VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ KATEDRA

Rekurentinių neuroninių tinklų taikymas maršruto kelių tinkle planavimui

Recurrent Neural Network Models for Route Planning in Road Networks

Projektinis darbas

Atliko: 4 kurso 6 grupės studentas
Lukas Valatka (parašas)

Darbo vadovas: Linas Petkevičius (parašas)

Vilnius – 2019

TURINYS

Įvadas

Efektyvi bei optimali maršruto paieška kelių tinkle - aktuali problema. Optimalaus maršruto paieška svarbi optimizuoti viešojo transporto, logistikos, piko meto dalyvių išlaidas. Laikant kelių tinklą grafu, galima taikyti klasikinius optimalaus kelio grafe paieškos algoritmus. Kurti optimalų maršrutą dideliuose grafuose, tipiškai, naudojami euristikomis paremti algoritmai, pavyzdžiui, A*. Šis bei kiti euristiniai algoritmai užtikrina sparčią paiešką neprarandant optimalumo (galiojant tam tikroms sąlygoms euristikoms) [Hel18b].

Planuojant optimalų maršrutą realiame kelių tinkle reikia įvertinti daug kintamųjų, potencialiai galinčių sutrumpinti paiešką: krypties (atstumas, kampas iki tikslo), aplinkos: (kliūtys, oro sąlygos, eismo intensyvumas). Tai didina algoritmų sudėtingumą. Sunku nusakyti, koks turėtų būti kiekvieno iš kintamųjų svoris žengiant žingsnį optimalaus maršruto paieškoje. Taip pat, dažna būsenų sprogimo (angl. state explosion) problema [Val98]. Giliųjų neuroninių tinklų algoritmai gali padėti spręsti šias problemas. Šie algoritmai yra apmokomi rasti optimalius maršrutus. Nereikia tiesiogiai programuoti juos pagreitinančių euristikų. Peršasi išvada, jog giliųjų neuroninių tinklų algoritmai, atsparūs realizacijos klaidoms bei užtikrinantys spartesnę paiešką - tinkamas pasirinkimas optimalaus maršruto kelių tinkle radimui [Zho18].

Problema

Šis darbas siekia pagerinti Zhou [Zho18] bei Alex et al. [GWR+16] neuroninių tinklų metodus spręsti optimalaus maršrutų kūrimo kelių tinkle problemą. Zhou planavimo algoritmo trūkumas - negebėjimas priimti globaliai optimalių sprendimų. Zhou naudotas pilnai jungus neuroninis tinklas geba priimti lokaliai optimalius sprendimus, nes neturi atminties. Alex et al. tyrimai rodo, jog jų sukurtas modelis geba rasti optimalų kelią grafe su 55.3% tikslumu. Kita vertus, įvertinimas buvo atliktas naudojant sintetinį grafą, tikėtina, neatitikantį realaus kelių tinklo struktūros.

Tikslas

Atlikęs literatūros analizę, autorius įvertino, jog norima išspręsti problema, sekos modeliavimas bei spėjimas, gali būti sprendžiama pasitelkiant rekurentinius neuroninius tinklus [Lip15]. Remiantis šia informacija bei siekiant praplėsti autorių rezultatus, buvo suformuluotas tyrimo tikslas:

Skaitiškai įvertinti rekurentinių neuroninių tinklų galimybes kurti optimalius maršrutus realiame kelių tinkle.

Uždaviniai

1. Paruošti optimalių maršrutų realiame kelių tinkle duomenų rinkinį.
2. Realizuoti rekurentiniu neuroniniu tinklu grįstą optimalaus kelio grafe paieškos algoritmą (toliau - RNT)
3. Paruoštus duomenis panaudoti apmokyti rekurentinio neuroninio tinklo modelį.

4. Palyginti klasikinio optimalaus kelio paieškos grafe metodo bei RNT algoritmo galimybes naviguoti realiame kelių tinkle pagal: optimalumą laiko atžvilgiu, sėkmingo tikslo pasiekimo (toliau - atvykimo) dažnį.

1. Teorinė dalis

1.1. Optimalaus kelio paieškos grafe algoritmai

Šiame poskyryje pateikta teorinė medžiaga apie optimalaus kelio paiešką grafe pasitelkiant klasikinius trumpiausio kelio paieškos algoritmus (Dijkstra ir A*).

1.1.1. Optimalus kelias

Optimalaus kelio paieškos grafe tikslas - rasti tokį kelią tarp dviejų viršūnių, kad rasto kelio briaunų kainų suma būtų minimali [SW18].

1.1.2. Optimalaus kelio algoritmai

Optimalaus kelio paieškai grafe naudojami įvairūs algoritmai. Keletas pavyzdžių: Floyd-Warshall, Bellman-Ford, Dijkstra, A* [MAR+17]. Aptarsime Dijkstra bei A*, kurių panaudojimas bei idėjos atsispindi šiame darbe.

Dijkstra. Dijkstra algoritmas naudojamas rasti optimalų kelią grafe tarp dviejų ar daugiau viršūnių [Yan14]. Paieškos metu yra atidaromos (vertinamos) grafo viršūnės viena po kitos. Toliau laikysime, kad x - būsenų paieškos problemoje - būsena, paieškos grafe problemoje - viršūnė. Vertinimo tvarką nusako viršūnės kainos funkcija:

$$f(x) = g(x)$$

$g(x)$ - kelio nuo pradinės iki einamosios viršūnės kaina. Algoritmas visada kaip sekantį žingsnį renkasi mažiausios f reikšmės viršūnę. Naudojant prioritetinę eilę saugoti paruoštoms patikrinti viršūnėms, algoritmo sudėtingumas siekia $O(|E| + |V|^2)$, kur E - briaunų aibė, V - viršūnių aibė [Red13].

A*. A* algoritmas - tai Dijkstra algoritmo modifikacija spartesnei paieškai. Galima laikyti, jog tai informuotas Dijkstra algoritmas [HNR68]. Skirtumas: viršūnės kainos įverčio funkcija:

$$f(x) = h(x) + g(x)$$

Papildomas narys $h(x)$ - euristinė viršūnės vertė. Euristinė vertė nusako tikėtiną trumpiausio kelio nuo x viršūnės iki tikslo viršūnės kainą. Norint užtikrinti optimalumą, euristinė funkcija turi tenkinti leistinumo ir pastovumo savybes [Hel18a]. Sukonstravus tokią euristinę funkcija, A* algoritmas, tipiškai, greičiau randa tikslą nei Dijkstra algoritmas, išlaikant optimalumą. Euristinių algoritmas sudėtingumą įvertinti sudėtinga, nes jis priklauso nuo pasirinktos euristinės funkcijos [HNR68].

1.2. Neuroniniai tinklai

Šiame skyriuje bus aptarta viena populiariausių mokymosi algoritmų klasių - **dirbtiniai neuroniniai tinklai** (angl. artificial neural networks).

1.2.1. Perceptronas

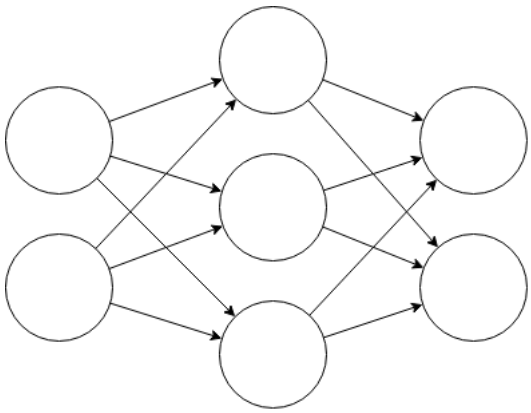
Perceptronas, išrastas Frank Rosenblatt 1957 metais, yra supaprastintas matematinis biologinio neurono modelis, aproksimuojantis dvejetainį klasifikavimą [Fei17]. Tai yra įvesties vektoriaus **x** priskyrimas **y**:

$$f(x) = \begin{cases} 1, & \text{if } \mathbf{w^T x} + \mathbf{b} \\ 0, & \text{kitu atveju} \end{cases}$$

f - **aktyvacijos funkcija** (angl. activation function), w - svoriai, realiųjų skaičių vektorius, x - pavyzdys (angl. example), realiųjų skaičių vektorius, b - polinkio (angl. bias) vektorius. Ši formulė gali būti traktuojama kaip tiesinis klasikatorius visiems įvesties vektoriams x. Problema iškyla, kai įvesties neįmanoma perskirsti tiese (pvz.: XOR funkcija) [Kaw00].

1.2.2. Daugelio lygių perceptronas

Norint aproksimuoti netiesinę funkciją, perceptrono modelis turi būti išplėstas iki **daugelio lygių perceptrono**, dar žinomo kaip **neuroninių tinklų** modelio. Šį modelį sudaro įvesties, 1 ar daugiau paslėptų (angl. hidden) bei išvesties sluoksnis. Kiekviena iš modelio grafo viršūnių atitinka vieną perceptroną. Pridėjus netiesinę aktyvacijos funkciją, teoriškai įrodyta, jog modelis geba aproksimuoti bet kokią tiesinę ar netiesinę funkciją [Goo16].



1 pav. Neuroninis tinklas su vienu paslėptu sluoksniu

1.2.3. Aktyvacijos funkcija

Aktyvacijos funkcijos pasirinkimas itin priklauso nuo neuroninio tinklo architektūros bei problemos. Rekomenduojama naudoti funkcija: ReLU [Goo16]. Norint priskirti objektą vienai ar daugiau klasių, naudojama Softmax funkcija. Gaunamas kiekvienos iš klasių tikimybinis pasiskirstymas taip, kad visų klasių tikimybių suma būtų lygi 1.

1.2.4. Tikslo funkcija

Optimizavimo metoduose, tikslo funkcija - funkcija, kurią siekiama minimizuoti. Šiuo atveju, neuronininių tinklų kontekste, tikslo funkciją atitinka **nuostolių funkcija** (angl. loss function). Jeigu neuroninis tinklas yra naudojamas prižiūrimo mokymosi (angl. supervised learning) užduočiai, spėjimo teisingumą leidžia įvertinti pastaroji funkcija. Šio tipo funkcijos naudojamos įvertinti neuroninio tinklo spėjimo nuokrypį. Nuostolių funkcija įvertina tikslo y skirtumą nuo modelio prognozės y_t.

1.2.5. Atgalinis sklidimas

Kiekvienai užduočiai korektiškai spręsti turi būti rasta tinkama neuroninio tinklo svorių konfigūracija. Procesas rasti šiai konfigūracijai yra vadinamas tinklo apmokymu. Neuroninių tinklų apmokymas paremtas nuostolių funkcijos minimizavimu pagal svorius kaip kintamuosius. Tai leidžia pasiekti atgalinis sklidimas (angl. backpropagation) [Dav89]. Kiekvieno iš n neuroninio tinklo svorių (parametrų) gerinimo žingsnis apskaičiuojamas, randant kainos funkcijos gradientą, vektorių, sudarytą iš kainos funkcijos dalinių išvestinių pagal kiekvieną iš tinklo svorių. Naudojant, pavyzdžiui, gradientinio nusileidimo optimizacijos metodą, žengiamas žingsnis nuostolių funkcijos mažėjimo kryptimi pakeičiant tinklo parametrus pagal dalinių išvestinių reikšmes konkrečiame taške (t.y. konkrečioje svorių konfiguracijoje) bei mokymosi spartos daugiklį [Rud16].

Tipiškai, rasti funkcijos gradientą yra sudėtinga. Užduotį supaprastina **grandinės taisyklės** (angl. chain rule) [Goo16], leidžianti nesudėtingai rasti sukomponuotų funkcijų gradientą. Tai - dažnai naudojamas metodas rasti kainos funkcijos gradientą, nes neuroninis tinklas yra didelė sukomponuota funkcija [Goo16].

1.2.6. Gilaus mokymosi neuroniniai tinklai

Gilaus mokymosi neuroninių tinklų (angl. deep learning neural networks) (toliau - gilieji neuroniniai tinklai) - tai terminas, skirtas apibūdinti neuroninius tinklus, turinčius daugiau nei vieną paslėptą sluoksnį. Šios architektūros pranašumas prieš įprastus neuroninius tinklus - gebėjimas išmokti spręsti sudėtingesnes užduotis t.y. aproksimuoti labiau kompleksinės funkcijas, tokias kaip grafikos klasifikacija, kalbos atpažinimas. Apie 2010 - uosius metus, skaičiavimo infrastruktūra tapo užtektinai pajėgi, kad gilieji neuroniniai tinklai galėtų būti apmokyti per sąlyginai trumpą laiką. Nuo to laiko, tai vienas populiauriausių mokymosi algoritmų metodų.

1.2.7. Rekurentiniai neuroniniai tinklai

Tam tikros problemos netenkina Markovo savybės, pavyzdžiui, laiko eilučių spėjimas, todėl metodai, prognozuojantys ateitį tik pagal dabarties būseną, pasidaro neefektyvūs. Šiai problemai spręsti tinka **rekurentiniai neuroniniai tinklai** (angl. recurrent neural networks) (trumpiau: RNT) [Goo16]. Šis modelis pasižymi tuo, jog turi atmintį. Praeities būsenos yra komponuojamos su įvestimi, kad gauti išvestį. Šie tinklai, pastaruoju metu, buvo itin sėkmingai naudojami spręsti sekų spėjimo problemoms [Lip15].

Rekurentiniai tinklai pasižymi dingstančių (angl. vanishing) arba sprogstančių (angl. exploding) gradientų problema [Raz13]. Problemai spręsti yra sukurtos modifikacijos: Trumpos-ilgos atminties (**LSTM**) [Jur97] bei rekurentiniai pertvariniai tinklai (**GRU**) [CGC+14].

2. Tyrimo eiga

Tyrimą sudarė 3 etapai. Prieš vykdant tyrimą, buvo paruošta tyrimo aplinka. Toliau - suformuota duomenų aibė apmokyti RNT algoritmo neuroninį tinklą. Sekančiu žingsniu, realizuotas RNT algoritmas. Galiausiai, RNT algoritmas buvo skaitiškai įvertintas.

Tyrimo aplinka. Tyrimas buvo vykdomas 2 - ose aplinkose. Tyrimo eiga buvo vykdyta ant Lenovo Thinkpad T480s su Intel I5 8550u procesoriumi bei Intel HD 620 vaizdo plokšte. Neuroninis tinklas buvo apmokytas Google Colab, riboto naudojimo nemokamoje PaaS tipo giliojo mokymosi platformoje. Google Colab teikia dalinę prieigą prie Tesla T80 vaizdo plokštės resursų.

2.1. Duomenų aibės paruošimas

Duomenų aibė buvo sudaryta iš OpenStreetMap atvirų žemėlapių. Pirma buvo parsisiųstas ir papildytas pasirinktos vietos gatvių tinklas. Vėliau buvo kuriami optimalūs maršrutai gatvių tinkle. Šie maršrutai, galiausiai, buvo transformuoti į pavyzdžių - etikečių formatą, taip paruošiant mokymo duomenis neuroniniu tinklu grįstam planavimo algoritmui.

2.1.1. Kelių tinklo gavimas

2 pav. Apmokymo etapo gatvių tinklo grafas. OSMnx vizualizacija.

Naudojantis OSMnx paketu buvo parsisiųstas 30 kilometrų kraštinės gatvių tinklo kvadratas. Pasirinktas centro taškas: $54.677194N, 25.268333E$ (Vilniaus miesto Naujamiesčio rajonas). Duomenų šaltinis: OpenStreetMap, internetinis atviros prieigos žemėlapių tiekėjas. Kelių tinklas tolesniam naudojimui buvo išsaugotas kaip grafas - viršūnės atitiko sankryžas, o briaunos - kelius.

2.1.2. Kelio tinklo papildymas

Autorius aptiko, jog dalis ar visi OpenStreetMap objektų (-ai) neturi neuroninio tinklo apmokymui reikiamų parametrų: maksimalaus leistino greičio (toliau - kelio greitis) bei išvestinio parametro - kelio greičiausio pervažiavimo trukmės (toliau - kelio trukmė). Kelių tinklas buvo papildytas šiais atributais.

Kelio greitis. Buvo pastebėta, jog dauguma kelių turi kelio greičio atributą, tačiau ne visi. Taip pat autorius pastebėjo, jog dauguma kelių turi kelio tipo atributą (pavyzdžiui: ”highway”, ”residential_road”). Neturintiems kelio tipo atributo, buvo priskirtas atributas ”default”. Buvo priimtas sprendimas keliams, neturintiems kelio greičio atributo, išvesti šį atributą iš kelio tipo pagal tokius atitikmenis:

1 lentelė. OpenStreetMap kelio tipo bei priskirto kelio greičio atitikmenys

Kelio tipas	Didžiausias greitis
”residential”	50
”secondary”	90
”primary”	90
”motorway”	120
”motorway_link”	120
”trunk”	110
”tertiary”	90
”default”	70

Kelio trukmė. Pridėjus kelio greitį visuose kelių tinklo keliuose, buvo pridėtas *kelio trukmės* atributas visiems kelių tinklo keliams. Šis atributas buvo apskaičiuotas padalinus kelio ilgio atributą iš kelio greičio atributo. Kiekvienas kelias turi kelio ilgio atributą, o greičio atributas kiekvienam keliui buvo apskaičiuotas prieš tai.

2.1.3. Optimalių maršrutų kūrimas

Apmokyti tinklą buvo paruošta duomenų aibė optimalių pagal laiką maršrutų kelių tinkle.

Autorius pasinaudojo OSMnx statinių paieškos algoritmų grafe API. Buvo naudojama trumpiausio kelio funkcija rasti optimalų kelią iš taško A į tašką B pagal kelio trukmės atributą Dijkstra algoritmu (A, B - konkrečios kelių tinklo viršūnės, identifikuojama skaitiniu ID).

Eksperimentui buvo sugeneruota 2000 optimalių maršrutų pagal kelio trukmę kelių tinkle (toliau - epizodai). Kiekvienam maršrutui buvo parenkamas atsitiktinis pradžios bei pabaigos taškas. Iš to seka, jog sukurtų maršrutų ilgis - kintantis, taip pat maršrutai - įvairūs.

2.1.4. Neuroninio tinklo apmokymo duomenų kūrimas

Toliau, visi epizodai buvo transformuoti į N ilgio sąrašą, kur $\forall n \in N, |n| = 64 = B$. B - pakrovimo į tinklą grupės (angl. batch) dydis (toliau - grupė). $N = E/B$, kur E - visų sukurtų epizodų skaičius.

Kiekvienas B narys sudarytas iš 2 sąrašų - pavyzdžių (angl. examples) $T \times X$ ir etikečių (angl. label) $T \times Y$ dydžio matricos, kur T - epizodo laiko žingsnių skaičius, dydis, lygus epizodo ilgiui.

Kadangi tiek apmokymo, tiek testavimo kelių tinkle sąnkryžos turėjo daugiausiai 5 išvažiavimus, buvo laikoma, jog didžiausia X vertė - 20, o Y - 5.

Pavyzdžiai ir etiketės. Pavyzdys - konkretaus epizodo žingsnio (viršūnės) kelių tinkle kaimyninių viršūnių parametrų sąrašas. Kiekvienai kaimyninei viršūnei buvo išskirta F sąrašo vietų (laikykime, kad F_k - subsąrašas kiekvienam kaimynui su indeksu k, atitinkančiu subsąrašo poziciją pavyzdyje), kur $\forall f \in F$ - parametras iš 6 lentelės. Etiketė - sekančios viršūnės pasirinkimas, subsąrašo F_k k indeksas pavyzdyje. Etiketės sąrašas - 0 ir 1 seka. Vienintelis 1 - etas sekoje įrašytas taip, kad jo indeksas sekoje atitiktų indeksą k.

2 lentelė. Pavyzdžio parametrai

Parametras	Paaiškinimas
Kaimyno kampas iki tikslo	Kampas laipsniais tarp kaimyno ir tikslo viršūnės.
Ar kelias vienos krypties iki kaimyno?	Boolean tipo reikšmė, nusakanti, ar kelias iki kaimyno yra vienos krypties.
Trukmė iki kaimyno (sekundėmis)	Kelionės trukmė iki kaimyno didžiausiu leidžiamu greičiu.
Kaimyno nuotolis nuo tikslo	Vincenty atstumas nuo kaimyno iki tikslo viršūnės.

Tarkime, turime epizodo žingsnio su 2 kaimynais pavyzdį. Tai atrodytų taip:

$$x = [45, 0, 54, 2500, 40, 1, 10, 100]$$

Šiuo atveju, tinklas rinksis tarp dviejų sekančių viršūnių. Pirmoji - 45, 2500 m nuo tikslo viršūnės bei 54 s nevienakrypčiu keliu nutolusi nuo einamosios viršūnės (0 indeksas). Antroji - 40, 100 m nuo tikslo viršūnės ir 10 s vienakrypčiu keliu nuo einamosios viršūnės nutolusi viršūnė (1 indeksas). Tarkime, tinklas pasirenka 1 viršūnę kaip sekančią. Tinklo sugeneruota etiketė atrodytų taip:

$$y_{spejimas} = [0, 1]$$

Apmokymo, validavimo ir testavimo duomenų aibių suformavimas. Pagrindinė duomenų aibė buvo padalinta, pirma, į dvi dalis: mokymosi ir testavimo (santykiu $80/20$). Mokymosi rinkinys buvo padalintas į dar dvi dalis: mokymosi ir validavimo ($80/20$). Kiekvienos epochos pabaigoje tinklas buvo įvertinamas su validavimo duomenimis. Po visų epochų, buvo atliktas įvertinimas su testavimo duomenimis.

Duomenų dauginimas. Viena iš pagrindinių Dijkstra algoritmo lemų yra, jog optimalaus kelio grafe subkeliai yra taip pat optimalūs keliai [Yan14]. Ši idėja buvo išnaudota kiekvieną E epizodą paversti į $E^{'}$ epizodų, kur $E^{'}$ dydis:

$$|E^{'}| = \sum_{n=1}^{|E|} |E| - (n-1)$$

Galiausiai, kiekvienos iš apmokymo, validavimo ir testavimo duomenų aibių poaibių epizodų ilgiai buvo suvienodinti grupėmis: visi epizodai buvo išrikiuoti pagal ilgį, toliau buvo imamos B dydžio grupės, surandamas ilgiausias epizodas grupėje ir visiems grupės epizodams pridedami trūkstami laiko žingsniai. Pavyzdžiai buvo užpildyti 20 ilgio vektoriais su 0, etiketės - 5 ilgio vektoriais su 0.

Atlikus duomenų dauginimą, buvo gauta 102795 epizodų. Kaip paskutinis žingsnis, apmokymo, validavimo ir testavimo (atskirai) B dydžio poaibiai buvo permaišyti.

2.2. RNT algoritmo realizavimas

Algoritmas buvo realizuotas dviem etapais. Pirma buvo apibrėžta neuroninio tinklo, svarbiausia algoritmo komponento, architektūra, remiantis tyrėjų rekomendacijomis. Toliau, buvo sukurtas algoritmas su dviem komponentais - žingsnio spėjimo komponentų (rekurentinis neuroninis tinklas) bei tyrinėjimo (angl. exploration) komponentu pagerinti algoritmo efektyvumą.

2.2.1. Neuroninio tinklo architektūra

3 lentelė. Tinklo architektūra

Įvesties sluoksnis (20)
16 vienetų LSTM. Išmestis (angl. dropout) (0.5). Rekurentinė išmestis (0.5)
Tanh aktyvacija
16 vienetų LSTM. Išmestis (0.4). Rekurentinė išmestis (0.4)
Tanh aktyvacija
Paskirstytas laike pilnai jungus sluoksnis
Softmax Išvestis (5)

Buvo pasirinktas LSTM rekurentinių neuroninių tinklų modelis, nes, tipiškai, apmokymo bei spėjimo epizodai ilgi, virš 50 žingsnių [Jur97]. Vienetų dydis atitinka RNT paslėptos būsenos dimensiją. Nėra formalios teorijos, nusakančios, koks turėtų būti paslėptos būsenos dydis. Autorius pasirinko tokį skaičių, kad jis būtų tarp didžiausio įvesties ir didžiausio išvesties dimensijos dydžio, 16. Bandymų metu buvo įvertinta, jog šis skaičius sukuria panašaus lygio tikslumą kaip dažniausiai naudojami skaičiai, pavyzdžiui, 128 bei padidina mokymosi spartą. Išmestis pasirinkta 0.5 ir 0.4 pagal literatūros rekomendacijas [CPP+17]. Rekurentinė išmesties efektyviausia vertė šiai autoriaus sprendžiamai problemai nustatyta derinimo metu (angl. fine-tunning). Tanh - standartinė LSTM aktyvacijos funkcija Keras bibliotekoje, todėl ši vertė nebuvo keista. Softmax aktyvacija pasirinkta, nes tinklo išvestis - ribotos apibrėžties aibės indeksai - kategorijos.

2.2.2. Neuroninio tinklo apmokymas

Tinklas buvo apmokytas viso per 10 epochų. Kiekvienos epochos metu buvo apmokoma $B = 64$ dydžio partijomis. Modelis buvo apmokytas (angl. trained) su 82236 įvairaus ilgio epizodų. Modelis buvo validuotas (toliau - validavimo aibė) bei ištestuotas (toliau - testavimo aibė) su 20559 įvairaus ilgio epizodų. Mokymo metu, kiekvienos epochos pradžioje, duomenų aibės B dydžio poaibiai buvo atsitiktinai permaišomi.

2.2.3. RNT algoritmas

Algoritmas 1 RNT algoritmas

procedūra GeneruotiTrumpiausiąKelią($g, m, s, g, stabdymoSalyga(c, p)$)
  $c \leftarrow s$ ▷ Nustatome einamąją viršūnę
  $p \leftarrow \varnothing$ ▷ Kelio viršūnių (tuščias) sąrašas
  while $c \neq g$ **do** ▷ Baigiame, kai pasiekiame tikslą
    if $stabdymoSalyga(c, p)$ **then**
      return p
    end if
    $x \leftarrow generuotiVirsunesAplinka(c, g)$
    $c \leftarrow m.spejimas(x)$
    $r \leftarrow$ atsitiktinis true/false, $(0.98, 0.02)$ tikimybės.
    if r **then**
      $c \leftarrow m.spejimas(g, a)$
    else
      $c \leftarrow$ atsitiktinis c kaimynas (viršūnė) grafe g
    end if
    $p \leftarrow p + c$
  end while
  return p
end procedūra

2.3. RNT algoritmo įvertinimas

Šiame etape buvo įvertintas RNT algoritmas mokymosi duomenų neapimančioje situacijoje.

2.3.1. Įvestis

Autorius testavimui sudarė $1, 2, 4, 6, 8, 10$ km gatvių tinklų grafų G_{visi} aibę su centro tašku $55.716167N, 21.152694E$ (Klaipėdos miesto šiaurinė dalis). Lyginant su apmokymo etapu, pasirinkta skirtinga duomenų aibė patikrinti RNT algoritmo neuroninio tinklo gebėjimą generalizuotis [Goo16]. Buvo sugeneruota $|A| = 500$ aibė porų atsitiktinai kiekvienam iš G_{visi} grafų parinktų viršūnių. Laikant, jog pirmasis kiekvienos iš aibių elementas S - pradžios taškas, o antrasis G

- pabaigos taškas, kiekvienai porai buvo sukurta aibė $S- > G$ optimalių pagal kelionės trukmę maršrutų.

Optimalumui įvertinti autorius atmetė pirmą elementą G_{visi} (1 km tinklą), pastebėjęs, jog optimalumo vertinimui skirtumas tarp $G_{visi}(0)$ ir $G_{visi}(1)$ minimalus, o atvykimo dažniui vertinti skirtumas nemažas.

2.3.2. Optimalumas

Kiekvienai porai iš einamojo vertinamo gatvių tinklo buvo kuriamas trumpiausias maršrutas pagal 1 algoritmą. Kiekvienas 1 algoritmo sukurtų maršrutų buvo santykiu bei nuokrypiu pagal kelionės trukmę palygintas su optimalaus maršruto analogu. Stabdymo sąlyga: maršrutas viršija optimalaus kelio einamajai taškų porai ilgį 150%. Įvykus stabdymui, pora nebuvo įtraukta į optimalumo vertinimą. Galutinis vertinimo etapo rezultatas - visų santykinų ir absoliučių palyginimų vidurkis (atskirai) einamajam kelių tinklui.

2.3.3. Atvykimo dažnis

Kiekvienai porai iš einamojo vertinamo gatvių tinklo buvo kuriamas trumpiausias maršrutas pagal 1 algoritmą. Stabdymo sąlyga: maršrutas viršija optimalaus kelio einamajai taškų porai ilgį 150%. Įvykus stabdymui, buvo laikoma, jog nebuvo atvykta į tikslą. Galiausiai, visų be stabdymo įvykusių eksperimentų skaičius buvo padalintas iš visų eksperimentų skaičiaus.

Rezultatai

Neuroninio tinklo apmokymas

Tyrimų metų buvo realizuotas RNT modelis. Šis modelis buvo apmokytas (angl. trained) su 82236 įvairaus ilgio epizodų 10 epochų. Modelis buvo validuotas su 20559 bei ištestuotas su 20559 įvairaus ilgio epizodų (plačiau žr. 2.2.2). Pastarųjų 3 aibių sankirta - tuščia aibė.

4 lentelė. RNT algoritmo rekurentinio neuroninio tinklo apmokymo rezultatai.

Rodiklis	Vertė
Apmokymo laikas	535s
Validavimo galutinės epochos tikslumas	86.49%
Testavimo tikslumas	86.59%

Apmokymo laikas nurodo modelio apmokymo trukmę. **Validavimo galutinės epochos tikslumas** - modelio validavimo duomenų aibės korektiškai prognozuotų etikečių dalis. **Testavimo tikslumas** - modelio testavimo aibės korektiškai prognozuotų etikečių dalis.

Pasiektas aukštesnis testavimo tikslumas (86.59%) už validavimo galutinės epochos tikslumą (86.49%) rodo, jog neuroninis tinklas ne įsiminė (angl. overfit) apmokymo duomenis, bet sėkmingai įsisavino duomenis apibrėžiančią funkciją - generalizavosi [Goo16]. Gauta neuroninio tinklo konfigūracija buvo užfiksuota, ir toliau pereita prie RNT algoritmo vertinimo dalies.

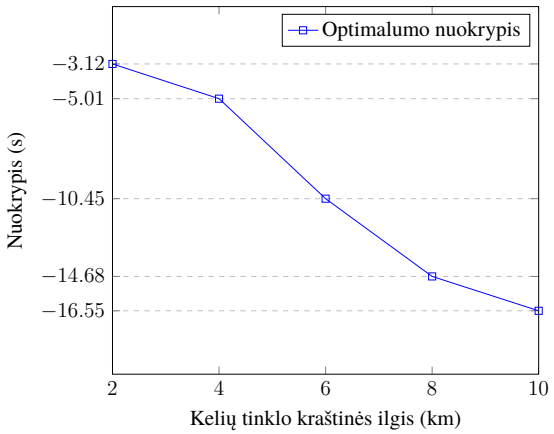
RNT algoritmo įvertinimas

Šiame poskyryje pateikti tyrimų metų realizuoto RNT algoritmo galimybės kurti optimalius maršrutus pagal trukmę kelių tinkle įvertinimo rezultatai. Buvo įvertintas vidutinis optimalumas nuokrypio (-s) bei santykio (%) forma, taip pat ir atvykimo dažnis.

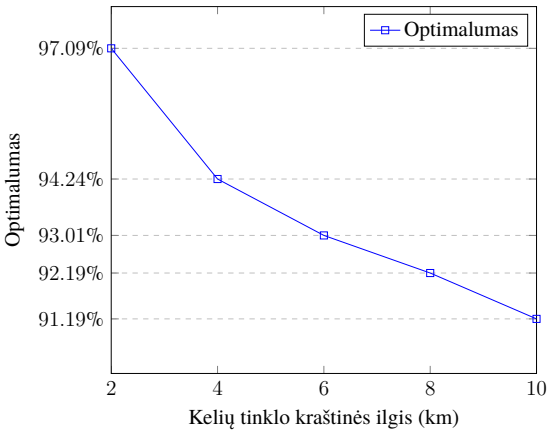
RNT algoritmo vidutinio optimalumo įvertinimas

Grafikuose 3, 4 bei lentelėje 5 pateikti vidutinio RNT algoritmo gebėjimo kurti optimalius maršrutus įverčiai. Kiekvienas iš 3 grafiko bei 4 grafiko taškų atitinka 500 RNT algoritmų sukurtų maršrutų optimalumo įverčių vidurkį konkretaus dydžio kelių tinkle (plačiau apie duomenų įvestį žr. 2.3.2).

3 - iam grafike taškai vaizduoja algoritmo sugeneruotų maršrutų vidutinį trukmės nuokrypį sekundėmis nuo atitinkamų optimalių maršrutų trukmės. 4 grafike taškai parodo algoritmo sugeneruotų maršrutų vidutinį trukmės santykį su optimalių atitinkamų maršrutų trukme. Tiek iš 3, tiek iš 4 grafiko matyti, jog RNT algoritmo maršrutų trukmės optimalumo nuokrypio ir santykio vidurkis tiesiškai krenta tiesiškai augant kelių tinklo pločiui. Tiek 3, tiek 4 grafiko rezultatus galima paaiškinti taip: RNT algoritmas, kiekvieną kartą priimdamas sekančio žingsnio sprendimą, gali priimti suboptimalų sprendimą p. Augant kelių tinklui, ilgėja vidutinis maršrutas, auga algoritmo prognozių skaičius, tuo pačiu auga suminis suboptimalių sprendimų skaičius, kurių rezultatas, galiausiai - suboptimalus maršrutas (kelias grafe).



3 pav. RNT algoritmo vidutinis suplanuoto maršruto trukmės skirtumas nuo optimalios maršruto trukmės. 0 - vidutiniškai maršrutai optimalūs pagal trukmę. Mažesnis nuokrypis atitinka prastesnį optimalumą.



4 pav. RNT algoritmo vidutinis suplanuoto maršruto trukmės santykis su optimalia maršruto trukme. 100% - vidutiniškai maršrutai optimalūs pagal trukmę.

5 lentelė. RNT algoritmo sukurto maršruto vidutinis laiko nuokrypis nuo optimalaus laiko bei santykis su optimalaus laikiu skirtingo dydžio kelių tinkluose

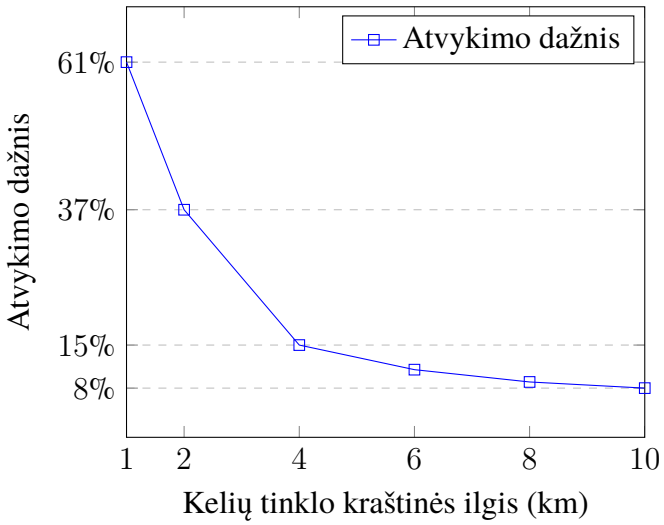
Kelių tinklo kraštinės ilgis (km)	Laiko nuokrypis (s)	Optimalumas (%)
2	−3.12	97.09
4	−5.01	94.24
6	−10.45	93.01
8	−14.68	92.19
10	−16.55	91.19

RNT algoritmo vidutinio atvykimo dažnio įvertinimas

Kiekvienas iš 5 grafiko taškų atitinka 500 RNT algoritmo sukurtų maršrutų sėkmingų atvykimų (tikslo pasiekimų) skaičiaus vidurkį konkretaus dydžio kelių tinkle (plačiau apie eksperimento detales žr. 2.3.3).

5 grafikas logaratminės skalės. Tik nedideliame 1 km pločio kelių tinkle daugiau nei pusė RNT algoritmo kelionių pasiekia tikslą. Kelių tinklui tiesiškai augant, atvykimo dažnis logaritmiškai mažėja, kol mažėjimas sulėtėja ties 10%.

Kaip ir optimalumo eksperimentų atveju, šiuos rezultatus galima paaiškinti tuo, jog, augant kelių tinklui, auga suminis suboptimalių sprendimus skaičius, didinantis bendrą nuokrypį.



5 pav. Grafikas vaizduoja RNT algoritmo vidutinį atvykimo dažnį skirtingo dydžio kelių tinkluose

6 lentelė. RNT algoritmo vidutinis atvykimo dažnis skirtingo dydžio kelių tinkluose

Kelių tinklo kraštinės ilgis (km)	Atvykimo dažnis (%)
1	61
2	37
4	15
6	11
8	9
10	8

Išvados

Remiantis rezultatais, matyti, jog rekurentinių tinklų architektūra, realiame kelių tinkle, tinkamas modelis optimalaus maršruto paieškai. Kita vertus, autoriaus pasiūlytas RNT grįstas algoritmas stokoja robastiškumo.

RNT algoritmas geba kurti globaliai optimalius maršrutus. Lyginant su Zhou pilnai jungių neuroninių tinklų architektūra paremtu algoritmu [Zho18], RNT algoritmas geba kurti vidutiniškai 5% optimalesnius maršrutus, taip pat tai geba pasiekti realiame, ne dirbtiniame kelių tinkle. Augant kelių tinklo dydžiui, nuokrypis tiesiškai auga. Kita vertus, to buvo tikėtąsi: ilgėjant epizodams, RNT algoritmas, sumoje, priima vis daugiau ir daugiau navigacijos sprendimų su nedideliu nuokrypiu.

Problema, jog autoriaus pasiūlytas RNT algoritmas dažnai patenka į ciklus. Tyrinėjimo žingsnis buvo pasiūlytas kaip išeitis spręsti problemai [1 algoritmas], bet, iš 5 grafiko, tai nepadėjo išspręsti problemos. Zhou pasiūlytas algoritmas robastiškesnis - atvykimo lygis siekia, vidutiniškai, 97% [Zho18]. Šiame darbe aprašyto algoritmo atvykimo dažnis, geriausiu atveju, siekia vos 61% [5 diagrama]. Kita vertus, Zhou algoritmas buvo įvertintas ant dirbtinio kelių tinklo.

Ateityje, RNT algoritmas galėtų būti pagerintas įvairiais būdais. Pagrindinis rodiklis, kurį vertėtų gerinti - atvykimo dažnis. Autoriaus nuomonę, būtų vertinga pabandyti įtraukti klasikinių paieškos algoritmų, tokių kaip A*, technikas į RNT algoritmą ir tokiu būdu užtikrinti robastiškumą [KTG+18]. Rekurentinių neuroninių tinklų dalį būtų galima naudoti kaip euristiką. Taip pat, būtų vertinga apsibrėžti nuostolių funkciją su baudos už pasikartojančių viršūnių tyrinėjimą komponentu bei pakartotinai apmokyti RNT modelį su ciklišKais epizodais. Optimalumo įverčiams gerinti būtų galima sutrumpinti maršrutą pašalinant pakartotinai aplankytas viršūnes, panašiai kaip klasikiniuose paieškos algoritmuose, tokiuose kaip Dijsktra, galutinis kelias visada yra sudarytas iš skirtingų viršūnių [Yan14].

Šaltiniai

[CGC+14] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho ir Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

[CPP+17] Gaofeng Cheng, Vijayaditya Peddinti, Daniel Povey, Vimal Manohar, Sanjeev Khudanpur ir Yonghong Yan. An exploration of dropout with lstms. *INTERSPEECH*, 2017.

[Dav89] Ronald J. Williams David E. Rumelhart Geoffrey E. Hinton. Learning representations by back-propagating errors. *nature*:323–523, 1989.

[Fei17] Serena Yeung Fei-Fei Li Justin Johnson. Lecture 5: convolutional neural networks. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf, 2017.

[Goo16] Courville A. Goodfellow I Bengio Y. *Deep Learning*. MIT Press, 2016.

[GWR+16] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley ir k.t. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

[Hel18a] Malte Helmert. State-space search: analysis of heuristics. https://ai.dmi.unibas.ch/_files/teaching/fs18/ai/slides/ai14.pdf, 2018.

[Hel18b] Malte Helmert. State-space search: greedy bfs, a∗, weighted a∗. https://ai.dmi.unibas.ch/_files/teaching/fs18/ai/slides/ai16.pdf, 2018.

[HNR68] P. E. Hart, N. J. Nilsson ir B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968-07. ISSN: 0536-1567. DOI: 10.1109/TSSC.1968.300136.

[Yan14] Melissa Yan. Dijkstra's algorithm. http://www-math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf, 2014.

[Jur97] Sepp Hochreiter Jurgen Schmidhuber. Long short-term memory, 1997.

[Kaw00] Kiyoshi Kawaguchi. Linear separability and the xor problem, 2000.

[KTG+18] Ariel Keselman, Sergey Ten, Adham Ghazali ir Majed Jubeh. Reinforcement learning with a* and a deep heuristic. *CoRR*, abs/1811.07745, 2018. arXiv: 1811.07745. URL: http://arxiv.org/abs/1811.07745.

[Lip15] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015. arXiv: 1506.00019. URL: http://arxiv.org/abs/1506.00019.

[MAR+17] Amgad Madkour, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman ir Saleh M. Basalamah. A survey of shortest-path algorithms. *CoRR*, abs/1705.02044, 2017. arXiv: 1705.02044. URL: http://arxiv.org/abs/1705.02044.

[Raz13] Yoshua Bengio Razvan Pascanu Tomas Mikolov. On the difficulty of training recurrent neural networks, 2013.

[Red13] Harika Reddy. Path finding - dijkstra's and a* algorithm's. http://cs.indstate.edu/hgopireddy/algor.pdf, 2013.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747:2–2, 2016. arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[SW18] Robert Sedgewick and Kevin Wayne. Shortest paths. https://algs4.cs.princeton.edu/44sp, 2018.

[Val98] Antti Valmari. *The state explosion problem*. *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*. Wolfgang Reisig ir Grzegorz Rozenberg, redaktoriai. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, p. 429–528. ISBN: 978-3-540-49442-3. DOI: 10.1007/3-540-65306-6_21. URL: https://doi.org/10.1007/3-540-65306-6_21.

[Zho18] Tianyu Zhou. *Deep Learning Models for Route Planning in Road Networks*. Disertacija, 2018-06. DOI: 10.13140/RG.2.2.30527.56486.