnį

Kitas galimas sprendimo būdas - pridėti mokymo žingsnio keitimo algoritmą. Buvo pabandyta pridėti du algoritmus: *CosineAnnealingLR*, kuris nustato atskirą mokymo žingsnį kiekvienai parametrui grupei pagal kosinusų atkaitinimą ir *ReduceLROnPlateau*, kuris keičia mokymo žingsnį, kai rezultatai negerėja per tam tikrą iteracijų kiekį. Šis sprendimo būdas nepadėjo ir abiem atvejais buvo neišmoktų apribojimų (7 lentelė).

Algoritmas	Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
CosineAnnealingLR	$\mathcal{L}_{cr,1}$	196 iš 10240
CosineAnnealingLR	$\mathcal{L}_{cr,2}$	0 iš 10240
CosineAnnealingLR	$\mathcal{L}_{cr,3}$	10240 iš 10240
CosineAnnealingLR	$\mathcal{L}_{cr,4}$	4250 iš 10240
ReduceLROnPlateau	$\mathcal{L}_{cr,1}$	226 iš 10240
ReduceLROnPlateau	$\mathcal{L}_{cr,2}$	10240 iš 10240
ReduceLROnPlateau	$\mathcal{L}_{cr,3}$	10240 iš 10240
ReduceLROnPlateau	$\mathcal{L}_{cr,4}$	648 iš 10240

7 lentelė: Ketvirto eksperimento rezultatai pridėjus mokymo žingsnio keitimo algoritmą

Dar vienas galimas sprendimo būdas padauginti apribojimą $\mathcal{L}_{cr,4}$ iš svorio. Buvo pabandytos kelios svorio α_4 reikšmės: 10, 100, 500. Atlikus eksperimentą paaiškėjo, kad pridėtas

svoris padeda išmokti $\mathcal{L}_{cr,4}$ apribojimą (jis buvo išmoktas su visomis svorio reikšmėmis), o pritaikius svorį $\alpha_4 = 500$ iš sugeneruotų 10240 tvarkaraščių net 10203 atitiko visus apribojimus (8 lentelė). Pabandžius šį svorį pritaikyti tik mokymo pradžioje (pirmose 50 iteracijų) ir sugeneravus 10240 tvarkaraščių, visus 4 apribojimus tenkino visi tvarkaraščiai.

Svoris	Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
10	$\mathcal{L}_{cr,1}$	8411 iš 10240
10	$\mathcal{L}_{cr,2}$	0 iš 10240
10	$\mathcal{L}_{cr,3}$	10240 iš 10240
10	$\mathcal{L}_{cr,4}$	10240 iš 10240
100	$\mathcal{L}_{cr,1}$	8645 iš 10240
100	$\mathcal{L}_{cr,2}$	10240 iš 10240
100	$\mathcal{L}_{cr,3}$	10240 iš 10240
100	$\mathcal{L}_{cr,4}$	10240 iš 10240
500	$\mathcal{L}_{cr,1}$	10203 iš 10240
500	$\mathcal{L}_{cr,2}$	10240 iš 10240
500	$\mathcal{L}_{cr,3}$	10240 iš 10240
500	$\mathcal{L}_{cr,4}$	10240 iš 10240
500 (taikoma tik 50 iteracijų)	$\mathcal{L}_{cr,1}$	10240 iš 10240
500 (taikoma tik 50 iteracijų)	$\mathcal{L}_{cr,2}$	10240 iš 10240
500 (taikoma tik 50 iteracijų)	$\mathcal{L}_{cr,3}$	10240 iš 10240
500 (taikoma tik 50 iteracijų)	$\mathcal{L}_{cr,4}$	10240 iš 10240

8 lentelė: Ketvirto eksperimento rezultatai keičiant $\mathcal{L}_{cr,4}$ apribojimo svorį

4.2.5 Penktas eksperimentas

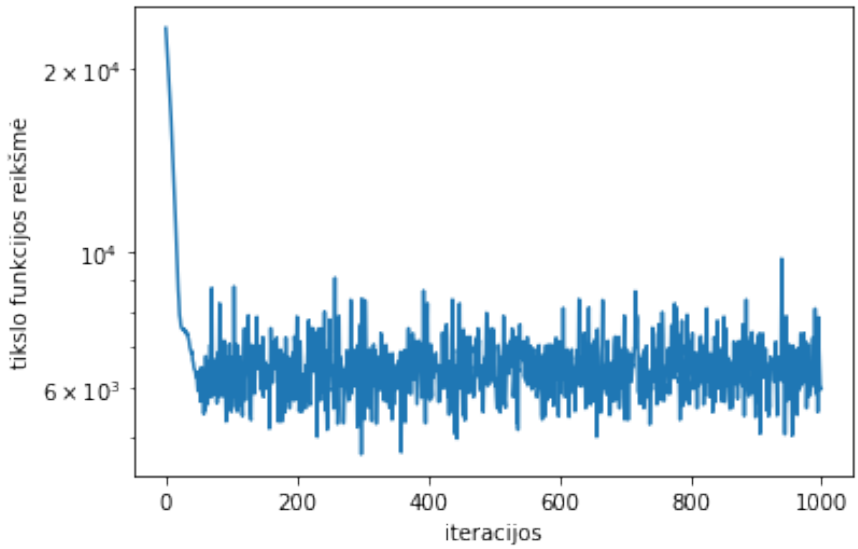
Atliekant kitus eksperimentus buvo pastebėta, kad visos sugeneruotos matricos tenkinančios visus apribojimus yra vienodos. Šis eksperimentas buvo atliekamas ketvirto eksperimento sąlygomis pritaikius svorį $\alpha_4 = 500$ apribojimui $\mathcal{L}_{cr,4}$. Eksperimento tikslas pabandyti išspręsti problemą, kad neuroninis tinklas sugeneruoja tik vieną unikalų sprendinį. Tam buvo pridėtas papildomas apribojimas, kuris įvertina ar sugeneruotos matricos yra unikalios. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1\mathcal{L}_{cr,1} + \alpha_2\mathcal{L}_{cr,2} + \alpha_3\mathcal{L}_{cr,3} + \alpha_4\mathcal{L}_{cr,4} + \alpha_5\mathcal{L}_{cr,5} \quad (10)$$

$$\mathcal{L}_{cr,5} = \#\text{unikalių sprendinių duomenų rinkinyje} \quad (11)$$

Čia $\alpha_1, \alpha_2, \alpha_3, \alpha_5 = 1, \alpha_4 = 500$.

Pridėjus apribojimą $\mathcal{L}_{cr,5}$ modelis tikslo funkcijos neišmoko ir užstrigo lokaliajame minimume (8 pav.). Sugeneravus 10240 tvarkaraščių, jie tenkino tik $\mathcal{L}_{cr,3}$ apribojimą (9 lentelė).



8 pav.: Penkto eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	5 iš 10240
$\mathcal{L}_{cr,2}$	0 iš 10240
$\mathcal{L}_{cr,3}$	10240 iš 10240
$\mathcal{L}_{cr,4}$	114 iš 10240
$\mathcal{L}_{cr,5}$	4945 iš 10240

9 lentelė: Penkto eksperimento rezultatai

Buvo pabandyti problemos sprendimo būdai:

- Papildomą apribojimą padauginti iš svorio.
- Papildomą apribojimą taikyti tik dalį iteracijų, o toliau mokyti be jo.

Pabandžius $\mathcal{L}_{cr,5}$ apribojimą padauginti iš svorio, α_5 priskyrus reikšmės 100, 500 ir 1000, geriausias rezultatas buvo gautas su svoriu $\alpha_5 = 500$ (10 lentelė). Nors modelis išmoko generuoti unikalias matricas jos nebetenkina kitų apribojimų.

Svoris	Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
100	$\mathcal{L}_{cr,1}$	6 iš 10240
100	$\mathcal{L}_{cr,2}$	0 iš 10240
100	$\mathcal{L}_{cr,3}$	10240 iš 10240
100	$\mathcal{L}_{cr,4}$	1019 iš 10240
100	$\mathcal{L}_{cr,5}$	7595 iš 10240
500	$\mathcal{L}_{cr,1}$	26 iš 10240
500	$\mathcal{L}_{cr,2}$	3189 iš 10240
500	$\mathcal{L}_{cr,3}$	10101 iš 10240
500	$\mathcal{L}_{cr,4}$	904 iš 10240
500	$\mathcal{L}_{cr,5}$	9558 iš 10240
1000	$\mathcal{L}_{cr,1}$	3 iš 10240
1000	$\mathcal{L}_{cr,2}$	0 iš 10240
1000	$\mathcal{L}_{cr,3}$	10239 iš 10240
1000	$\mathcal{L}_{cr,4}$	1545 iš 10240
1000	$\mathcal{L}_{cr,5}$	9194 iš 10240

10 lentelė: Penkto eksperimento rezultatai keičiant $\mathcal{L}_{cr,5}$ apribojimo svorį

Dar vienas būdas pagerinti rezultatą - pradėti mokyti modelį taikant papildomą apribojimą dėl matricos unikalumo, o vėliau jo nebetaikyti. Šiam eksperimentui buvo paliktas svoris $\alpha_5 = 500$ ir pabandyta taikyti $\mathcal{L}_{cr,5}$ apribojimą tik pirmose 100, 200 ir 300 iteracijų. Šis pakeitimas nepagerino rezultato ir geresnis rezultatas buvo gautas apribojimą taikant viso mokymo metu (11 lentelė).

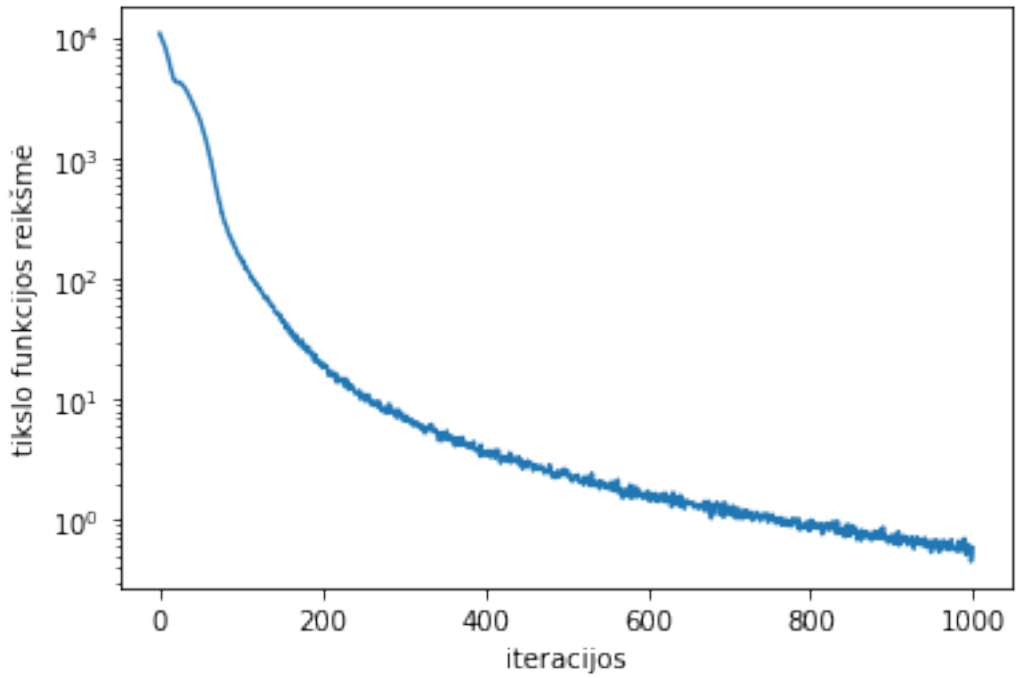
Iteracijų kiekis taikant $\mathcal{L}_{cr,5}$	Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
100 iš 1000	$\mathcal{L}_{cr,1}$	190 iš 10240
100 iš 1000	$\mathcal{L}_{cr,2}$	0 iš 10240
100 iš 1000	$\mathcal{L}_{cr,3}$	10220 iš 10240
100 iš 1000	$\mathcal{L}_{cr,4}$	4554 iš 10240
100 iš 1000	$\mathcal{L}_{cr,5}$	8640 iš 10240

200 iš 1000	$\mathcal{L}_{cr,1}$	1 iš 10240
200 iš 1000	$\mathcal{L}_{cr,2}$	0 iš 10240
200 iš 1000	$\mathcal{L}_{cr,3}$	10203 iš 10240
200 iš 1000	$\mathcal{L}_{cr,4}$	5525 iš 10240
200 iš 1000	$\mathcal{L}_{cr,5}$	8899 iš 10240
300 iš 1000	$\mathcal{L}_{cr,1}$	53 iš 10240
300 iš 1000	$\mathcal{L}_{cr,2}$	0 iš 10240
300 iš 1000	$\mathcal{L}_{cr,3}$	10235 iš 10240
300 iš 1000	$\mathcal{L}_{cr,4}$	5036 iš 10240
300 iš 1000	$\mathcal{L}_{cr,5}$	7468 iš 10240

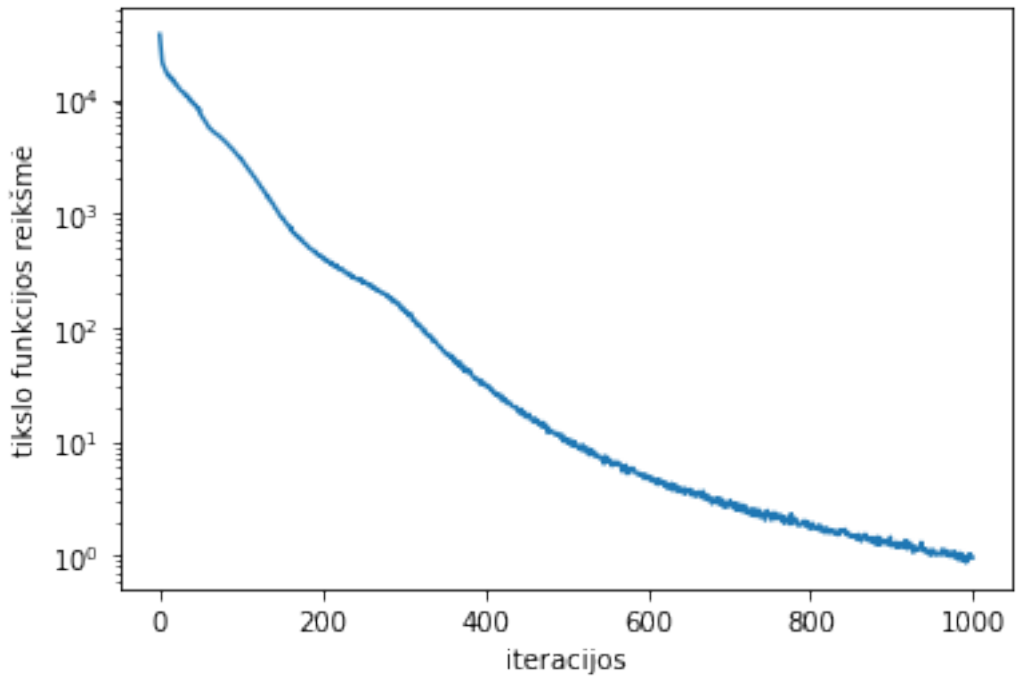
11 lentelė: Penkto eksperimento rezultatai taikant $\mathcal{L}_{cr,5}$ apribojimą nustatytą iteracijų kiekį

4.2.6 Šeštas eksperimentas

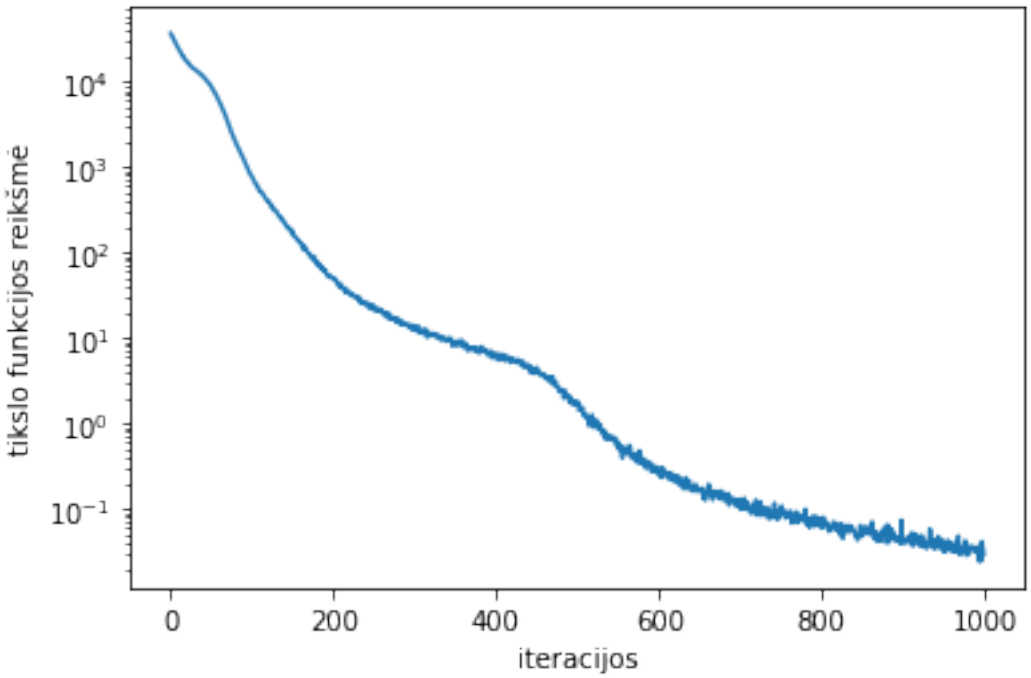
Šis eksperimentas buvo atliekamas ketvirto eksperimento sąlygomis pritaikius svorį $\alpha_4 = 500$ apribojimui $\mathcal{L}_{cr,4}$. Eksperimento tikslas įvertinti darbuotojų kiekio didinimo įtaką modelio mokymui ir mokymo greičiui. Darbuotojų kiekis buvo didinamas logaritminėje skalėje nuo 2 iki 32. Kai buvo nustatyti 2 darbuotojai nebuvo taikomas apribojimas $\mathcal{L}_{cr,4}$, nes jis šiuo atveju negali būti tenkinamas. Modelis išmoko tikslo funkciją su visais nustatytais darbuotojų kiekiais (9-13 pav.). Sugeneravus 10240 tvarkaraščių visi iš jų tenkino visus apribojimus iki 16 darbuotojų, o su 32 darbuotojais $\mathcal{L}_{cr,1}$ apribojimą tenkino 10153 tvarkaraščių (12 lentelė). Modelio mokymas užtrunka nuo 4 sekundžių, kai nustatyti 2 darbuotojai iki 130 sekundžių, kai nustatyti 32 darbuotojai, o vieno sprendinio radimo greitis užtrunka sekundės dalis su visais darbuotojų kiekiais (13 lentelė).



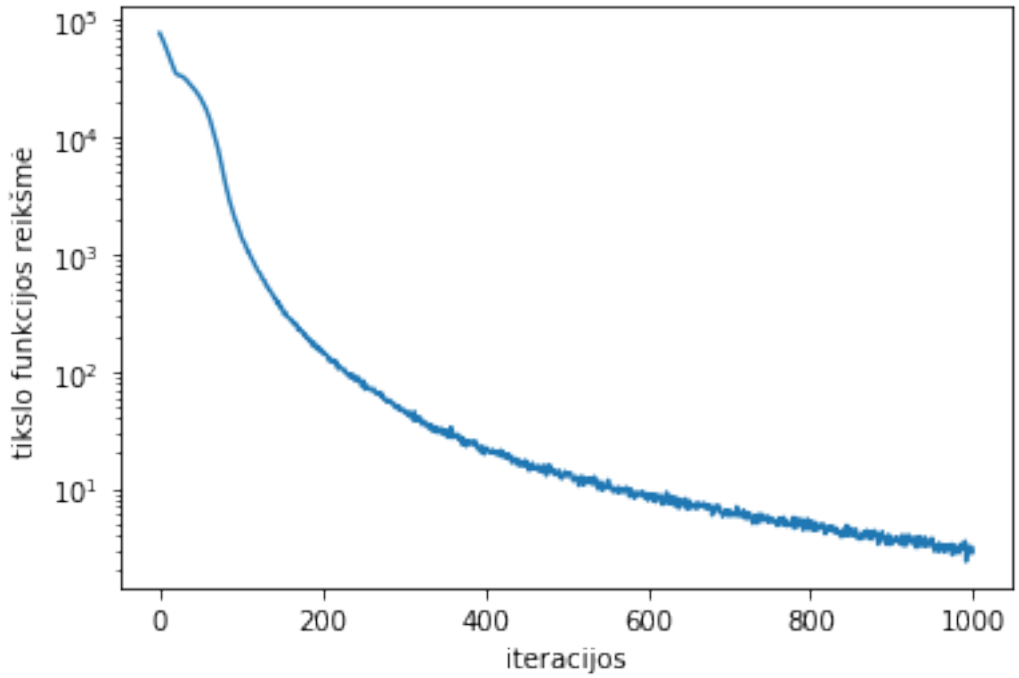
9 pav.: Modelio mokymo rezultatai su 2 darbuotojais



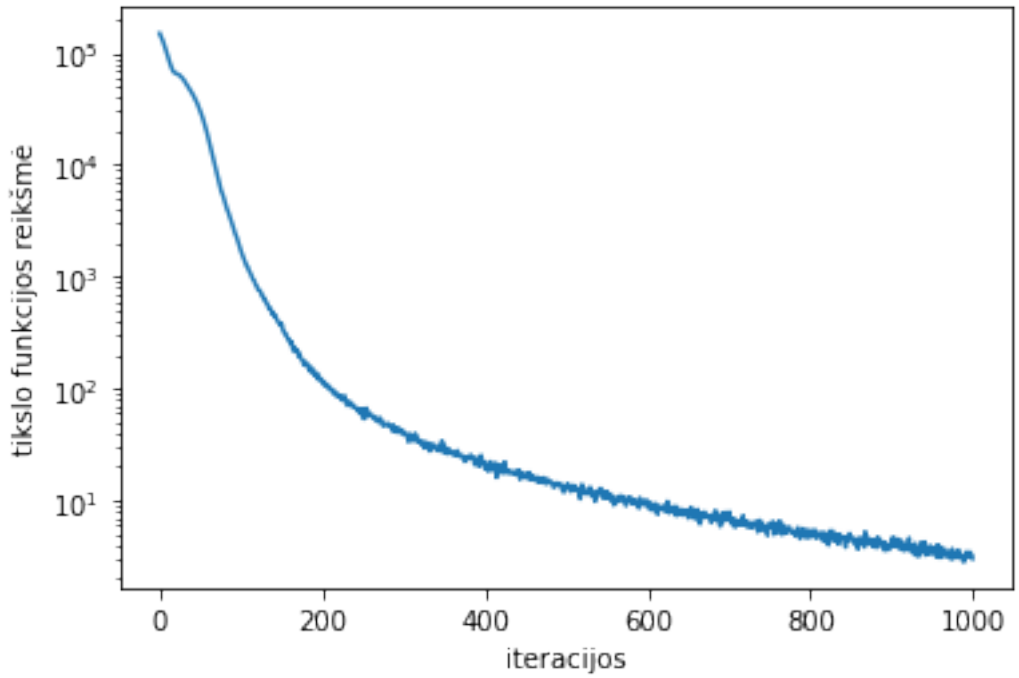
10 pav.: Modelio mokymo rezultatai su 4 darbuotojais



11 pav.: Modelio mokymo rezultatai su 8 darbuotojais



12 pav.: Modelio mokymo rezultatai su 16 darbuotojų



13 pav.: Modelio mokymo rezultatai su 32 darbuotojais

Darbuotojų kiekis	Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
2	$\mathcal{L}_{cr,1}$	10240 iš 10240
2	$\mathcal{L}_{cr,2}$	10240 iš 10240
2	$\mathcal{L}_{cr,3}$	10240 iš 10240
2	$\mathcal{L}_{cr,4}$	-
4	$\mathcal{L}_{cr,1}$	10240 iš 10240
4	$\mathcal{L}_{cr,2}$	10240 iš 10240
4	$\mathcal{L}_{cr,3}$	10240 iš 10240
4	$\mathcal{L}_{cr,4}$	10240 iš 10240
8	$\mathcal{L}_{cr,1}$	10240 iš 10240
8	$\mathcal{L}_{cr,2}$	10240 iš 10240
8	$\mathcal{L}_{cr,3}$	10240 iš 10240
8	$\mathcal{L}_{cr,4}$	10240 iš 10240
16	$\mathcal{L}_{cr,1}$	10240 iš 10240
16	$\mathcal{L}_{cr,2}$	10240 iš 10240
16	$\mathcal{L}_{cr,3}$	10240 iš 10240
16	$\mathcal{L}_{cr,4}$	10240 iš 10240

32	$\mathcal{L}_{cr,1}$	10153 iš 10240
32	$\mathcal{L}_{cr,2}$	10240 iš 10240
32	$\mathcal{L}_{cr,3}$	10240 iš 10240
32	$\mathcal{L}_{cr,4}$	10240 iš 10240

12 lentelė: Šešto eksperimento rezultatai

Darbuotojų kiekis	Modelio mokymo laikas (s)	Vieno sprendinio radimo laikas (s)
2	4,76	0,000236
4	6,67	0,000253
8	10,37	0,000377
16	18,8	0,000409
32	130,86	0.000722

13 lentelė: Modelio mokymo ir sprendinio radimo laikas

4.3 Realaus uždavinio eksperimentai

Atlikus pradinius eksperimentus norint išbandyti siūlomą neuroninio tinklo modelį tvarkaraščių generavimui, pereita prie realistiškų praktikoje taikomų apribojimų sudarant pamainio darbo tvarkaraščius. Buvo atlikti eksperimentai bandant apmokyti neuroninį tinklą taikant po vieną tvarkaraščio apribojimą arba kelis susijusius apribojimus. Eksperimentų tikslas yra patikrinti ar apibrėžtos tikslo funkcijos gali būti išmoktos ir atlikti tam tikrus modelio architektūros ar parametrų pakeitimus, jei jų nepavyksta išmokti. Vėliau šiuos apribojimus bus galima apjungti į vieną uždavinį. Dėl sudėtingesnių apribojimų buvo pakoreguota ir modelio architektūra. Neuronų kiekis sluoksniuose buvo padidintas nuo 32 iki 64. Taip pat pridėta duomenų rinkinio normalizacija (angl. batch normalization) po pirmo ir antro sluoksnio, kuri gali palengvinti mokymą. Pakoreguota modelio architektūra nustačius duomenų rinkinio dydį 512, 168 valandas ir 4 darbuotojus yra pavaizduota 14 paveikslėlyje. Šiems eksperimentams yra taikomi tie patys techniniai modelio parametrai kaip ir pradiniams eksperimentams 4.2 skyriuje.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Generator                                [512, 168, 4]             --
├─Linear: 1-1                            [512, 64]                 4,160
├─BatchNorm1d: 1-2                       [512, 64]                 128
├─ReLU: 1-3                              [512, 64]                 --
├─Linear: 1-4                            [512, 64]                 4,160
├─BatchNorm1d: 1-5                       [512, 64]                 (recursive
├─ReLU: 1-6                              [512, 64]                 --
├─Linear: 1-7                            [512, 672]                43,680
├─Sigmoid: 1-8                           [512, 168, 4]             --
==========================================================================
```

14 pav.: Realaus uždavinio modelio architektūra

4.3.1 Pirmas eksperimentas

Šiam eksperimentui buvo nustatyti 4 darbuotojai ir 7 dienos (168 valandos). Eksperimentui buvo realizuoti 2 apribojimai dėl darbuotojų dirbamų valandų skaičiaus:

- Minimalus dirbamų valandų skaičius per tvarkaraščio periodą.
- Maksimalus dirbamų valandų skaičius per tvarkaraščio periodą.

Buvo nustatytas minimalus valandų skaičius $T_{min} = 20$ ir maksimalus valandų skaičius $T_{max} = 50$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

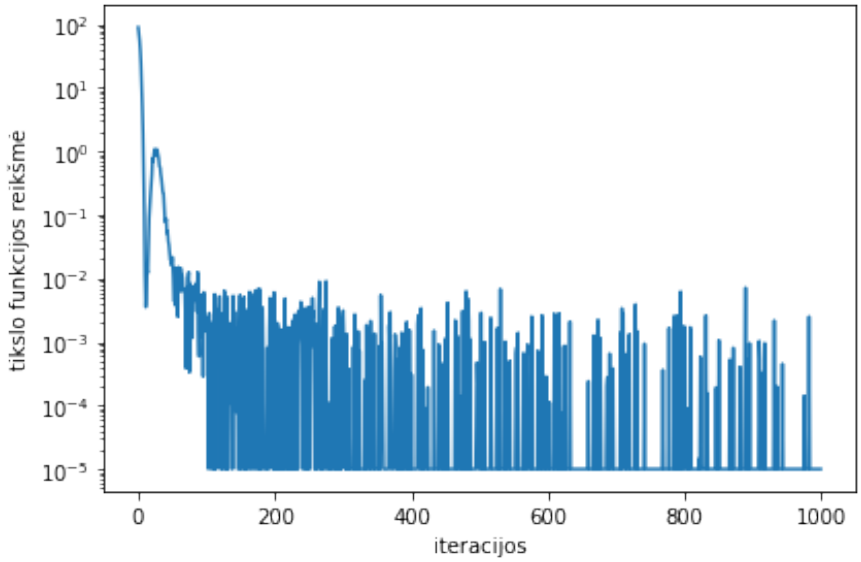
$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} + \alpha_2 \mathcal{L}_{cr,2} \tag{12}$$

$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} ReLU\left(T_{min} - \left(\sum_{t=1}^{T} \hat{S}_{t,p}\right)\right) \tag{13}$$

$$\mathcal{L}_{cr,2} = \sum_{p=1}^{P} ReLU\left(\left(\sum_{t=1}^{T} \hat{S}_{t,p}\right) - T_{max}\right) \tag{14}$$

Čia $\alpha_k = 1, k = 1, 2$.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (15 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, visi iš šių tvarkaraščių atitiko abu apribojimus (14 lentelė).



15 pav.: Pirmo eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240
$\mathcal{L}_{cr,2}$	10240 iš 10240

14 lentelė: Pirmo eksperimento rezultatai

4.3.2 Antras eksperimentas

Šiam eksperimentui buvo nustatyti 4 darbuotojai ir 7 dienos (168 valandos). Eksperimentui buvo realizuotas 1 apribojimas:

- Maksimalus dirbtų valandų skaičius per tam tikrą skaičių kalendorinių dienų.

Buvo nustatyta, kad per 3 dienas $D_k = 3$ maksimalus valandų skaičius būtų ne daugiau 30 $T_k = 30$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

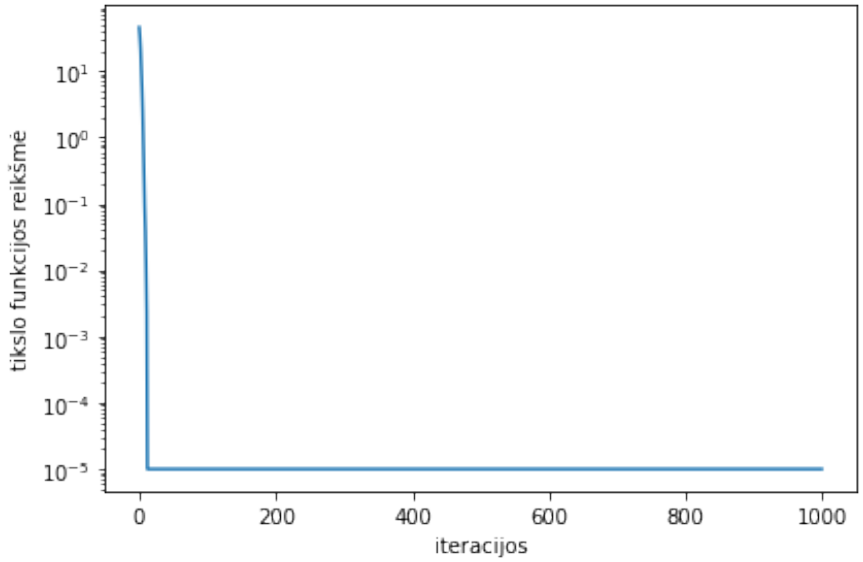
$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} \tag{15}$$

$$K = \{1, 24 \cdot D_k + 1\} \tag{16}$$

$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} \sum_{t' \in K} ReLU\left(\left(\sum_{t=t'}^{t'+24 \cdot D_k - 1} \hat{S}_{t,p}\right) - T_k\right) \tag{17}$$

Čia $\alpha_1 = 1$.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (16 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, visi iš šių tvarkaraščių atitiko apribojimą (15 lentelė).



16 pav.: Antro eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240

15 lentelė: Antro eksperimento rezultatai

4.3.3 Trečias eksperimentas

Šiam eksperimentui buvo nustatyti 4 darbuotojai ir 7 dienos (168 valandos). Eksperimentui buvo realizuoti 2 apribojimai:

- Minimalus dirbamų dienų skaičius tvarkaraščio periode.
- Maksimalus dirbamų dienų skaičius tvarkaraščio periode.

Diena laikoma dirbama, jei šią dieną yra dirbama bent vieną valandą. Buvo nustatytas 2 minimalus dienų skaičius $D_{min} = 2$ ir 6 maksimalus dienų skaičius $D_{max} = 6$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} + \alpha_2 \mathcal{L}_{cr,2} \tag{18}$$

$$D = \{1, 25, 49, \ldots, 145\} \tag{19}$$

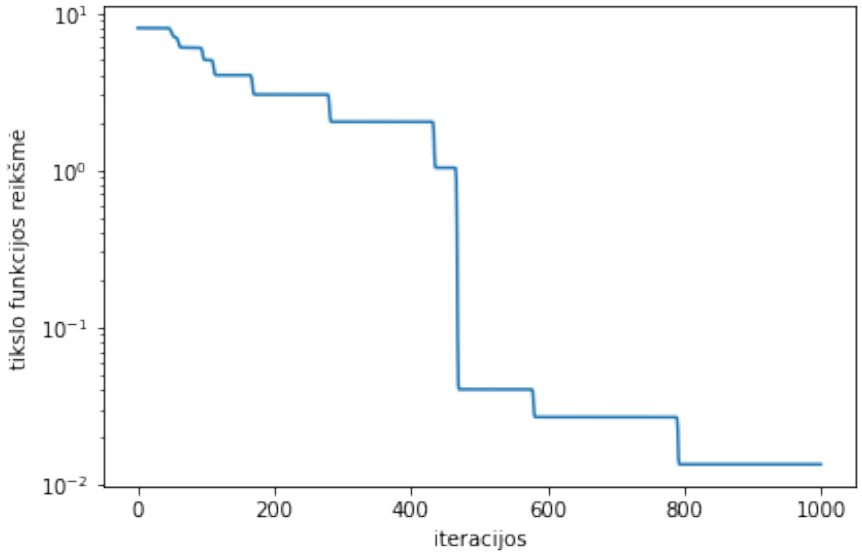
$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} ReLU\left(D_{min} - \left(\sum_{t' \in D} \sigma\left(\left(\left(\sum_{t=t'}^{t'+23} \hat{S}_{t,p}\right) - 0.5\right) \cdot \gamma\right)\right)\right) \tag{20}$$

$$\mathcal{L}_{cr,2} = \sum_{p=1}^{P} ReLU\left(\left(\sum_{t' \in D} \sigma\left(\left(\left(\sum_{t=t'}^{t'+23} \hat{S}_{t,p}\right) - 0.5\right) \cdot \gamma\right)\right) - D_{max}\right) \tag{21}$$

$$\tag{22}$$

Čia $\alpha_k = 1, k = 1, 2; \sigma(x)$ - sigmoidinė aktyvacijos funkcija; $\gamma = 100$.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (17 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, visi iš šių tvarkaraščių atitiko abu apribojimus (16 lentelė).



17 pav.: Trečio eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240
$\mathcal{L}_{cr,2}$	10240 iš 10240

16 lentelė: Trečio eksperimento rezultatai

4.3.4 Ketvirtas eksperimentas

Šiam eksperimentui buvo nustatyti 4 darbuotojai ir 7 dienos (168 valandos). Eksperimentui buvo realizuotas 1 apribojimas:

- Maksimalus iš eilės dirbamų dienų skaičius.

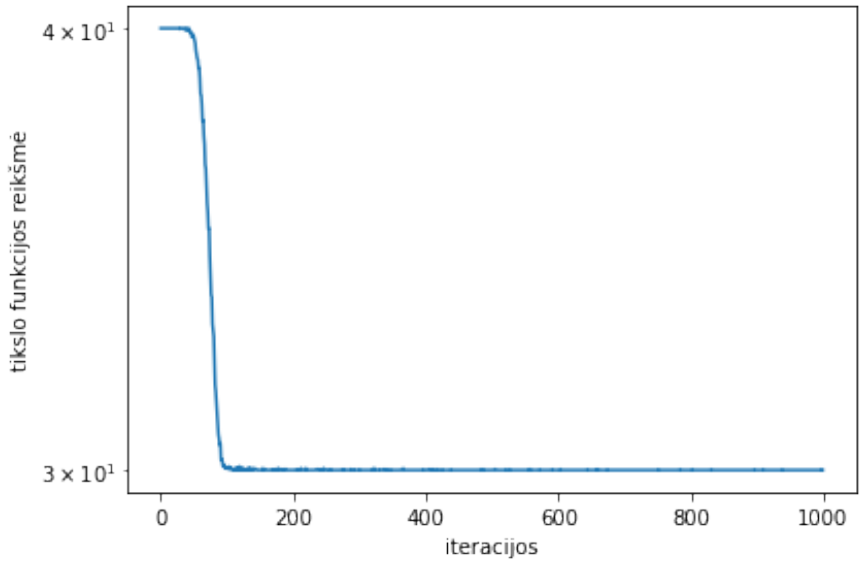
Diena laikoma dirbama, jei šią dieną yra dirbama bent vieną valandą. Eksperimentui buvo nustatytas maksimalaus iš eilės dirbamų dienų skaičius $D_e = 3$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} \tag{23}$$

$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} ReLU\left(E_{max,p} - D_e\right) \tag{24}$$

Čia $\alpha_1 = 1; E_{max,p}$ = p-tojo darbuotojo iš eilės dirbtų dienų skaičius.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija nėra išmokstama (18 pav.). Po neuroninio tinklo apmokymo sugeneravus 10240 tvarkaraščių, nei vienas iš jų netenkino apribojimo (17 lentelė).



18 pav.: Ketvirto eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	0 iš 10240

17 lentelė: Ketvirto eksperimento rezultatai

Norint išspręsti problemą buvo pabandyta mokyti kartu su kitais apribojimais. Buvo pridėti apribojimai iš pirmo eksperimento ir nustatyta, kad periode turi būti dirbama ne mažiau 40 valandų - $T_{min} = 40$ ir ne daugiau 60 valandų - $T_{max} = 60$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

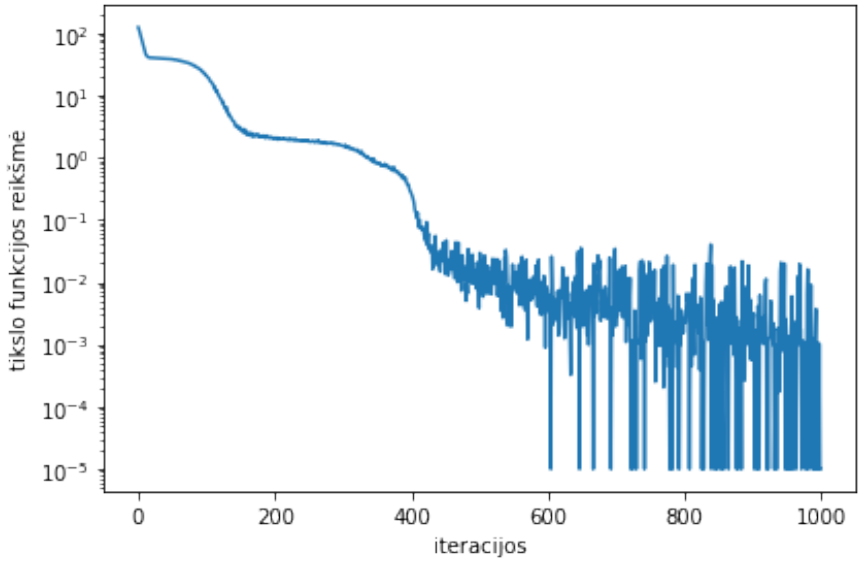
$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} + \alpha_2 \mathcal{L}_{cr,2} + \alpha_3 \mathcal{L}_{cr,3} \tag{25}$$

$$\mathcal{L}_{cr,2} = \sum_{p=1}^{P} ReLU\left(T_{min} - \left(\sum_{t=1}^{T} \hat{S}_{t,p}\right)\right) \tag{26}$$

$$\mathcal{L}_{cr,3} = \sum_{p=1}^{P} ReLU\left(\left(\sum_{t=1}^{T} \hat{S}_{t,p}\right) - T_{max}\right) \tag{27}$$

Čia $\alpha_k = 1, k = 1, 2, 3$.

Atlikus eksperimentą su papildomais apribojimais paaiškėjo, kad tikslo funkcija yra išmokstama (19 pav.). Po neuroninio tinklo apmokymo sugeneravus 10240 tvarkaraščių, visi jie tenkino $\mathcal{L}_{cr,2}$ ir $\mathcal{L}_{cr,3}$ apribojimą, o $\mathcal{L}_{cr,1}$ tenkino 10232 tvarkaraščių (18 lentelė).



19 pav.: Ketvirto eksperimento modelio mokymo rezultatai su papildomais apribojimais

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10232 iš 10240
$\mathcal{L}_{cr,2}$	10240 iš 10240
$\mathcal{L}_{cr,3}$	10240 iš 10240

18 lentelė: Ketvirto eksperimento rezultatai su papildomais apribojimais

4.3.5 Penktas eksperimentas

Šiam eksperimentui buvo nustatyti 4 darbuotojai ir 7 dienos (168 valandos). Eksperimentui buvo realizuotas 1 apribojimas:

- Maksimalus paeiliui einančių dirbamų naktinių pamainų skaičius.

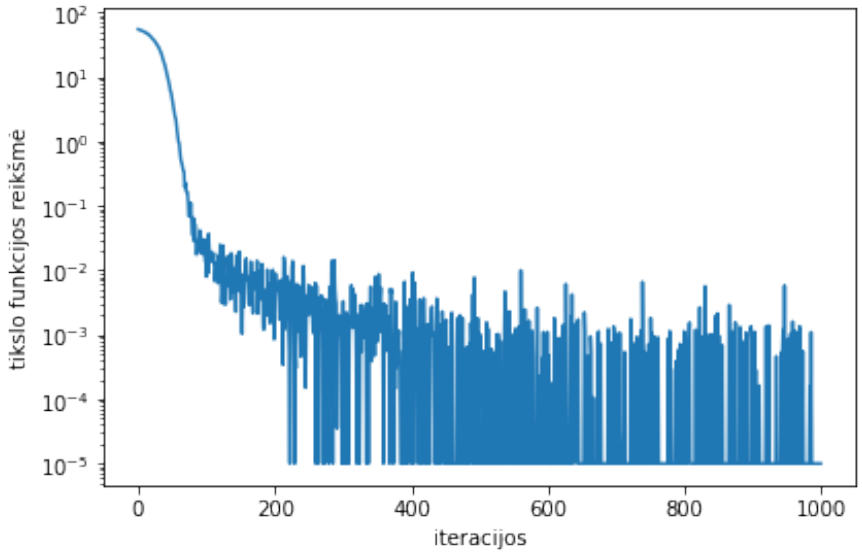
Naktine pamaina laikomas darbas nuo 22 valandos iki 6 valandos. Apribojimas taikomas, jei yra dirbama bent viena valanda naktinės pamainos metu nustatytą kiekį dienų iš eilės. Eksperimentui buvo nustatytas maksimalus paeiliui einančių dirbamų naktinių pamainų skaičius $D_n = 2$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} \tag{28}$$

$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} ReLU\left(N_{max,p} - D_n\right) \tag{29}$$

Čia $\alpha_1 = 1; N_{max,p} =$ p-tojo darbuotojo iš eilės dirbtų naktinių pamainų skaičius.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (20 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, 10235 iš šių tvarkaraščių atitiko apribojimą (19 lentelė).



20 pav.: Penkto eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10235 iš 10240

19 lentelė: Penkto eksperimento rezultatai

4.3.6 Šeštas eksperimentas

Šiam eksperimentui buvo nustatyti 4 darbuotojai ir 7 dienos (168 valandos). Eksperimentui buvo realizuotas 1 apribojimas:

- Maksimalus santykis tarp visame periode dirbtų valandų naktinėse ir dieninėse pamainose.

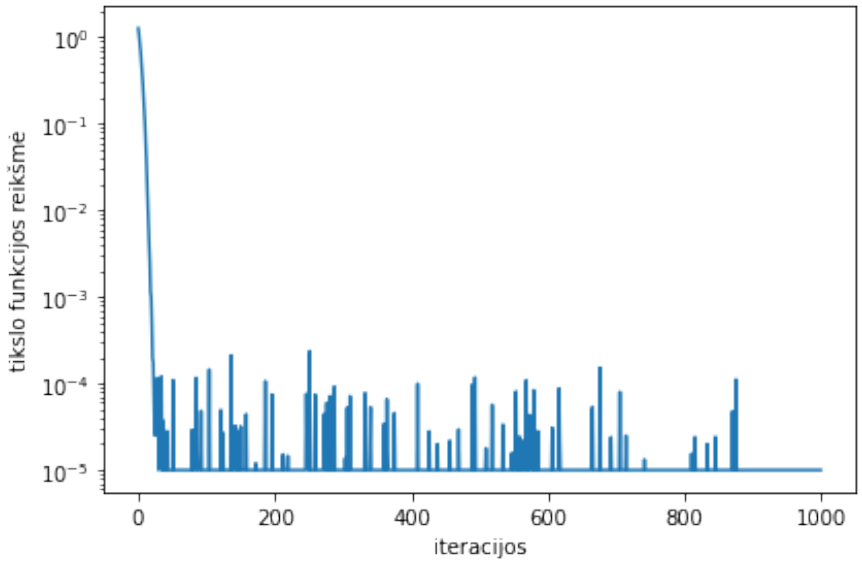
Dienine pamaina yra laikomas darbas nuo 8 iki 18 valandos, naktine pamaina - darbas valandomis nuo 22 iki 6. Eksperimentui buvo nustatytas maksimalus santykis $T_s = 0.5$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} \tag{30}$$

$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} ReLU\left(\frac{T_{n,p}}{T_{d,p}} - T_s\right) \tag{31}$$

Čia $\alpha_1 = 1; T_{n,p}$ = p-tojo darbuotojo dirbtų valandų naktinėse pamainose skaičius; $T_{d,p}$ = p-tojo darbuotojo dirbtų valandų dieninėse pamainose skaičius.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (21 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, visi iš šių tvarkaraščių atitiko apribojimą (20 lentelė).



21 pav.: Šešto eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240

20 lentelė: Šešto eksperimento rezultatai

4.3.7 Septintas eksperimentas

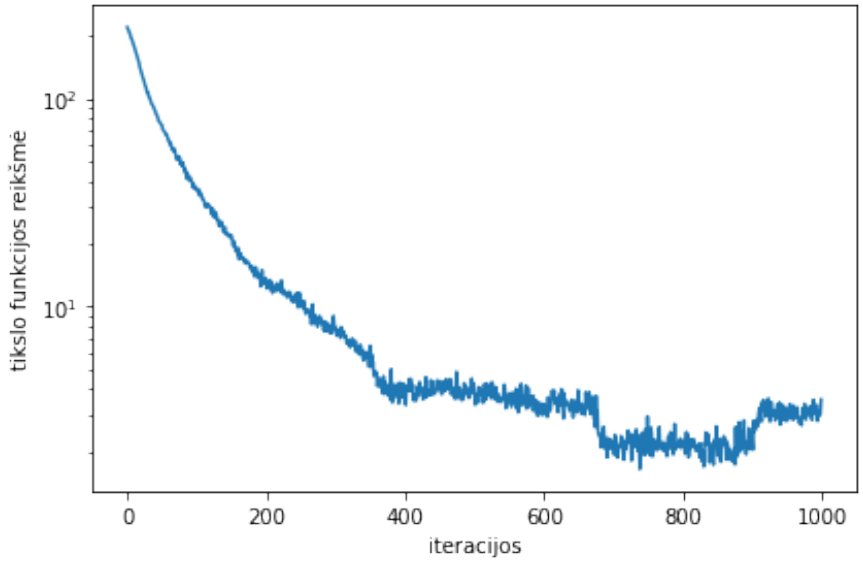
Šiam eksperimentui buvo nustatyti 4 darbuotojai ir 7 dienos (168 valandos). Eksperimentui buvo realizuotas sudėtingas apribojimas darbuotojų paskirstymui pamainomis. Šios pamainos sudarytos iš 10 valandų: 8 darbo valandų ir 2 valandų pertraukų. Šis apribojimas turi didelę įtaką neuroninio tinklo apmokymo laikui, todėl duomenų rinkinio dydis buvo sumažintas nuo 512 iki 64. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1 \mathcal{L}_{cr,1} \tag{32}$$

$$\mathcal{L}_{cr,1} = \#10 \text{ valandų pamainų sudarymo klaida} \tag{33}$$

Čia $\alpha_1 = 1$.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija nėra išmokstama (22 pav.). Modelis sėkmingai mokėsi iki 900 iteracijų, bet vėliau tikslo funkcijos reikšmė pradėjo didėti. Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, nei vienas iš jų netenkino apribojimo (21 lentelė).

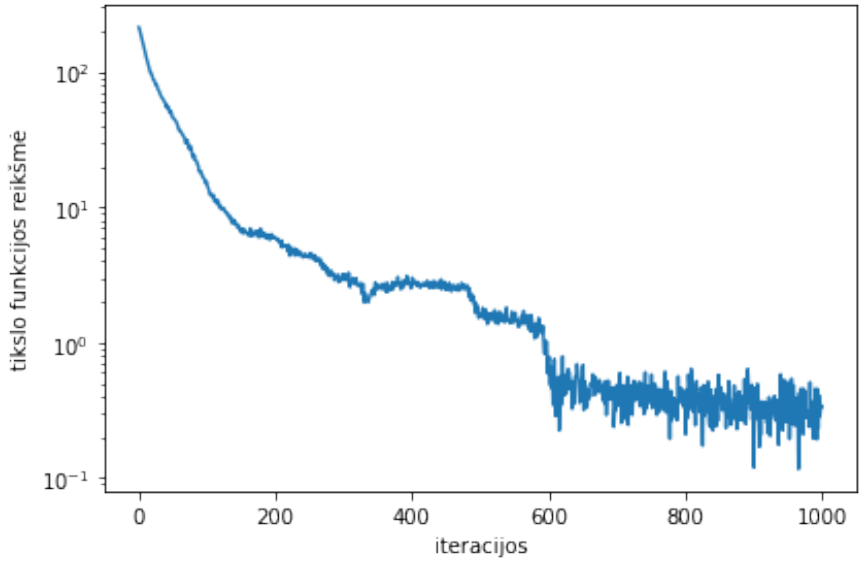


22 pav.: Septinto eksperimento modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	0 iš 10240

21 lentelė: Septinto eksperimento rezultatai

Norint, kad neuroninis tinklas sėkmingai išmoktų šį apribojimą buvo pakeista neuroninio tinklo architektūra. Buvo pridėti dar 2 papildomi 64×64 sluoksniai prieš paskutinį sluoksnį ir tada modelį sudarė 5 sluoksniai. Po šių pakeitimų modeliui pavyko išmokti apribojimą $\mathcal{L}_{cr,1}$ (23 pav.), o sugeneravus 10240 tvarkaraščių 9355 iš jų tenkino šį apribojimą (22 lentelė).



23 pav.: Septinto eksperimento modelio mokymo rezultatai pakeitus architektūrą

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	9355 iš 10240

22 lentelė: Septinto eksperimento rezultatai pakeitus architektūrą

Šiam eksperimentui taip pat buvo realizuoti 3 papildomi su pamainomis susiję apribojimai:

- Minimalus nepertraukiamo poilsio valandų skaičius po pamainos.
- Minimalus nepertraukiamo poilsio valandų skaičius per tam tikrą skaičių kalendorinių dienų.
- Minimalus vienu metu dirbamų pamainų kiekis.

Eksperimentui buvo nustatytas minimalus 2 valandų poilsio laikas po pamainos $T_p = 2$, nepertraukiamas pertraukų laikas 30 valandų $T_{np} = 30$ per 3 kalendorines dienas $D_k = 3$ ir bent viena dirbama pamaina vienu metu $P_d = 1$. Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1\mathcal{L}_{cr,1} + \alpha_2\mathcal{L}_{cr,2} + \alpha_3\mathcal{L}_{cr,3} + +\alpha_4\mathcal{L}_{cr,4} \tag{34}$$

$$K = \{1, 24 \cdot D_k + 1\} \tag{35}$$

$$\mathcal{L}_{cr,1} = \#10 \text{ valandų pamainų sudarymo klaida} \tag{36}$$

$$\mathcal{L}_{cr,2} = \sum_{p=1}^{P}\left(\sum_{i=1}^{M_p} ReLU\left(T_p - M_{i,v}\right)\right) \tag{37}$$

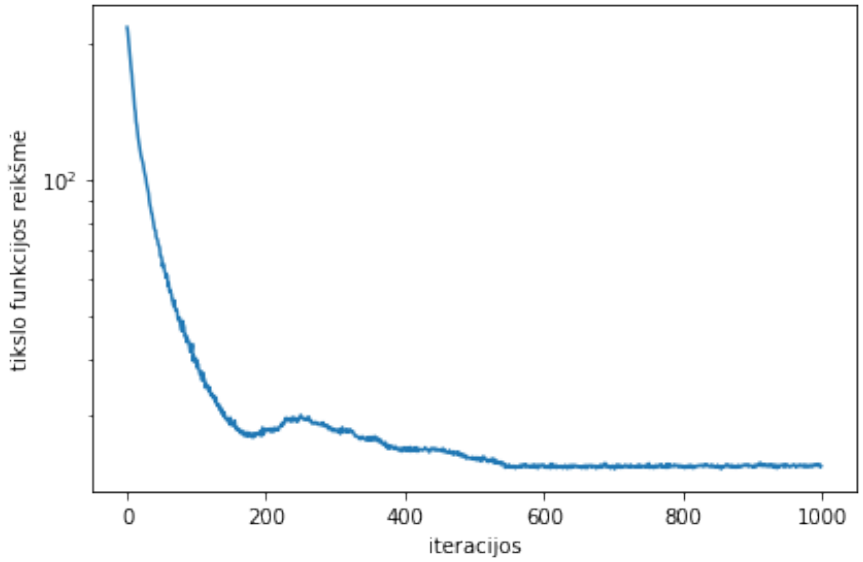
$$\mathcal{L}_{cr,3} = \sum_{p=1}^{P}\sum_{t'\in K} ReLU\left(T_{np} - \left(\sum_{t=t'}^{t'+24\cdot D_k-1}\dot{S}_{t,p}\right)\right) \tag{38}$$

$$\mathcal{L}_{cr,4} = \sum_{t=1}^{T} ReLU\left(P_d - \left(\sum_{p=1}^{P}\ddot{S}_{t,p}\right)\right) \tag{39}$$

$$\tag{40}$$

Čia $\alpha_k = 1, k = 1, 2, 3, 4$; M_p = p-tojo darbuotojo pamainų skaičius; $M_{i,v}$ = poilsio valandų skaičius po i-tosios pamainos; $\dot{S}$ = matrica (tvarkaraštis), kurioje visos valandos tarp pamainų yra 1, o visos pamainų valandos yra 0; $\ddot{S}$ = matrica (tvarkaraštis), kurioje visos pamainų valandos, įskaitant pertraukas yra 1, o visos kitos reikšmės yra 0;

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija nėra išmokstama (24 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, tik $\mathcal{L}_{cr,1}$ apribojimas buvo tenkinamas, bet jį tenkino tik 3254 tvarkaraščiai (23 lentelė).



24 pav.: Septinto eksperimento modelio mokymo rezultatai su papildomais apribojimais

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	3254 iš 10240
$\mathcal{L}_{cr,2}$	0 iš 10240
$\mathcal{L}_{cr,3}$	0 iš 10240
$\mathcal{L}_{cr,4}$	0 iš 10240

23 lentelė: Septinto eksperimento rezultatai su papildomais apribojimais

4.4 Tvarkaraščių sudarymas naudojant optimizavimo įrankį

Tvarkaraščių sudarymui optimizavimo algoritmais buvo pasirinktas optimizavimo įrankis *OptaPlanner*. Šis įrankis buvo pasirinktas dėl didelių konfigūracijos galimybių, galimybės pasirinkti optimizavimo algoritmą. *OptaPlanner* palaiko 3 tipų optimizavimo algoritmus: tiksliuosius algoritmus, euristinius algoritmus ir metaeuristinius algoritmus [32]. Galimi tikslieji algoritmai:

- išvardijimo algoritmas (angl. brute force),
- šakų ir ribų algoritmas.

Galimi euristiniai algoritmai:

- pirmo atitikmens (angl. first fit),
- silpniausio atitikmens (angl. weakest fit),

- stipriausio atitikmens (angl. strongest fit),
- pigiausio įterpimo (angl. cheapest insertion).

Galimi metaeuristinai algoritmai:

- kalnų laipiojimas (angl. hill climbing),
- tabu paieška,
- modeliuojamasis atkaitinimas,
- vėlyvo priėmimo (angl. late acceptance).

Algoritmus galima laisvai pasirinkti ir kombinuoti. Paprastai yra naudojamas vienas euristinis algoritmas, kuris sukonstruoja pradinį sprendinį ir vienas metaeuristinis algoritmas, kuris siekia pagerinti pradinį sprendinį. Šiame įrankyje buvo realizuotas tas pats pamaininio darbo tvarkaraščių sudarymo uždavinys ir visi apribojimai aprašyti 4.3 skyriuje. Buvo išbandyti keli optimizavimo algoritmai ir galiausiai buvo pasirinktas pirmo atitikmens algoritmas pradinio sprendinio radimui ir modeliuojamasis atkaitinimas jo optimizavimui. Modeliuojamojo atkaitinimo algoritmui buvo nustatyti 2 parametrai: pradinė temperatūra - *simulatedAnnealingStartingTemperature* ir atliktų veiksmų limitas - *acceptedCountLimit*. Pradinė temperatūra reiškia maksimalią reikšmę kuria gali padidėti bendra sprendinio klaida po atlikto veiksmo (angl. move), nustatyta reikšmė - 25. Atliktų veiksmų limitas reiškia kiek atliktų veiksmų turi būti patikrinta kiekviename algoritmo žingsnyje, nustatyta reikšmė - 4.

4.5 Neuroninio tinklo ir optimizavimo įrankio palyginimas

Norint įvertinti sudaryto neuroninio tinklo modelio galimybes generuoti pamaininio darbo tvarkaraščius buvo atliktas palyginimas su optimizavimo įrankiu *OptaPlanner* realizuojant tą pati uždavinį ir taikant daug apribojimų vienu metu. Kadangi ne visi 4.3 skyriaus apribojimai buvo išmokti neuroninio tinklo, buvo sudaryti 2 skirtingi uždaviniai. Pirmam uždaviniui buvo pritaikyti visi apribojimai kuriuos sėkmingai išmoko neuroninis tinklas (1-6 eksperimentai). Antram uždaviniui buvo taikomi visi 4.3 skyriaus apribojimai įskaitant pamainų apribojimus. Pirmame uždavinyje buvo lyginamas neuroninio tinklo ir *OptaPlanner* tvarkaraščių sudarymas. Kadangi neuroninis tinklas negali sugeneruoti teisingo visus antro uždavinio apribojimus atitinkančio tvarkaraščio, šis sugeneruotas tvarkaraštis buvo

paduodamas *OptaPlanner*, kuris toliau sprendė uždavinį. Antrame uždavinyje yra lyginamas tvarkaraščių sudarymas naudojant *OptaPlanner* su tvarkaraščių sudarymu paduodant pradinį sprendinį *OptaPlanner*, kuris buvo sugeneruotas neuroninio tinklo.

4.5.1 Pirmas uždavinys

Šiam uždaviniui buvo nustatytos 168 valandos (7 dienos) ir 4 darbuotojai. Uždaviniui buvo pritaikyti visi 1-6 eksperimentų iš 4.3 skyriaus apribojimai. Uždaviniui naudojama 14 pav. pavaizduota modelio architektūra ir duomenų rinkinio dydis - 512. Nustatyti apribojimai ir jų reikšmės:

- Minimalus dirbamų valandų skaičius per tvarkaraščio periodą - $T_{min} = 20$.
- Maksimalus dirbamų valandų skaičius per tvarkaraščio periodą - $T_{max} = 60$.
- Maksimalus dirbtų valandų skaičius per tam tikrą skaičių kalendorinių dienų - $T_k = 30, D_k = 3$.
- Minimalus dirbamų dienų skaičius tvarkaraščio periode - $D_{min} = 2$.
- Maksimalus dirbamų dienų skaičius tvarkaraščio periode - $D_{max} = 6$.
- Maksimalus iš eilės dirbamų dienų skaičius - $D_e = 5$.
- Maksimalus paeiliui einančių dirbamų naktinių pamainų skaičius - $D_n = 2$.
- Maksimalus santykis tarp dirbtų valandų naktinėje ir dieninėje pamainoje - $T_s = 0.5$.

Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\mathcal{L} = \alpha_1\mathcal{L}_{cr,1} + \alpha_2\mathcal{L}_{cr,2} + \alpha_3\mathcal{L}_{cr,3} + \alpha_4\mathcal{L}_{cr,4} + \alpha_5\mathcal{L}_{cr,5} + \alpha_6\mathcal{L}_{cr,6} \quad (41)$$

$$+ \alpha_7\mathcal{L}_{cr,7} + \alpha_8\mathcal{L}_{cr,8} \quad (42)$$

$$K = \{1, 24 \cdot D_k + 1\} \quad (43)$$

$$D = \{1, 25, 49, \ldots, 145\} \quad (44)$$

$$\mathcal{L}_{cr,1} = \sum_{p=1}^{P} ReLU\left(T_{min} - \left(\sum_{t=1}^{T} \hat{S}_{t,p}\right)\right) \quad (45)$$

$$\mathcal{L}_{cr,2} = \sum_{p=1}^{P} ReLU\left(\left(\sum_{t=1}^{T} \hat{S}_{t,p}\right) - T_{max}\right) \quad (46)$$

$$\mathcal{L}_{cr,3} = \sum_{p=1}^{P} \sum_{t' \in K} ReLU\left(\left(\sum_{t=t'}^{t'+24\cdot D_k-1} \hat{S}_{t,p}\right) - T_k\right) \quad (47)$$

$$\mathcal{L}_{cr,4} = \sum_{p=1}^{P} ReLU\left(D_{min} - \left(\sum_{t' \in D} \sigma\left(\left(\left(\sum_{t=t'}^{t'+23} \hat{S}_{t,p}\right) - 0.5\right) \cdot \gamma\right)\right)\right) \quad (48)$$

$$\mathcal{L}_{cr,5} = \sum_{p=1}^{P} ReLU\left(\left(\sum_{t' \in D} \sigma\left(\left(\left(\sum_{t=t'}^{t'+23} \hat{S}_{t,p}\right) - 0.5\right) \cdot \gamma\right)\right) - D_{max}\right) \quad (49)$$

$$\mathcal{L}_{cr,6} = \sum_{p=1}^{P} ReLU\left(E_{max,p} - D_e\right) \quad (50)$$

$$\mathcal{L}_{cr,7} = \sum_{p=1}^{P} ReLU\left(N_{max,p} - D_n\right) \quad (51)$$

$$\mathcal{L}_{cr,8} = \sum_{p=1}^{P} ReLU\left(\frac{T_{n,p}}{T_{d,p}} - T_s\right) \quad (52)$$

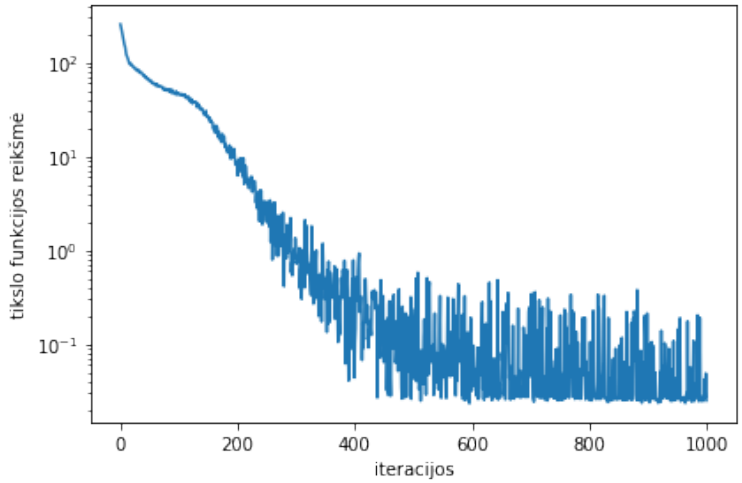
Čia $\alpha_k = 1, k = 1, 2, 3, 4, 5, 6, 7, 8$; $\sigma(x)$ - sigmoidinė aktyvacijos funkcija; $\gamma = 100$; $E_{max,p}$ = p-tojo darbuotojo iš eilės dirbtų dienų skaičius; $N_{max,p}$ = p-tojo darbuotojo iš eilės dirbtų naktinių pamainų skaičius; $T_{n,p}$ = p-tojo darbuotojo dirbtų valandų naktinėse pamainose skaičius; $T_{d,p}$ = p-tojo darbuotojo dirbtų valandų dieninėse pamainose skaičius.

Atlikus eksperimentą paaiškėjo, kad tikslo funkcija yra išmokstama (25 pav.). Apmokius neuroninį tinklą ir sugeneravus 10240 tvarkaraščių, kiekvieną iš apribojimų tenkino nuo 10103 tvarkaraščių iki visų 10240 tvarkaraščių (24 lentelė). Tas pat eksperimentas buvo atliktas ir su *OptaPlanner* optimizavimo įrankiu. Pamatavus abiejų uždavinio sprendimo metodų vykdymo greitį paaiškėjo, kad neuroninis tinklas sprendinį randą daug kartų greičiau, bendras vykdymo laikas - 56.23 s., o sprendinio radimo laikas apmokius neuroninį tinklą - 0.01 s. (25 lentelė). Tuo tarpu *OptaPlanner* sprendinį rado per 516.17 s.

Norint padaryti uždavinį sudėtingesniu buvo pridėtas dar vienas apribojimas:

- Procentas tvarkaraščio (matricos) pozicijų, kuriose negalima dirbti.

Šios matricos pozicijos buvo atsitiktinai generuojamos. Realizavus šį apribojimą neuroniniame tinkle ir *OptaPlanner* optimizavimo įrankyje, buvo atliktas dar vienas eksperimentas. Šio eksperimento tikslas patikrinti apribojimo įtaką sprendinio radimo laikui ir uždavinio išsprendžiamumui. Negalimų matricos pozicijų kiekis buvo didinamas nuo 10% iki 90% kas 10%. Atlikus eksperimentą paaiškėjo, kad neuroninio tinklo vykdymo laikui šis apribojimas neturi įtakos ir didinant procentą negalimų matricos pozicijų vykdymo laikas nedidėja, skirtingai nei *OptaPlanner*, kur vykdymo laikas didėja eksponentiškai (26 pav.). Nustačius 90% procentų *OptaPlanner* visai nerado sprendinio, o neuroninis tinklas neišmoko nustatytų apribojimų. Neuroniniam tinklui taip pat buvo apskaičiuotas vidurkis procentų kiekvieną apribojimą atitinkančių tvarkaraščių iš 10240 sugeneruotų. Šis procentas didinant negalimų pozicijų kiekį sumažėjo nuo 99.8% iki 99.2%, kol galiausiai nebebuvo galima sugeneruoti tinkamo sprendinio ir buvo gauta 40.5 % (26 pav.)



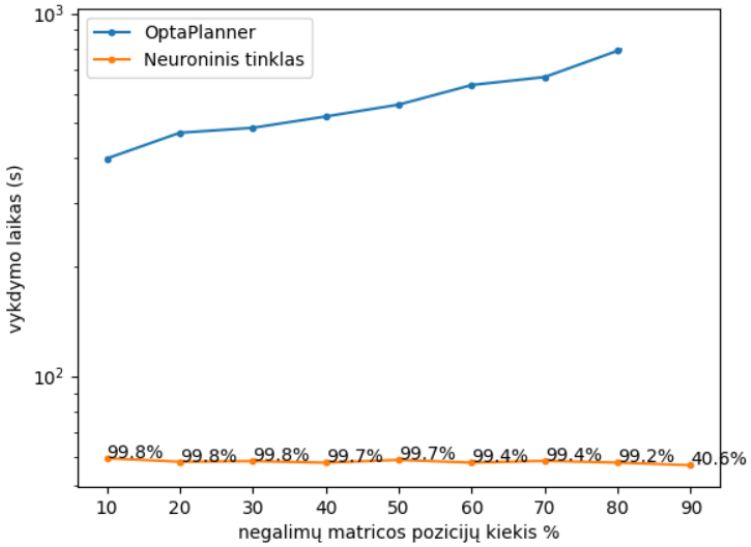
25 pav.: Pirmo uždavinio modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240
$\mathcal{L}_{cr,2}$	10240 iš 10240
$\mathcal{L}_{cr,3}$	10240 iš 10240
$\mathcal{L}_{cr,4}$	10240 iš 10240
$\mathcal{L}_{cr,5}$	10179 iš 10240
$\mathcal{L}_{cr,6}$	10181 iš 10240
$\mathcal{L}_{cr,7}$	10221 iš 10240
$\mathcal{L}_{cr,8}$	10103 iš 10240

24 lentelė: Pirmo uždavinio neuroninio tinklo rezultatai

Metodas	Mokymo laikas	1 sprendinio radimo laikas	Bendras vykdymo laikas
Neuroninis tinklas	56.23 s	0.01 s	56.24 s
OptaPlanner	-	-	516.71 s

25 lentelė: Neuroninio tinklo ir *OptaPlanner* pirmo uždavinio vykdymo laikas



26 pav.: Uždavinio su negalimomis matricos pozicijomis rezultatai

4.5.2 Antras uždavinys

Šiam uždaviniui buvo nustatytos 168 valandos (7 dienos) ir 4 darbuotojai. Uždaviniui buvo pritaikyti visi apribojimai aprašyti 4.3 skyriuje. Šiam uždaviniui naudojama modelio architektūra su 2 papildomais sluoksniais (27 pav.), nustatytas duomenų rinkinio dydis - 64.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Generator                                [64, 168, 4]              --
├─Linear: 1-1                            [64, 64]                  4,160
├─BatchNorm1d: 1-2                       [64, 64]                  128
├─ReLU: 1-3                              [64, 64]                  --
├─Linear: 1-4                            [64, 64]                  4,160
├─BatchNorm1d: 1-5                       [64, 64]                  (recursive)
├─ReLU: 1-6                              [64, 64]                  --
├─Linear: 1-7                            [64, 64]                  4,160
├─BatchNorm1d: 1-8                       [64, 64]                  (recursive)
├─ReLU: 1-9                              [64, 64]                  --
├─Linear: 1-10                           [64, 64]                  4,160
├─BatchNorm1d: 1-11                      [64, 64]                  (recursive)
├─ReLU: 1-12                             [64, 64]                  --
├─Linear: 1-13                           [64, 672]                 43,680
├─Sigmoid: 1-14                          [64, 168, 4]              --
==========================================================================
```

27 pav.: Antro uždavinio modelio architektūra

Pirmi 8 apribojimai ir jų reikšmės sutampa su pirmu uždaviniu. Nustatyti papildomi apribojimai ir jų reikšmės:

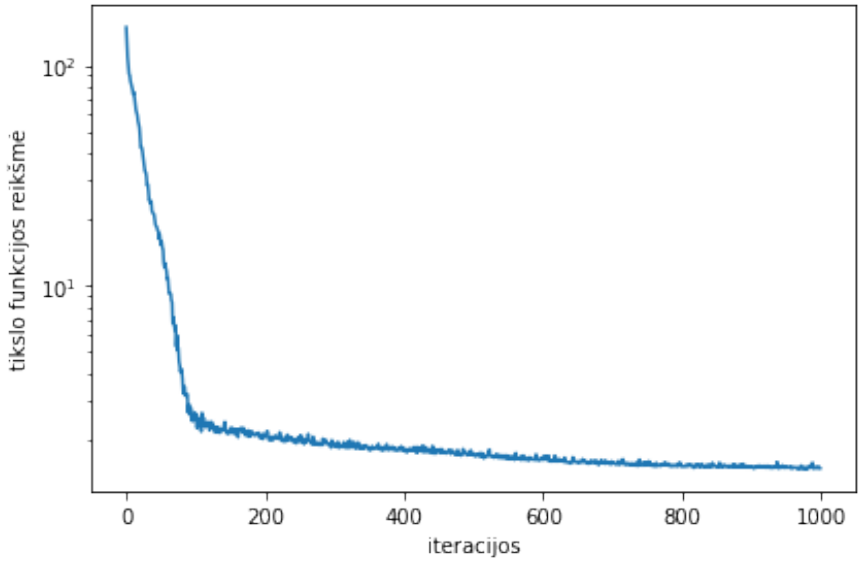
- Darbo suskirstymas 10 valandų pamainomis su 2 val pertraukomis.
- Minimalus nepertraukiamo poilsio valandų skaičius po pamainos - $T_p = 2$.
- Minimalus nepertraukiamo poilsio valandų skaičius per tam tikrą skaičių kalendorinių dienų - $T_{np} = 30$, $D_k = 3$.
- Minimalus vienu metu dirbamų pamainų kiekis - $P_d = 1$.

Formaliai tikslo funkcijos reikšmė būtų aprašoma:

$$\begin{aligned}
\mathcal{L} &= \alpha_1\mathcal{L}_{cr,1} + \alpha_2\mathcal{L}_{cr,2} + \alpha_3\mathcal{L}_{cr,3} + \alpha_4\mathcal{L}_{cr,4} + \alpha_5\mathcal{L}_{cr,5} + \alpha_6\mathcal{L}_{cr,6} && (53)\\
&+ \alpha_7\mathcal{L}_{cr,7} + \alpha_8\mathcal{L}_{cr,8} + \alpha_9\mathcal{L}_{cr,9} + \alpha_{10}\mathcal{L}_{cr,10} + \alpha_{11}\mathcal{L}_{cr,11} + \alpha_{12}\mathcal{L}_{cr,12} && (54)\\
K &= \{1, 24 \cdot D_k + 1\} && (55)\\
\mathcal{L}_{cr,9} &= \#10 \text{ valandų pamainų sudarymo klaida} && (56)\\
\mathcal{L}_{cr,10} &= \sum_{p=1}^{P}\left(\sum_{i=1}^{M_p} ReLU\left(T_p - M_{i,v}\right)\right) && (57)\\
\mathcal{L}_{cr,11} &= \sum_{p=1}^{P}\sum_{t'\in K} ReLU\left(T_{np} - \left(\sum_{t=t'}^{t'+24\cdot D_k-1} \dot{S}_{t,p}\right)\right) && (58)\\
\mathcal{L}_{cr,12} &= \sum_{t=1}^{T} ReLU\left(P_d - \left(\sum_{p=1}^{P} \ddot{S}_{t,p}\right)\right) && (59)\\
& && (60)
\end{aligned}$$

Čia $\alpha_k = 1, k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$; M_p = p-tojo darbuotojo pamainų skaičius; $M_{i,v}$ = poilsio valandų skaičius po i-tosios pamainos; $\dot{S}$ = matrica (tvarkaraštis), kurioje visos valandos tarp pamainų yra 1, o visos pamainų valandos yra 0; $\ddot{S}$ = matrica (tvarkaraštis), kurioje visos pamainų valandos, įskaitant pertraukas yra 1, o visos kitos reikšmės yra 0;

Atlikus eksperimentą paaiškėjo, kad neuroninis tinklas pilnai išmoko ne visus apribojimus, $L_{cr,2}$, $L_{cr,9}$ ir $L_{cr,12}$ netenkino nei vienas sugeneruotas tvarkaraštis (28 pav., 26 lentelė). Pridėti nauji pamainų apribojimai žymiai sulėtino tiek modelio mokymo greitį, tiek *OptaPlanner* sprendinio radimo greitį (27 lentelė) lyginant su pirmu uždaviniu. Apmokius neuroninį tinklą ir padavus gautą nepilną sprendinį OptaPlanner, kuris jį toliau optimizavo, rezultatas buvo gautas daugiau nei 3 kartus greičiau nei naudojant tik *OptaPlanner* optimizavimo įrankį.



28 pav.: Antro uždavinio modelio mokymo rezultatai

Apribojimas	Sugeneruoti tvarkaraščiai tenkinantys apribojimą
$\mathcal{L}_{cr,1}$	10240 iš 10240
$\mathcal{L}_{cr,2}$	0 iš 10240
$\mathcal{L}_{cr,3}$	10232 iš 10240
$\mathcal{L}_{cr,4}$	10240 iš 10240
$\mathcal{L}_{cr,5}$	10237 iš 10240
$\mathcal{L}_{cr,6}$	10237 iš 10240
$\mathcal{L}_{cr,7}$	10229 iš 10240
$\mathcal{L}_{cr,8}$	10240 iš 10240
$\mathcal{L}_{cr,9}$	0 iš 10240
$\mathcal{L}_{cr,10}$	7843 iš 10240
$\mathcal{L}_{cr,11}$	9866 iš 10240
$\mathcal{L}_{cr,12}$	0 iš 10240

26 lentelė: Antro uždavinio neuroninio tinklo rezultatai

Metodas	Modelio mokymo laikas	1 sprendinio radimo laikas	*OptaPlanner* vykdymo laikas	Bendras vykdymo laikas
Neuroninis tinklas ir *OptaPlanner*	634.84 s	0.01 s	967.71 s	1602.56 s
OptaPlanner	-	-	5000.66 s	5000.66 s

27 lentelė: Antro uždavinio vykdymo laikas

Rezultatai ir išvados

Atlikus darbą buvo pasiekti rezultatai:

- Suformuluotas tvarkaraščio sudarymo uždavinys bei apribojimai įvesti kaip tikslo funkcija.
- Pasiūlytas ir sudarytas naujas generatyvinis modelis pamaininio darbo tvarkaraščių generavimui (3 pav.), šis modelis praktiškai realizuotas (4 pav.)
- Atlikti eksperimentai modelyje realizavus tvarkaraščio apribojimus ir jį apmokant.
- Atliktas sukurto modelio ir optimizavimo įrankio *OptaPlanner* palyginimas sprendžiant 2 tvarkaraščio sudarymo uždavinius su dideliu kiekių apribojimų. Pirmo uždavinio atveju neuroninis tinklas sprendinį rado per 56.24 sekundes, įskaitant apmokymą, o optimizavimo įrankis per 516.71 sekundę. Antro uždavinio atveju sugeneravus pradinį sprendinį neuroniniu tinklu ir jį toliau optmizuojant su *OptaPlanner* bendras vykdymo laikas buvo 1602.56 s., o naudojant tik *OptaPlanner* - 5000.66 s.

Išvados:

- Pasiūlytas generatyvinis neuroninio tinklo modelis gali sėkmingai generuoti nustatytus apribojimus atitinkančius tvarkaraščius (24 lentelė).
- Pasiūlytas modelis tvarkaraštį sugeneruoja kelis kartus greičiau, įskaitant ir apmokymo laiką, nei sprendinį randa *OptaPlanner* optimizavimo įrankis (25 lentelė).
- Uždavinio sudėtingumo didinimas ne visada įtakoja neuroninio tinklo veikimo greitį, skirtingai nei *OptaPlanner* optimizavimo įrankio (26 pav.)
- Naudojant pasirinktą neuroninio tinklo architektūrą ir nustatytus parametrus ne visi apribojimai gali būti išmokti (26 lentelė).
- Neuroniniam tinklui nepavykus sugeneruoti visus apribojimus atitinkančio tvarkaraščio, jį galima paduoti *OptaPlanner* kaip pradinį sprendinį, taip sumažint vykdymo laiką, jei būtų naudojamas tik *OptaPlanner* (27 lentelė).

Literatūra

[1] DY Sha and Cheng-Yu Hsu. A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering*, 51(4):791–808, 2006.

[2] Guohui Zhang, Xinyu Shao, Peigen Li, and Liang Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309–1318, 2009.

[3] John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

[5] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[7] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[8] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[9] Jacques Carlier and Éric Pinson. An algorithm for solving the job-shop problem. *Management science*, 35(2):164–176, 1989.

[10] Emanuel Falkenauer and S Bouffouix. A genetic algorithm for job shop. In *ICRA*, pages 824–829. Citeseer, 1991.

[11] Thomas Stützle et al. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3, pages 1560–1564, 1998.

[12] Ahmed Wasfy and F Aloul. Solving the university class scheduling problem using advanced ilp techniques. In *IEEE GCC Conference*. Citeseer, 2007.

[13] Esra Aycan and Tolga Ayav. Solving the course scheduling problem using simulated annealing. In *2009 IEEE International Advance Computing Conference*, pages 462–466. IEEE, 2009.

[14] Sk Imran Hossain, MAH Akhand, MIR Shuvo, Nazmul Siddique, and Hojjat Adeli. Optimization of university course scheduling problem using particle swarm optimization with selective search. *Expert Systems with Applications*, 127:9–24, 2019.

[15] Stefan Bunte and Natalia Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317, 2009.

[16] Giorgio Carpaneto, Mauro Dell'Amico, Matteo Fischetti, and Paolo Toth. A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19(5):531–548, 1989.

[17] Uwe Aickelin and Kathryn A Dowsland. An indirect genetic algorithm for a nurse-scheduling problem. *Computers & operations research*, 31(5):761–778, 2004.

[18] Walter J Gutjahr and Marion S Rauner. An aco algorithm for a dynamic regional nurse-scheduling problem in austria. *Computers & Operations Research*, 34(3):642–666, 2007.

[19] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks. *arXiv preprint arXiv:1708.05509*, 2017.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[21] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional

gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[22] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*, 2016.

[23] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face aging with conditional generative adversarial networks. In *2017 IEEE international conference on image processing (ICIP)*, pages 2089–2093. IEEE, 2017.

[24] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Gp-gan: Towards realistic high-resolution image blending. In *Proceedings of the 27th ACM international conference on multimedia*, pages 2487–2495, 2019.

[25] Huang Bin, Chen Weihai, Wu Xingming, and Lin Chun-Liang. High-quality face image sr using conditional generative adversarial networks. *arXiv preprint arXiv:1707.00737*, 2017.

[26] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *arXiv preprint arXiv:1610.07584*, 2016.

[27] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

[28] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

[29] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[30] Laurent Perron and Vincent Furnon. *OR-Tools*. Google.

[31] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Solver Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016.

[32] Geoffrey De Smet and open source contributors. *OptaPlanner User Guide*. Red Hat, Inc. or third-party contributors, 2006. OptaPlanner is an open source constraint solver in Java.