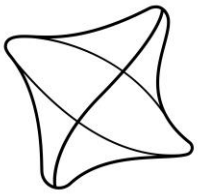


ŽILINSKÁ UNIVERZITA
V ŽILINE

Fakulta riadenia a informatiky

Časová analýza skreslenia spôsobeného grafovými neurónovými sieťami pre úlohy učenia posilňovaním

Diplomová práca

Bc. Miloš Murín.

Študijný program: Inteligentné informačné systémy

Študijný odbor: Informatika

Školiace pracovisko: Žilinská univerzita v Žiline,

Vedúci diplomovej práce: Ing. Peter Tarábek, PhD.

Žilina 2025

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY DIPLOMOVEJ PRÁCE.

Študijný program: Inteligentné informačné systémy

Meno a priezvisko
Miloš Murín

Osobné číslo
562443

Názov práce v slovenskom aj anglickom jazyku

Časová analýza skreslenia spôsobeného grafovými neurónovými sieťami pre úlohy učenia posilňovaním

Temporal Analysis of Bias Induced by Graph Neural Networks for Reinforcement Learning Tasks

Zadanie úlohy, ciele, pokyny pre vypracovanie (Ak je málo miesta, použite opačnú stranu)

Cieľ diplomovej práce:
Cieľom tejto diplomovej práce je preskúmať, ako skreslenie výberu vrcholov, ktoré je spôsobené koreláciou medzi stupňami vrcholov a hodnotami logitov v grafových neurónových sieťach (ďalej EDB), ovplyvňuje proces učenia posilňovaním. Konkrétne sa práca zameriava na analýzu toho, ako EDB ovplyvňuje exploráciu uzlov s rôznymi stupňami počas učenia modelu. Práca prostredníctvom časovej analýzy preskúma, či a ako toto skreslenie vplýva na učenie sa agentov. Súčasťou práce je výber vhodných úloh učenia sa posilňovaním pre skúmanie daného efektu. Predpokladá sa výber jednej exaktnej úlohy (ako napr. hľadanie kostry v grafe alebo hľadanie najkratšej cesty) a výber jednej NP ťažkej kombinatoricko optimalizačnej úlohy (napr. úloha obchodného cestujúceho). Dané úlohy bude potrebné transformovať do podoby vhodnej pre učenie posilňovaním a implementovať pre ne simulátor.

Obsah:

1. Analýza súčasného stavu s dôrazom na učenie posilňovaním a grafové neurónové siete. Oboznámenie sa s problematikou vrcholového skreslenia v grafových neurónových sieťach.
2. Výber vhodných úloh a ich formulácia pre učenie posilňovaním.
3. Návrh postupu testovania. Návrh vhodných metrík a vizualizácii.
4. Implementácia prostredí.
5. Trénovanie agentov a skúmanie vplyvu daného skreslenia na tréning.
6. Vyhodnotenie výsledkov a diskusia.

Meno a pracovisko vedúceho DP: Ing. Peter Tarábek, PhD., KMMOA, ŽU
Meno a pracovisko tútora DP:

garant štud. programu
(dátum a podpis)

Zadanie zaregistrované dňa 31. 10. 2024 pod číslom 3310/2024 podpis _____

Čestné vyhlásenie

Vyhlasujem, že som zadanú diplomovú prácu vypracoval samostatne, pod odborným vedením vedúceho práce/školiteľa a používal som len literatúru uvedenú v práci.

Žilina 29. apríla 2025

podpis

Poďakovanie

Chcel by som sa poďakovať môjmu školiteľovi Ing. Petrovi Tarábkovi, PhD. za jeho pripomienky, pomoc a odborné vedenie pri vypracovávaní diplomovej práce.

Abstrakt

MURÍN, Miloš: Časová analýza skreslenia spôsobeného grafovými neurónovými sieťami pre úlohy učenia posilňovaním. [Diplomová práca] / Miloš Murín. – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra matematických metód a operačnej analýzy. – Školiteľ: Ing. Peter Tarábek, PhD. Žilina: FRI UNIZA, 2025.

Práca sa venuje dokázanému skresleniu výberu vrcholov v grafových neurónových sieťach, ktoré je spôsobené koreláciou medzi stupňami vrcholov a hodnotami logitov (EDB). Analyzuje sa vplyv EDB na priebeh tréningu a výsledky agentov pri učení posilňovaním. V práci sa pokúšame natrénovať agentov na hľadanie najlacnejšej kostry v grafe. Vykonali sme niekoľko experimentov, pri ktorých sme sa snažili zmierniť vplyv EDB a popisujeme, že EDB má vplyv na tréning agentov učenia posilňovaním, ale nie je jediným problémom grafových neurónových sietí pri riešení úlohy hľadania najlacnejšej kostry a iných úloh, ktorých riešenie je založené na zameraní sa na jednu hlavnú informáciu práve na jednom mieste v grafe.

Kľúčové slová: učenie posilňovaním, grafové neurónové siete, Exploration Degree Bias, EDB, najlacnejšia kostra grafu.

Abstract

This thesis is dedicated to a confirmed bias with node selection in graph neural networks, which is caused by the correlation between the degree of a node and the logit values in the network (EDB). We analyze the influence of EDB on the training process and the achieved results of trained agents in reinforcement learning. In the thesis we are trying to train agents to search for the minimal tree of a simple graph. After our experiments attempting to lower the influence of EDB we concluded that EDB is influential in the training process of agents using reinforcement learning. However, it is not the only limitation of graph neural networks with solving the problem of finding the minimal tree of a graph and other problems that require the networks to make decisions based on one main information on exactly one place in the graph.

Keywords: Reinforcement learning, graph neural networks, Exploration Degree Bias, EDB, minimal tree of a graph.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

Skratka	Anglický význam	Slovenský význam
EDB	Exploration Degree Bias	Skreslenie stupňov pri objavovaní
RL	Reinforcement Learning	Učenie posilňovaním
GNN	Graph Neural Network	Grafové neurónové siete
lr	Learning rate	Rýchlosť učenia
PPO	Proximal Policy Optimization	Optimalizácia blízkych stratégií
GAT	Graph Attention network	Grafová neurónová sieť s parametrami pozornosti
FRI	Faculty of Management Sciences and Informatics	Fakulta Riadenia a Informatiky

ÚVOD

Učenie posilňovaním dosahuje v mnohých oblastiach výsledky na špičkovej úrovni. Najčastejšie sa pri riešení úloh využíva reprezentácia pomocou vektorov a matíc, ale niektoré úlohy je vhodnejšie reprezentovať formou grafu. Na lepšiu prácu s grafmi a grafovými úlohami boli vytvorené grafové neurónové siete. Tie pracujú priamo s grafom úlohy a preto by mali vedieť lepšie zachytiť aj priestorové informácie.

Grafové neurónové siete sa oproti bežným konvolučným sieťam líšia v jednom zásadnom aspekte, ktorý ovplyvňuje ich výstup – ide o spôsob agregácie informácie z okolia. Kým konvolučné siete pracujú s pevne definovanou veľkosťou okolia, pri grafových neurónových sieťach môže byť veľkosť okolia variabilná v závislosti od stupňa vrcholu.

Tento spôsob agregácie umožňuje grafovým neurónovým sieťam flexibilne pracovať s dátami, kde je variabilita veľkosti okolia prirodzená. Táto schopnosť robí s grafových neurónových sietí silný nástroj pre úlohy, kde sa tradičné siete ťažko uplatňujú.

Na druhej strane však táto flexibilita môže viesť k skresleniu, kedy rôzne veľkosti okolia môžu pri agregácii viesť k preferencii vrcholov s väčším (resp. menším) stupňom vrcholu a zanedbaniu vrcholov s priemerným stupňom. Takéto skreslenie môže spôsobiť problémy počas trénovania – napríklad predĺžiť konvergenciu alebo spôsobiť nestabilitu riešenia.

Pri učení s učiteľom (angl.: supervised learning) už bolo navrhnutých niekoľko spôsobov na zníženie vplyvu tohto skreslenia, ale vplyv skreslenia pri učení posilňovaním je stále málo preskúmané. Preto sme sa rozhodli prácu venovať preskúmavaniu spomínaného vplyvu pri použití učenia posilňovaním.

V úvode práce (Kapitola 1) si zadefinujeme niekoľko cieľov, ktoré by sme chceli dosiahnuť v práci a popíšeme, aký prínos má práca na analýzu riešeného skreslenia.

Následne v druhej kapitole popíšeme potrebnú teóriu používanú v práci. Zadefinujeme si dôležité pojmy potrebné na zrozumiteľné popísanie riešeného skreslenia. Nakoniec popíšeme samotný problém skreslenia v grafových neurónových sieťach.-

V tretej kapitole zanalyzujeme súčasný stav riešenia, pozrieme sa na doposiaľ dosiahnuté výsledky pomocou učenia posilňovaním, grafových neurónových sietí, prípadne ich kombináciou. Na záver kapitoly si uvedieme aj niekoľko článkov, ktoré sa venovali problému skreslenia.

Štvrtá kapitola obsahuje informácie ohľadom nášho návrhu riešenia. Popíšeme navrhovaný postup pri tréningu, spôsob testovania natrénovaných agentov a spôsob analýzy vplyvu skreslenia. Taktiež popíšeme úlohy vybrané na analýzu, spôsob ich reprezentácie, udeľovania odmien a prípadné problémy, ktoré je potrebné prekonať.

Konečnú implementáciu agenta a prostredí popíšeme v piatej kapitole. Uvedieme používané knižnice a popíšeme implementačné detaily riešenia. Taktiež popíšeme aj parametre prostredia, agenta a tréningu, ktoré budú upravované počas experimentov.

V šiestej kapitole budeme popisovať experimenty, ktoré boli vykonané a ich výsledky. Popíšeme, čo sme chceli dosiahnuť vykonanými experimentami, dôležité zmeny vykonané pri danom experimente a vyhodnotíme zistenia z výsledkov experimentov.

Posledná, siedma, kapitola poskytuje zhodnotenie výsledku samotnej práce a návrhy na budúce smerovanie výskumu v oblasti grafových neurónových sietí a ich skreslenia. Popíšeme výsledky experimentov ako celok a vyzdvihneme najdôležitejšie a najzaujímavejšie výsledky.

1 CIELE A MOTIVÁCIA

Práca nadväzuje na článok „*Exploration Degree Bias: The Hidden Influence of Node Degree in Graph Neural Network-Based Reinforcement Learning*“ [1]. V článku bolo popísané skreslenie v grafových neurónových sieťach pri použití učenia posilňovaním (Exploration Degree Bias - EDB), ktoré hovorí, že grafové neurónové siete majú tendenciu vyberať vrcholy s vyšším stupňom alebo nižším stupňom (viac v kapitole 2.4.1). Keďže spomínaná práca neanalyzuje vývoj a vplyv skreslenia počas tréningu agentov, myslíme si, že popísané skreslenie a jeho vplyv na učenie posilňovaním je možné ďalej skúmať. Preto by sme chceli zistiť aký vplyv má skreslenie na proces učenia posilňovaním a overiť, či sú grafové neurónové siete schopné riešiť vybrané úlohy.

Taktiež bol motiváciou aj prieskum zameraný na riešenie kombinatorických optimalizačných úloh [2], ktorý naznačuje, že riešenie týchto úloh pomocou učenia posilňovaním by mohlo dosahovať lepšie výsledky ako pri použití bežných heuristík. Tieto úlohy sú častokrát reprezentované pomocou grafov a teda pri ich riešení môžeme skúmať aj skreslenie. Chceli by sme vyskúšať, či sa dané úlohy dajú riešiť aj pomocou učenia posilňovaním a grafových neurónových sietí.

Pod časovou analýzou sa myslí analýza priebehu tréningu –

Pred vytvorením tejto práce sme si stanovili niekoľko cieľov, ktoré by mali viesť návrh riešenia problému a aj samotné experimenty.

Hlavným cieľom práce je analyzovať ako vplýva skreslenie v grafových neurónových sieťach, spôsobené koreláciou medzi stupňami vrcholov a hodnotami logitov v sieti, na proces učenia posilňovaním. Chceli by sme sa pokúsiť natrénovať agentov pomocou učenia posilňovaním, ktorý by boli schopný riešiť vybrané úlohy a následne zanalyzovať ako veľmi vplýva EDB na výsledky, na proces tréningu a či tento vplyv vieme zmierniť.

Popri hlavnom cieli sme určili aj vedľajší cieľ, ktorý by sme mohli dosiahnuť plnením hlavného cieľa. Chceme otestovať, či sú grafové neurónové siete vhodné na riešenie vybraných grafových úloh a pri úspešných riešeniach analyzovať spôsob konštrukcie riešenia.

Je dôležité podotknúť, že skúmame veľmi málo preskúmanú oblasť v odvetví učenia posilňovaním, preto niektoré stanovené ciele nemusia byť uskutočniteľné.

2 POPIS PROBLÉMU

V tejto kapitole popíšeme problém, ktorý budeme v práci skúmať (EDB) a taktiež použitú teóriu učenia posilňovaním, grafových štruktúr, grafových neurónových sietí a optimalizačných úloh.

2.1 Učenie posilňovaním

Učenie posilňovaním, angl.: Reinforcement Learning (RL) je odvetvie strojového učenia využívané hlavne v robotike, riadiacich systémoch, hernej umelej inteligencii alebo v úlohách, kde je ťažké vopred určiť najlepší postup. Hlavnou myšlienkou RL je napodobniť učenie sa nás, ľudí. Keď sa novorodenec pohybuje, obzerá okolie, interaguje s prostredím okolo neho, zbiera skúsenosti a z nich sa učí následky svojich akcií a čo robiť keď chce dosiahnuť nejaký cieľ.

Učenie posilňovaním sa líši od obidvoch najznámejších častí strojového učenia, učenia pod dohľadom a učenia bez dohľadu. Na rozdiel od učenia pod dohľadom, angl.: Supervised Learning, pri učení s posilňovaním nemáme označkovaný súbor dát so správnymi odpoveďami. A na rozdiel od učenia bez dohľadu, angl.: Unsupervised Learning, kde je cieľom odhaliť skrytú štruktúru v neoznačených dátach, sa pri RL snažíme maximalizovať signál odmien, pri čom hľadanie štruktúry, môže ale nemusí byť pomocou.

Učenie posilňovaním popíšeme na príklade šachového hráča, ktorý sa na základe informácii z rozloženia figúrok má rozhodnúť, akú akciu vykoná. Rozhodnutie vykonáva na základe aktuálneho stavu, minulých skúseností a možných akcií súpera.

Agent, alebo trénovaný subjekt (v našom príklade hráč), je časť prostredia alebo pozorovateľ, ktorý svojimi akciami dokáže vplývať na prostredie. Pred tréningom nemá žiadne poznatky o prostredí (pred prvou hrou v živote). Počas tréningu agent vykonáva prípustné akcie, za ktoré mu prostredie pridelí kladnú (výhra, vyradenie súperovej figúrky), neutrálnu alebo zápornú (prehra) odmenu. Taktiež po každej akcii dostane aj informáciu o novom stave prostredia z jeho pohľadu (napr.: nové rozloženie hracej plochy po súperovom ťahu). Túto informáciu o stave si spolu s odmenou agent uloží ako skúsenosti a na základe získaných skúseností potom upravuje svoju stratégiu tak, aby v budúcnosti dosiahol čo najvyššie odmeny.

Jeden z problém pre agenta pri učení posilňovaním je určiť pomer skúšania nových akcií (angl.: exploration) a využívania už natrénovaných akcií (angl.: exploitation). Aby

dosiahol, čo najvyššie odmeny musí používať akcie už vyskúšal a viedli k dobrým odmenám. Ale aby takéto akcie našiel musí skúšať aj akcie s neznámymi výsledkami.

Prostredie reprezentuje, alebo obsahuje ako svoju časť určitú úlohu (v našom prípade je to hracia plocha, časovač, a súper), v ktorej sa agent snaží dosiahnuť nejaký cieľ (vyradiť súperove figúrky, vyhrať).

Ďalšie dôležité časti učenia posilňovaním sú **stratégia** (angl.: policy), **signál odmien** (angl.: reward signal), **hodnotiaca funkcia** (angl.: value function) a **model** prostredia.

Počas tréningu sa agent pokúša vylepšovať svoju stratégiu. **Stratégia** agenta udáva správanie agenta v danej situácii. Jednoducho povedané, stratégia je mapovanie stavu prostredia na akcie, ktoré agent vykoná. Môže mať podobu jednoduchej funkcie, vyhľadávacej tabuľky alebo rozsiahleho vyhľadávacieho procesu. Rozlišujeme dva hlavné prístupy trénovania stratégie: **off-policy** a **on-policy**. On-policy prístup pri tréningu upravuje stratégiu, ktorú používa na výber akcií. Naopak off-policy môže využívať na výber akcií aj iné metódy. Príkladom off-policy môže byť – Q-učenie (angl.: Q-learning) a príkladom on-policy je PPO (Proximal Policy Optimization) agent používaný aj v tejto práci.

Za každú akciu dostane agent z prostredia odmenu. **Odmena** môže byť kladná – pri dosiahnutí celkového alebo čiastkového cieľa, alebo záporná – pri nesprávnej akcii alebo porušení pravidiel. Spôsob udeľovania týchto odmien je nazývaný **signál odmien** a definuje cieľ riešeného problému. Tento signál je plne riadený prostredím. Agent nemá naň žiadny vplyv, môže ovplyvniť iba jednotlivé odmeny a to zmenou svojej stratégie resp. akcie, ktorú v danom momente vykoná. Ak je akcia, vykonaná podľa stratégie, odmenená nízkou alebo zápornou odmenou, tak sa stratégia môže zmeniť aby sa nabudúce vykonala iná akcia, ktorá môže viesť k vyšším odmenám.

Hodnotiaca funkcia má za úlohu určovať hodnoty pre možné akcie. **Hodnota** (angl.: value) sa snaží čo najlepšie odhadovať budúce odmeny – odmeny, ktoré môžeme očakávať v stavoch po vykonaní danej akcie. Agent dáva veľký dôraz na hodnoty preto, aby dokázal vykonať aj akciu, ktorá v danom momente môže byť zlá (dostane nízku alebo zápornú odmenu) ale môže neskôr viesť k vyšším odmenám. Efektívne odhadovať hodnoty je jednou z najhlavnejších a najťažších častí učenia posilňovaním.

Poslednou spomínanou časťou učenia posilňovaním je **model** prostredia. Jeho úlohou je snažiť sa predpovedať správanie prostredia – napríklad predpovedaním nasledujúceho stavu a odmeny. Napomáha agentovi plánovať svoje akcie vopred tým, že berie do úvahy aj možné budúce situácie, ktoré môžu nastať. Metódy, ktoré

využívajú modely, sú nazývané **model-based**. Naopak metódy, ktoré používajú pokus-omyl sú nazývané **model-free**. [3]

2.1.1 Proximal Policy Optimization

PPO agent (Proximal Policy Optimization) je agent používaný pri učení s posilňovaním, ktorý bol vytvorený pre stabilnejší priebeh tréningu a efektívne zlepšovanie stratégie.

Agent používa pri tréningu dve siete alebo sieť s dvomi hlavami (sieť, ktorá sa rozdelí do separátnych sietí – každá z rozdelených sietí sa nazýva hlava). Prvá hlava sa nazýva actor (slov.: konateľ) a jej úlohou je vytvárať pravdepodobnostné rozdelenie akcií, z ktorého sa bude vyberať akcia na vykonanie. Druhá hlava sa nazýva kritik (angl.: critic) a je zodpovedná za predpovedanie hodnôt (angl.: values) pre daný stav. Tá sa používa na výpočet tzv. výhod, ktoré navádzajú úpravy stratégie.

Pri úprave stratégie agent využíva niekoľko pomôcok na to aby bola úprava efektívna a stabilná. Prvá z týchto pomôcok je advantage estimator (slov.: odhadovač výhod), ktorý vytvára **funkciu výhod** (angl.: advantage funtion). V tejto funkcii sa častokrát využíva tzv. Generalized Advantage Estimation (GAE, slov.: všeobecný odhad výhod). Funkcia slúži na porovnanie akcie k aktuálnemu odhadu výhod – ako sa daná akcia porovnáva s ostatnými v aktuálnom stave. To znamená, že akcie s vyššími výhodami sa viacej utvrdzujú v stratégii.

Ako druhú pomôcku si agent ukladá predošlú stratégiu pomocou, ktorej vypočítava pomer zmeny od novej stratégie ($r(\theta)$).

Tento pomer sa používa pri tretej najhlavnejšej pomôcke a to je stratová funkcia (angl.: loss function), ktorá tento pomer ohraničí pomocou parametra epsilon (ε) do intervalu $<1-\varepsilon;1+\varepsilon>$. Parameter ε teda určuje maximálnu povolenú zmenu stratégie. Toto orezávanie znižuje motiváciu prehľadávať akcie s vysokou zmenou stratégie. [4]

Pri PPO agentovi, je pred tréningom potrebné nastaviť niekoľko parametrov.

Learning rate – lr, určuje veľkosť kroku pri zostupe gradientom (angl.: gradient descent). Pri vyšších hodnotách lr môže nastať nestabilný tréning a pri veľmi nízkych môže spomaliť tréning. Odporúčané hodnoty sú v intervale $<\mathbf{1e^{-5}}; \mathbf{1e^{-3}}>$.

Buffer size udáva veľkosť **pamäte skúseností** (experience buffer) agenta, ktoré sa ukladajú pred každým vykonaním kroku učenia (vykonanie niekoľko zostupov gradientom). Odporúčané veľkosti pamäte sú od **2048** po **409 600**.

Počet **epoch** určuje koľkokrát bude použitá pamäť skúseností pri kroku učenia. Odporúčaná hodnota je v rozmedzí **3** a **10**.

Batch size určuje počet uložených skúseností, ktoré sa použijú na jeden zostup gradientom. Pre diskrétne úlohy sa odporúča hodnota z rozsahu **32**-**512** a pre spojité úlohy rozsah **512**-**5120**. Taktiež sa odporúča aby hodnota batch size bola deliteľom hodnoty buffer size.

Epsilon – ε upravuje už spomínané ohraničovanie pri loss funkcii. Odporúčaná hodnota pre tento parameter je od 0.1 po 0.3.

Beta (alebo koeficient entropie, angl.: entropy coeficient) zabezpečuje pokles entropie počas tréningu. Zvýšenie **entropia** znamená, že agent bude vyberať náhodnejšie akcie. Počas tréningu by sa mala entropia znižovať. Ak sa entropia znižuje príliš rýchlo odporúča sa zvýšiť parameter beta, naopak ak sa entropia znižuje pomaly je potrebné zvýšiť beta. Odporúčaný rozsah hodnoty parametru beta je od $1e^{-4}$ po $1e^{-2}$. [5]

A iné parametre, ktorými sa ale v práci nebudeme zaoberať.

2.1.2 Analýza súčasného stavu

Učenie posilňovaním sa používa na riešenie veľa rôznych problémov z reálneho ale aj virtuálneho sveta. V reálnom svete bolo využívané napríklad pri vývoji nových čipov, nazývaných Tensor Porcessing Unit (TPU), ktoré boli vyvinuté na lepšiu prácu s n-rozmernými vektormi (Tensors), ktoré sú často používané pri strojovom učení. [6]

Na virtuálny svet hier sa najviac zameriavajú vedci z firmy DeepMind, kde vytvorili niekoľko agentov v rôznych hrách, ktorý boli schopný poraziť ľudských hráčov. Jedným z najznámejších príkladov je AlphaZero – agent, ktorý dokázal poraziť aj vtedy najlepší šachový počítač „Stockfish“. [7]

Prieskum „Reinforcement Learning for Combinatorial Optimization: A Survey“ [2] sa zameriava na používanie učenia posilňovaním na riešenie kombinatorických optimalizačných úloh ako je napríklad úloha obchodného cestujúceho. Odkazuje sa na rôzne články, ktoré sa tomu venujú a snaží sa poskytnúť zhrnutie a porovnanie doposiaľ dosiahnutých výsledkov. Poskytujú niekoľko spôsobov ako rozdeliť rôzne prístupy z článkov do skupín pre lepšie porovnanie.

Prvé rozdelenie, ktoré definujú je podľa použitej metódy učenia posilňovaním prvého levelu (model-based a model-free) alebo druhého levelu (policy-based, value-based, Monte-Carlo based). Druhý navrhovaný spôsob delenia je podľa spôsobu učenia stratégie. Na jednej strane popisujú základný postup učenia (angl.: principal learning) tak, že agent vytvára celé riešenie samostatne a na druhej strane spojený postup učenie (angl.: joint training), pri ktorom sa učí robiť správne kroky v rámci už existujúceho algoritmu (napr.: heuristika vetiev a hraníc). Posledný navrhovaný spôsob

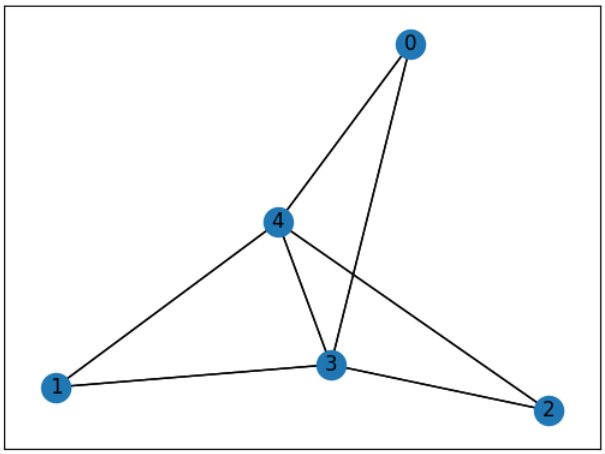
delenia je založený na spôsobe vytvárania riešenia. Rozdeľuje postupy na konštrukčné (angl.: construction) a vylepšujúce (angl.: improvement). Pri konštrukčných postupoch agent vytvára riešenie od začiatku až po koniec a pri vylepšujúcich postupoch agent dostane už vytvorené riešenie a jeho úlohou je ho čo najviac vylepšiť.

V článku taktiež popisujú výhody, ktoré môže učenie posilňovaním priniesť do oblasti riešenia kombinatorických optimalizačných úloh. Medzi najhlavnejšie výhody patrí zníženie časovej náročnosti oproti bežne používaným heuristikám, ale taktiež popisujú problémy, ktoré je potrebné ešte prekonať ako napríklad generalizácia a škálovateľnosť na väčšie úlohy. [2]

Spojením učenia posilňovaním a grafových neurónových sietí vznikol napríklad aj článok „Solving uncapacitated P-Median problem with reinforcement learning assisted by graph attention networks“ [8], v ktorom autori natrénovali agenta na riešenie optimalizačných úloh nazývaných p-medián. Vo výsledkoch ukázali, že ich agent dokáže riešiť tieto úlohy s malou veľkosťou (do 100 vrcholov) lepšie ako bežne používané heuristiky a strednou (do 300 vrcholov) veľkosťou porovnateľne s heuristikami. Hlavnou výhodou ako bolo spomínané aj v predošlom odstavci je rýchlosť vytvárania riešenia, ktorá je aj pri veľkých úlohách značne vyššia. [8]

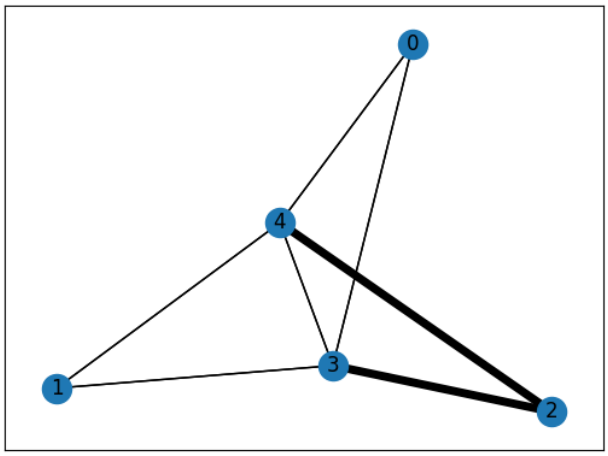
2.2 Grafové štruktúry

Grafy (angl.: simple graphs) sú usporiadané dvojice *G*=(*V*, *H*). Prvky v množine V sa nazývajú **vrcholy** (angl.: vertices, nodes) a prvky v množine H sú **hrany** (angl.: edges). Množina V musí obsahovať aspoň jeden prvok a H je množina neusporiadaných dvojíc, resp. usporiadaných dvojíc pri orientovanom grafe, typu {u, v} takých, že u ∈ V , v ∈ V a u ≠ v. To znamená, že každá hrana grafu začína v niektorom z vrcholov z množiny V a končí v inom vrchole z množiny V.



Obrázok 1 Príklad grafu

Vrchol *v* je incidentný s hranou *h* v grafe G=(V, H), práve vtedy keď vrchol *v* je jedným z vrcholov hrany *h*. Množina hrán incidentných s vrcholom *v* sa označuje H(v). Inými slovami povedané vrchol je incidentný s hranou ak daná hrana začína alebo končí v danom vrchole (Obrázok 2).



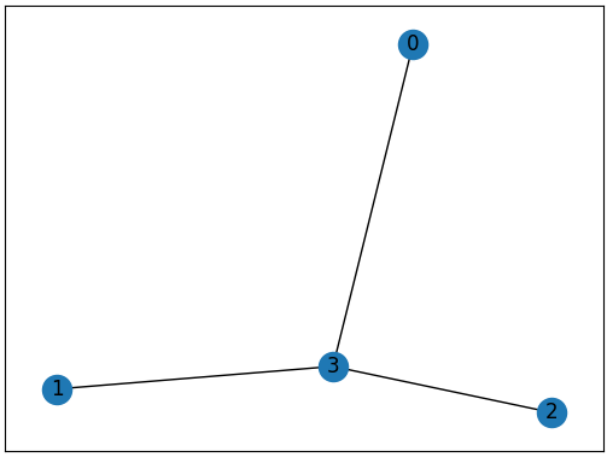
Obrázok 2 Vyznačené hrany incidentné s vrcholom 2

Stupeň vrchola deg(*v*) je počet hrán incidentných s vrcholom v – deg(v) = |H(v)|.

Tabuľka 1 Stupne vrcholov v grafe z Obrázok 1

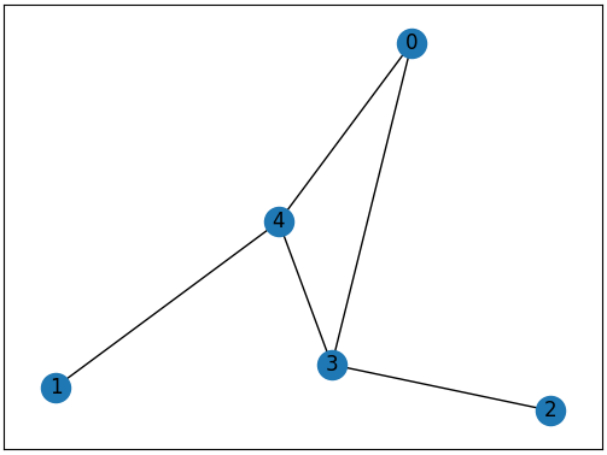
Index	0	1	2	3	4
Stupeň	2	2	2	4	4

Podgrafom grafu G=(V, H) je graf G′=(V′, H′) práve vtedy ak platí V′ ⊆ V a H′ ⊆ H. Podgraf je teda graf vytvorený iba z vrcholov a hrán pôvodného grafu. Je dôležité povedať, že podgraf musí obsahovať všetky vrcholy, z ktorých a do ktorých vedú hrany podgrafu.



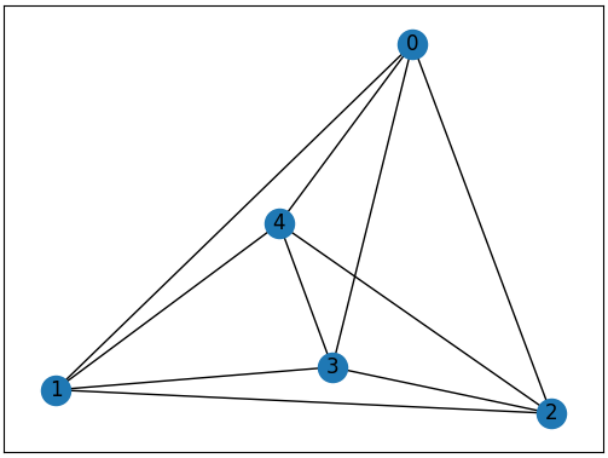
Obrázok 3 Podgraf grafu z Obrázok 1

Keď v podgrafe G′=(V′, H′) grafu G=(V, H) platí V=V′ - nachádzajú sa v ňom všetky vrcholy pôvodného grafu, nazýva sa **faktorovým podgrafom**.



Obrázok 4 Príklad faktorového podgrafu z Obrázok 1

Úplný graf je taký graf, kde množina H obsahuje všetky možné kombinácie (u, v) také, kde u, v ∈ V a u ≠ v. Inak povedané obsahuje hranu z každého vrcholu do každého iného vrcholu v grafe. Úplný graf o n vrcholoch sa značí K_n.

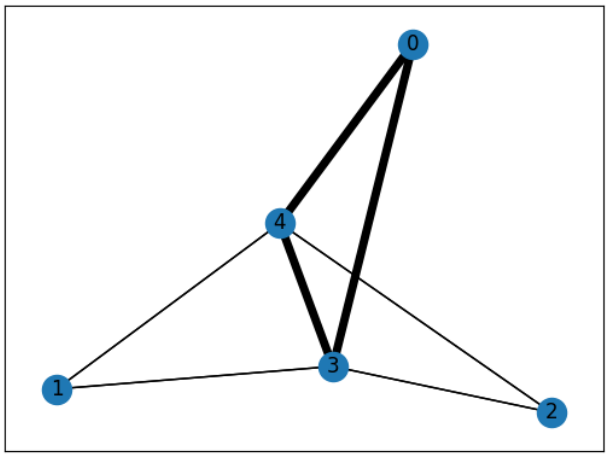


Obrázok 5 Príklad úplného grafu

Hranovo ohodnotený graf G=(V, H, c) obsahuje navyše aj funkciu c: H → R definovanú na množine H. funkcia priradzuje každej hrane h reálne číslo c(h) nazývané **cena hrany**.

Sled v grafe G = (V, H) je ľubovoľná alternujúca postupnosť vrcholov a hrán tvaru $\mu(v_1, v_k) = (v_1, \{v_1, v_2\}, v_2, \{v_2, v_3\}, v_3, \ldots, \{v_k-1, v_k\}, v_k)$. Ak sa v slede neopakuje žiadna hrana nazývame ho **ťah** a ak sa neopakuje žiadny vrchol, tak tento sled nazývame **cesta**.

Ak je v ťahu prvý vrchol ten istý ako posledný – $v_1=v_k$, tak sa nazýva **uzavretý**. Ak navyše platí, že sa žiadny iný vrchol neopakuje, tak takýto sled sa nazýva **cyklus**. Ak sa v grafe nenachádza cyklus, tak sa nazýva **acyklický graf**.

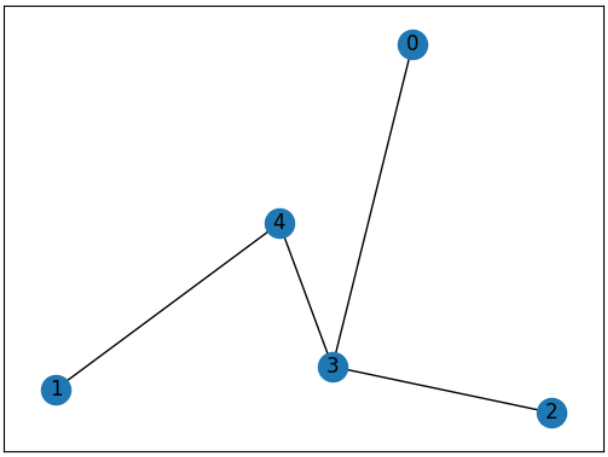


Obrázok 6 Príklad cyklu v grafe z Obrázok 1

V **súvislom grafe** existuje medzi každými dvoma vrcholmi u, v ∈ V cesta.

Strom je taký graf, ktorý je zároveň súvislý a acyklický.

Kostra grafu je faktorový podgraf, ktorý je zároveň aj stromom. Inak povedané je to podgraf, ktorý obsahuje všetky vrcholy pôvodného grafu, je súvislý a neobsahuje cyklus.



Obrázok 7 Kostra grafu z Obrázok 1

Ak vytvárame kostru z hranovo ohodnoteného grafu G=(V, H, c), môžeme vypočítať cenu kostry c(K). **Cena kostry** je súčet všetkých ohodnotení hrán danej kostry.

Najlacnejšia kostra je kostra grafu, ktorá má najmenšiu cenu. To znamená, že neexistuje lacnejšia kostra daného grafu. Opačne kostra s najväčšou cenou sa nazýva **najdrahšia kostra**.

Na hľadanie najlacnejšej kostry existuje jednoduchý algoritmus – Kruskalov algoritmus. Algoritmus pozostáva z troch krokov:

1. Zoraď hrany grafu $G = (V, H, c)$ podľa ich ohodnotenia c od najlacnejšej po najdrahšiu do postupnosti
2. Vyberieme a odstránime prvú hranu z postupnosti. Ak nevytvára s hranami už zaradenými do kostry cyklus, tak ju zaradíme do kostry.
3. Ak sa počet hrán v kostre rovná $|V| - 1$ alebo už nie sú hrany v postupnosti – algoritmus končí. Inak prejdeme znovu na krok 2.

Algoritmus vieme jednoducho upraviť na hľadanie najdrahšej kostry úpravou prvého kroku tak, že vytvoríme postupnosť od najdrahšej hrany po najlacnejšiu. [9]

2.3 Úloha obchodného cestujúceho

Jedna z kombinatorických úloh z teórie grafov je úloha obchodného cestujúceho (angl.: traveling salesman problem).

V úlohe obchodného cestujúceho riešenej na súvislom hranovo ohodnotenom grafe G=(*V*, *H*, *c*) je cieľom navštíviť všetky vrcholy riešeného grafu, tak aby cena prejdených hrán bola čo najnižšia.

Formálna definícia znie tak, že v súvislom hranovo ohodnotenom grafe je potrebné nájsť najkratší hamiltonovský sled. Ak zakazujeme návštevu vrcholov viac ako jedenkrát, tak je potrebné nájsť hamiltonovský cyklus.

Hamiltonovský sled (resp. cyklus) je taký sled v grafe, ktorý obsahuje všetky vrcholy grafu.

Nájsť najkratší hamiltonovský cyklus je NP-ťažká úloha. Z čoho vyplýva že úloha obchodného cestujúceho je NP-ťažká. Čo znamená, že táto pri tejto úlohe je veľmi ťažké, možno aj nemožné nájsť optimálne riešenie.

Preto sa pri riešení týchto úloh používajú suboptimálne algoritmy alebo heuristiky na zistenie riešení blízkych k optimu.

Príkladom takejto heuristiky môže byť takzvaná pažravá metóda (angl.: greedy algorithm). V úplnom hranovo ohodnotenom grafe *G*=(*V*, *H*, *c*) heuristika prebieha v troch korkoch:

1. Vyber ľubovoľný vrchol a do riešenia pridaj najlacnejšiu hranu incidentnú s týmto vrcholom
2. Ak bolo vybraných |*V*| - 1 hrán, dokonči riešenie uzavretím cyklu (pridaním hrany z posledného navštíveného vrcholu do počiatočného vrcholu) a algoritmus končí.
3. Vyber takú hranu incidentnú z posledným vrcholom vytváraného riešenia, ktorá ešte nie je v riešení, nie je incidentná so žiadnym iným vrcholom v riešení a je najlacnejšia možná. Následne choď na krok 2. [9]

Je dôležité si uvedomiť, že pri riešení takýchto kombinatorických úloh, je veľmi jednoduché pokaziť riešenie vykonaním jednej zlej akcie (výberom zlého vrcholu alebo hrany).

2.4 Grafové neurónové siete

Grafové neurónové siete (angl.: Graph neural netwrok - GNN) sú siete schopné pracovať s grafovou reprezentáciu.

Grafy používané v grafových neurónových sieťach sú reprezentované pomocou niekoľkých vektorov a matíc. Konkrétne sa skladá z matice vrcholov, zoznamu hrán a nepovinnej matice vlastností hrán. Matica vrcholov je matica s rozmermi *n***v*, kde *n* je počet vrcholov a *v* je počet vlastností vrcholu. Susednosť sa neurčuje pomocou matice susedností ale pomocou zoznamu hrán (angl.: adjacency list, edge list), ktorý je pamäťovo efektívnejší. Zoznam hrán je definovaný ako zoznam usporiadaných dvojíc (*u*, *v*), kde *u* a *v* sú indexy vrcholov grafu. Nie všetky grafy obsahujú vlastnosti (napr.: cenu) na hranách a preto je matica vlastností hrán nepovinná. Má rozmery *n***v*, kde *n* je počet hrán grafu a *v* je počet vlastností hrany.

Najjednoduchšia GNN je taká, ktorá na každú časť grafu použije vlastný multilayer perceptron (MLP). Pre každý vrcholový vektor sa použije MLP a dostaneme aktualizovaný vrcholový vektor. To isté platí aj pre hrany. V GNN sa taktiež vytvára global-context vector (slov.: vektor globálneho kontextu), ktorý sa učí nejakú vlastnosť celého grafu.

Jednou z hlavných súčastí grafových neurónových sietí je tzv. message passing (slov.: posielanie správ). Ide o proces výmeny informácii medzi susednými vrcholmi aby sa sieťam podala informácia o spojeniach v grafe. Najjednoduchší typ message passing-u funguje v troch krokoch:

1. Pre každý vrchol zozbieraj informácie od susedných vrcholov.
2. Agreguj všetky zozbierané informácie pomocou nejakej funkcie, napr.: súčet, max, priemer
3. Agregované dáta sa posúvajú do aktualizačnej funkcie (angl.: update function)

Message passing nám umožňuje riešiť úlohy na grafoch s rôznymi stupňami vrcholov.

Pridávaním viacerých GNN sa pre každý vrchol agreguje väčšie a väčšie okolie podobne ako pri bežných konvolučných sieťach. [10]

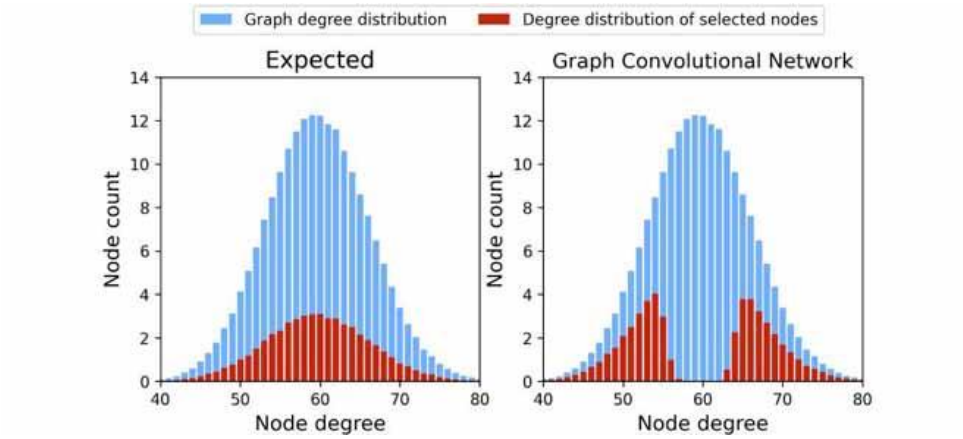
Špeciálnym typom GNN používaným aj v tejto práci je Graph Attention Network (GAT). Táto sieť sa navyše v každom vrchole priraďuje každému zo svojich susedných vrcholov tzv. attention coeficient, ktorý určuje dôležitosť daného vrcholu pri agregácii okolia. Pre každý vrchol môže existovať aj niekoľko týchto koeficientov inak nazývaných aj hláv (angl.: heads). Toto umožňuje určovať dôležitosť susedného vrcholu pre viac vlastností naraz.

GAT taktiež po agregácii aplikuje nelineárnu aktivačnú funkciu (angl.: activation function) napríklad ELU, čo pridáva možnosť naučiť sa komplexnejšie informácie z grafu. [11]

2.4.1 Exploration Degree Bias

Aj keď grafové neurónové siete preukázali užitočnosť v rôznych úlohách, bolo preukázané v nich preukázané skreslenie, ktoré môže negatívne vplývať na tréning. Pri učení posilňovaním to môže obmedzovať schopnosť agenta objavovať prostredie.

V článku [1] autori skúmali, či veľkosť stupňov vrcholov vplýva na pravdepodobnosť výberu daného vrcholu. Hlavnou myšlienkou bolo, že vrcholy s väčším stupňom budú pri agregácii svojho okolia zbierať informácie z viacerých okolitých vrcholov, čím sa môže zvýšiť výstupná hodnota neurónov (logit). Táto zvýšená hodnota posilňuje signál vyslaný z daného vrcholu, čo môže na konci viesť k preferencii výberu daného vrcholu. Tento jav môže obmedziť schopnosť agenta objavovať nové riešenia, ak je pri nich potrebný výber vrcholu s priemerným stupňom. Toto skreslenie nazvali Exploration Degree Bias (EDB).



Obrázok 8 očakávaný vs. skutočný výber vrcholov [1]

Z grafov na obrázku môžeme vidieť očakávanú distribúciu výberu vrcholov (naľavo) v porovnaní so skutočnou distribúciou výberu (napravo). Môžeme vidieť, že v skutočnosti skúmané grafové neurónové siete preferujú vyberať vrcholy s väčším alebo menším stupňom vrcholu a vynechávajú vrcholy s priemerným stupňom.

Bolo ukázané, že EDB sa prejavuje v grafových neurónových sieťach GCN, GraphSAGE, GAT, a GIN, z ktorých sa v GCN a GIN skreslenie prejavuje najviac.

V článku boli testované dva typy agentov DQN (Deep Q Network) a PPO (Proximal Policy Optimization). Ukázalo sa, že PPO agent dokázal zmierniť vplyv EDB pomocou pravdepodobnostného výberu akcii. [1]

2.4.2 Analýza súčasného stavu

Skreslenie podľa stupňov vrcholov ako súčasť grafových neurónových sietí bolo skúmané už v niekoľkých vedeckých článkoch zameraných na supervised learning (slov.: učenie s učiteľom). [12] [13] V článku [1], od ktorého sa odvíja aj naša práca sa autori zamerali na vplyv skreslenia pri používaní učenia posilňovaním, ktorého výsledky sme popísali vyššie.

V článku od Liu a spol. [12] skúmali toto skreslenie a navrhli nový typ GNN nazývaný DegFairGNN. Táto sieť sa snaží zmierňovať skreslenie tak, že sa učí tzv. proti skresľovaciu funkciu (angl.: debiasing function), ktorá sa snaží vyrovnávať rozdiely spôsobené rôznymi stupňami vrcholov. [12]

Článok od autorov Wu a spol. [13] navrhuje zmierniť skreslenie pomocou tzv. Self-supervised Graph Learning (SGL). Tento prístup učenia vytvára pre každý vrchol niekoľko upravených pohľadov pomocou troch operácií – node dropout, edge dropout a random walk - ktoré upravujú reprezentáciu grafu rôznymi spôsobmi, čo pomáha agentovi zmierňovať vplyv skreslenia. [13]

3 NÁVRH RIEŠENIA

Popísaný problém navrhujme riešiť pomocou programovacieho jazyka Python. Jazyk Python obsahuje veľké množstvo knižníc vytvorených pre potreby tréningu neurónových sietí a strojového učenia. Taktiež knižnicu „torch_geometric“, ktorá poskytuje možnosť tvorby grafových údajových štruktúr a neurónové vrstvy schopné pracovať s vytvorenými grafovými štruktúrami, na ktorých bol dokázaný výskyt riešeného skreslenia.

Naším cieľom je skúmanie prítomnosti EDB v grafových NN. Hlavne nás zaujíma vplyv EDB na proces tréningu RL agenta. To znamená ako sa EDB vyvíja počas tréningu a aké má dopady na kvalitu nájdeného riešenia. Pre testovanie sme vybrali 2 úlohy, ktoré majú nasledovné vlastnosti:

- Vieme dané úlohy formulovať ako sekvenčný rozhodovací proces vhodný pre riešenie pomocou RL.
- Prípustné riešenie úlohy bude obsahovať všetky vrcholy.
- Predpokladáme, že stupeň vrcholu nie je kľúčová informácia pre vytváranie riešenia. Ideálne je aby kvalita riešenia bola agnostická voči stupňu vrcholu. Hlavne je dôležité, aby náhodne inicializovaná sieť, ktorá v prípade prítomnosti EDB bude vyberať vrcholy podľa stupňa vrcholu, nedosahovala dobré riešenie.

Pre riešenie úlohy boli vybrané dve úlohy s rôznou obtiažnosťou. Prvou je hľadanie najlacnejšej (resp. najdrahšej) kostry. Ide o úlohu, ktorá má exaktné riešenie, a teda ju považujeme za pomerne jednoduchú. Druhou vybranou úlohou bola úloha obchodného cestujúceho. Pri tejto úlohe je potrebné navštíviť všetky vrcholy a ide o NP ťažkú úlohu. Predpokladáme, že úloha hľadania najkratšej kostry nebude predstavovať zásadný problém a preto bude potrebné skúmať aj náročnejšiu úlohu, teda úlohu obchodného cestujúceho. Plánujeme preskúmať obidve úlohy, ale ak sa nepodarí vyriešiť úlohu hľadania kostry, tak odporúčame zostať iba pri tejto úlohe a detailne ju preskúmať

Na dosiahnutie cieľov práce sme navrhli nasledovný postup:

1. Implementácia prostredia a agenta
2. Otestovanie funkčnosti prostredia a agenta
3. Vykonanie experimentov na úlohe prvého typu
4. Vyhodnotenie výsledkov
5. Vykonanie experimentov na úlohe druhého typu
6. Vyhodnotenie výsledkov

3.1 Postup vykonávania experimentov

Pre plynulý priebeh tréningov a prípadné odladenie chýb navrhujeme nasledovný postup:

1. Otestovanie prostredia a funkčnosti tréningu
2. Malé statické úlohy
3. Väčšie statické úlohy
4. Testovanie generalizácie
5. Analyzovať vplyv EDB na tréning

Ak sa v ktoromkoľvek kroku nepodarí natrénovať agenta, odporúčame nepokračovať k ďalším krokom, ale zistiť prečo sa agent nedokázal natrénovať. Zistiť, či malo EDB vplyv na neúspech tréningu.

Pre malé úlohy navrhujeme veľkosť 10 až 25 vrcholov, pre veľké 50 až 100 vrcholov. Pod statickými úlohami myslíme úlohy, pri ktorých sa riešený graf počas tréningu nemení, naopak pri testovaní generalizácie sa po každom riešení vygeneruje nový graf – tzn. že agent neuvidí jeden riešený graf viac ako raz.

3.1.1 Agent

Pri tréningu navrhujeme použiť PPO agenta.

Vytvorená trieda pre agenta by mala byť schopná jednoducho upraviť parametre pre učenie (learning rate, gamma, epsilon, entropy,...) a tréning (počet iterácií, počet epoch,...).

Agent by nemal byť závislý na neurónovej sieti, ktorú bude možné nastaviť pred tréningom. Pri tréningoch navrhujeme používať napríklad grafové neurónové vrstvy GAT (Graph Attention), ktoré sú schopné spracovať aj informácie z hrán na grafe a na ktorých bolo dokázané aj riešené skreslenie (EDB).

Pre možnosť znovu spustenia tréningu z vybraných iterácií by agent mal ukladať váhy neurónovej siete pri zlepšení výsledku alebo po každých *x* iteráciách, kde *x* je prirodzené kladné číslo. Taktiež uložiť svoje váhy po ukončení tréningu aby bolo možné ich načítať pre potreby testovania.

Aby sme vedeli spätne pozorovať aj priebeh tréningu navrhujeme aby sa ukladali aj priemerné odmeny a v každej iterácii tréningu aj hodnoty doposiaľ najlepšej odmeny. Na pozorovanie korelácie je potrebné ukladať vypočítané veľkosti korelácie, aby sme vedeli po tréningu pozorovať jej vplyv. Môžu sa ukladať aj informácie pre ladenie agenta – v každej iterácii hodnota entropie, learning rate-u a iných.

3.1.2 Prostredie

Prostredie by malo zodpovedať za tieto časti tréningu:

- Reštartovať riešenie
- Udržiavanie stavu
- Úpravu stavu podľa akcie agenta
- Určovanie odmien
- Maskovanie akcií (určovanie povolených akcií)
- Prechod stavu
- Ukončenie behu prostredia

Navrhujeme aby prostredie udržiavalo informácie o aktuálnom stave prostredia – udržiavať aktuálny graf na ktorom sa rieši úloha. Taktiež ukladať informáciu o optimálnom riešení a iné dáta, ktoré sa počas tréningu nemenia alebo sa menia ale.

Pri každej akcii agenta by prostredie malo upraviť stav – označiť agentom vybraný vrchol/hranu. Následne zistiť, či bol dosiahnutý cieľ úlohy, ak áno tak ukončiť riešenie, vypočítať odmenu a vrátiť informáciu o odmene a ukončení agentovi.

Ak nebol dosiahnutí cieľ úlohy, tak by prostredie malo určiť odmenu pre agenta – vo väčšine prípadoch by mala byť rovná nule. Potom zistiť aké akcie vedú k prípustným riešeniam a vytvoriť masku akcií, tú s novým stavom a odmenou vrátiť agentovi.

Na to aby agent vykonával iba prípustné akcie navrhujeme aby prostredie vytváralo masky, ktoré budú hovoriť o tom, ktoré akcie môže agent vykonať v danom kroku. Tieto masky budú poskytnuté agentovi po každom vykonanom kroku pre nasledujúci krok. To znamená, že po prvom kroku dostane masku platnú pre druhý krok, po druhom kroku pre ďalší atď. Masku pre prvý krok agent dostane pri reštartovaní prostredia.

Vypočítavanie odmien, informácie o reprezentácii stavu a maskovanie akcií budú konkrétnejšie popísané pri riešených úlohách.

3.1.3 Generovanie grafov

Aby sme odľahčili prostredie o povinnosť generovať grafy navrhujeme vytvoriť triedu, ktorá bude zodpovedná za generovanie náhodných grafov podľa špecifikovaných parametrov na požiadanie.

Trieda by mala byť schopná generovať náhodné grafy s cenovo ohodnotenými hranami G=(V, H, c) podľa týchto parametrov: počet vrcholov, maximálny počet hrán, minimálna

cena hrany, maximálna cena hrany a iných podľa potreby experimentov. Je taktiež potrebné aby boli vytvárané súvislé grafy.

Pri riešení statických úloh by mala byť schopná zafixovať jeden graf a poskytovať iba ten. Taktiež sa vytvorené grafy môžu aj uložiť, aby sa dalo vykonať niekoľko tréningov na rovnakom grafe na porovnanie rôznych prístupov.

3.2 Testovanie

Pri testovaní by agent mal vyberať akcie bez akéhokoľvek vplyvu náhody, preto navrhujeme vyberať akciu s najväčším signálom.

Pri statických úlohách navrhujeme vyhodnocovať riešenie, ktoré vytvorí agent akciami s najväčším signálom. Pri náhodne generovaných úlohách navrhujeme vyhodnotiť a porovnať riešenia na niekoľkých náhodne vygenerovaných úlohách.

Agentov navrhujeme porovnávať oproti trom rôznym hodnotám. Ako prvá porovnávacia hodnota odporúčame optimálne riešenie úlohy. Ak nebude možné vypočítať optimálne riešenie môže sa použiť výsledok vybranej heuristiky. Druhú porovnávaciu hodnotu navrhujeme aby vytvorila náhodne inicializovaná sieť s rovnakými parametrami. Týmto sa bude dať zistiť aké zlepšenie nastalo počas tréningu. Poslednou porovnávacou hodnotou by mala byť hodnota riešenia vytvoreného rovnomerným náhodným výberom akcií. Porovnaním s touto hodnotou sa bude dať určiť či sa agent naučil nejaké riešenie alebo vyvára riešenia náhodne.

3.2.1 Korelácia logitov a stupňov vrcholu

Aby sme vedeli určiť vplyv riešeného skreslenia (EDB), navrhujeme merať koreláciu medzi výstupnými logitmi a stupňami vrcholu rovnako ako v článku EDB [1].

Na zistenie počiatočného vplyvu EDB navrhujeme aby sa korelácia zistila aj pri náhodne inicializovaných sieťach, aby sa dalo zistiť, aký veľký vplyv má skreslenie na začiatku tréningu. Túto koreláciu je tiež potrebné porovnať aj s náhodným výberom akcií, aby sme videli porovnať, či je vplyv korelácie na výsledok výrazný.

Taktiež bude dobré vyhodnotiť aj výsledky korelácie z dát uložených počas tréningu a pozorovať, či sa korelácia počas tréningu nejako mení. Tieto informácie spolu s informáciou, či sa agent niečo naučil, môžeme použiť na zistenie vplyvu EDB počas tréningu.

3.3 Formulácia riešenej úlohy

Na analýzu vplyvu skreslenia sme sa rozhodli vybrať úlohy, v ktorých si myslíme, že skreslenie nebude mať veľký vplyv na riešenie daných úloh. Ako úlohu prvého typu sme si vybrali úlohu hľadania najlacnejšej kostry v grafe. Ak sa podarí úspešne natrénovať danú úlohu a analyzovať vplyv skreslenia, začneme tréning a analýzu na druhej úlohe - úlohe obchodného cestujúceho.

Dané úlohy obsahujú hlavnú hodnotiacu informáciu na hranách grafu, preto pri implementácii budeme musieť vybrať také grafové neurónové siete, ktoré sú schopné pracovať aj s informáciami na hranách.

3.3.1 Hľadanie najlacnejšej kostry v grafe

Prvý problém, ktorý vzniká pri úlohe hľadania najlacnejšej kostry je, že agent musí vyberať hrany a grafové neurónové siete pracujú primárne s vrcholmi, musíme upraviť buď reprezentáciu úlohy alebo postup výberu hrán. Navrhujeme dva možné prístupy.

Prvý prístup upraví reprezentáciu tak, že každú hranu grafu rozdelíme na polovicu novým vrcholom a povolíme agentovi vyberať iba tieto nové vrcholy.

Druhý prístup neupravuje reprezentáciu ale spôsob výberu hrán rozdelí do dvoch rozhodovacích krokov (dvojkrokový prístup). V prvom rozhodovacom kroku sa vyberie akýkoľvek vrchol z grafu a v druhom kroku sa vyberie druhý vrchol taký, do ktorého vedie hrana incidentná s vrcholom vybraným v prvom kroku. Tieto dva kroky opakujeme až do dosiahnutia prípustného finálneho riešenia.

V našej práci budeme preskúmavať druhý prístup preto, aby sme nemuseli upravovať reprezentáciu grafu.

Návrh na maskovanie akcií pri dvojkrokovom prístupe bude rozdielny pri obidvoch krokoch. Pri tomto prístupe vyberáme vrcholy, preto sú „výbery akcií“ a „výbery vrcholov“ ekvivalentné a v texte ich niekedy zamieňame.

Pred prvým krokom by malo prostredie poskytnúť takú masku, aby znemožnila výber vrcholov, ktoré už nemajú žiadne možné hrany na výber, tzn.: už boli všetky hrany z okolia vybraté alebo výber ktorejkoľvek hrany z okolia by viedol k neprípustnému riešeniu. Ide hlavne o maskovanie vrcholov, ktorých výberom by sa agent dostal do miesta odkiaľ by nemal žiadnu hranu na výber.

Ak sme sa dostali do druhého kroku nejakou odmaskovanou akciou, tak pred druhým krokom môžeme tvrdiť, že existuje aspoň jedna hrana, ktorú agent môže vybrať.

Prostredie by teda malo vytvoriť masku tak, aby znemožnila výber vrcholov pre ktoré platí aspoň jedna z týchto podmienok:

Hrana z vrcholu vybraného v prvom kroku do možného vrcholu...

- neexistuje
- už bola pridaná do riešenia
- vytvorí cyklus v riešení – výber vedie k neprípustnému riešeniu

Navrhujeme aby sa odmeny (*R*) agentovi udeľovali iba po dokončení riešenia vypočítaním rovnice $R = -\frac{U_f}{U_{opt}} + 2$, kde U_f je cena vytvorenej kostry agentom a U_{opt} je cena najlacnejšej kostry, ktorá sa dá vytvoriť z riešeného grafu – optimálne riešenie (optimum). Pomerom $\frac{U_f}{U_{opt}}$ zistíme kvalitu dosiahnutého riešenia v porovnaní s optimálnym riešením – čím vyššia hodnota dosiahnutého riešenia, tým vyššia hodnota pomeru. Ak agent dosiahne optimálne riešenie, tak bude pomer rovný 1. Týmto pomerom dostaneme hodnoty z intervalu <1,∞), avšak pri skutočných úlohách bude interval ohraničený aj zhora hodnotou $\frac{U_w}{U_{opt}}$, kde U_{opt} je hodnota optimálneho riešenia a U_w je hodnota najhoršieho možného riešenia. Keďže sa jedná o minimalizačnú úlohu potrebujeme tento pomer obrátiť tak, aby sme dosiahli najvyššiu hodnotu pomeru pri nájdení optimálneho riešenia. Vynásobením pomeru hodnotou -1 dostaneme odmeny z intervalu (-∞,*-1*>. Ako poslednú úpravu odmenu zvýšime o hodnotu 2, aby sme dostali odmeny z intervalu (-∞,*1*>, kde dosiahnutie optimálneho riešenia znamená odmenu 1.

3.3.2 Úloha obchodného cestujúceho

Pri úlohe obchodného cestujúceho nevzniká problém ako pri prvej úlohe, pretože vyberáme jednu súvislú cestu v grafe. To znamená, že agent pri vyváraní riešenia by mal vyberať vrcholy.

Pri tejto úlohe avšak budeme musieť zabezpečiť to aby pri vytváraní grafov nevznikol vrchol so stupňom menším ako 2, preto sa dalo vytvoriť prípustné riešenie. Ak by existoval vrchol so stupňom 1 nevedeli by sme po výbere takého vrcholu pokračovať v riešení, pretože by sme museli porušiť pravidlo úlohy o navštívení každého vrcholu práve raz.

Maskovanie akcií by malo prebiehať tak, aby pred každým krokom bol znemožnený prístup do vrcholov, ktoré už sú súčasťou riešenia.

Riešenie by sa malo ukončiť práve vtedy, keď sa navštívia všetky vrcholy. Prostredie po ukončení riešenia by malo pridať ešte hranu z posledného vybratého vrcholu do prvého vybratého vrcholu a nakoniec by sa mala vypočítať hodnota riešenia.

Navrhujeme aby sa odmeny udeľovali rovnako ako pri hľadaní najlacnejšej kostry. To znamená po ukončení riešenia rovnicou $R = -\frac{U_f}{U_{opt}} + 2$, kde U_{opt} môže byť buď hodnota optimálneho riešenia alebo hodnota výsledku dosiahnutá náhodným agentom alebo heuristikou.

4 IMPLEMENTÁCIA RIEŠENIA

Pri implementácii riešenia používame programovací jazyk python. Na implementáciu neurónových sietí používame knižnicu „pytorch“ [14] a pri grafových neurónových sieťach knižnicu „pytorch_geometric“ [15]. Medzi ostatné používané knižnice patria aj:

- matplotlib.pyplot [16]– na zobrazovanie výsledkov do stĺpcových alebo čiarových grafov a taktiež zobrazenie riešení, vytvorených natrénovanými agentami
- networkx [17] – knižnica potrebná na zobrazenie grafových štruktúr pomocou matplotlib, taktiež poskytuje niekoľko pomocných metód napríklad na výpočet najlecnejšej kostry
- numpy [18]– poskytuje metódu na výpočet korelácie (corrcoef)
- pandas [19]– poskytuje jednoduché ukladanie a načítavanie dát

V návrhu boli popísané dve odporúčané úlohy na riešenie, ale keďže agenti mali problém sa natrénovať už na malých statických úlohách hľadania najlacnejšej kostry, sme úlohu obchodného cestujúceho neriešili.

Všetky kódy a grafy používané pri experimentoch sú dostupné na githube. [20]

4.1 Prostredie

Prostredie udržiava aktuálne vytvorené riešenie, určuje odmeny pre agenta a určuje povolené akcie (maskovanie akcií).

Pri statických úlohách sa na začiatku tréningu buď vygeneroval náhodný graf podľa parametrov alebo sa načítal už vytvorený graf z iného experimentu. Počas tréningu sa riešený graf nemenil, iba sa reštartovali dočasné informácie v stave, napríklad vybrané hrany. Experimentov na generalizáciu nebolo robených veľa, ale pri ich priebehu sa pri každom dokončení riešenia vytvoril nový náhodne generovaný graf.

Po každej agentovej akcii prostredie vráti informáciu o novom stave, masku akcií, odmenu, informáciu o skončení riešenia a môže byť odoslaná aj dodatočná informácia používaná hlavne na odladenie agenta (pri experimentoch sa nepoužívala). Nový stav v našich experimentoch predstavuje reprezentáciu grafu s vyznačenými informáciami aktuálneho riešenia (ktoré hrany už boli použité). Aby nedošlo k neoprávneným zmenám v grafe, poskytujeme agentovi iba kópiu tejto reprezentácie. Informácia o skončení riešenia je iba boolovská hodnota true – ak sa ukončilo riešenie a false ak ešte riešenie nie je dokončené.

4.1.1 Reprezentácia a udržiavanie stavu

Keďže sme sa v práci rozhodli používať **dvojkrokový prístup** (viac v kapitole 3.3.1), je potrebné pri riešení udržiavať dve informácie – hrany pridané do riešenia a vrchol vybraný v prvom rozhodovacom kroku, aby sme agentovi posunuli informáciu o vrchole, ktorého incidentné hrany vyberáme. Pri rozhodovacích krokoch teda **upravujeme stav** takto. Po vykonaní druhého rozhodovacieho kroku agentom máme v prostredí informáciu o dvoch vrcholoch medzi ktorými označíme hranu ako vybranú do riešenia. Aby sme sa nemuseli starať o mazanie značky o výbere vrcholu v prvom kroku, môžeme túto značku vyznačovať iba v kópii riešenia posielanej agentovi a v prostredí budeme priamo udržiavať index vrcholu, ktorý využijeme v druhom kroku.

Reprezentácia stavu v prostredí bude teda vyzerať takto:

- Matica vrcholov – tensor o veľkosti *n*2*. V prvom riadku je pre každý vrchol definovaný index od *0* po *n*. Druhý riadok bude obsahovať informáciu o vrchole vybranom v predošlom (prvom) kroku – bude rovný 1 ak daný vrchol bol vybraný v predošlom kroku, 0 inak. V dvojkrokovom prístupe bude po prvom kroku vždy vyznačený iba jeden vrchol. Po druhom kroku budú všetky vrcholy v druhom riadku matice obsahovať hodnotu 0.
- Zoznam hrán – tensor o veľkosti *2*e*. V každom riadku je definovaná jedna hrana dvojicou indexov vrcholov definovaných v prvom riadku matice vrcholov.
- Matica vlastností hrán – tensor o veľkosti *e*3*. V prvom a treťom riadku je definovaná cena hrany. V treťom riadku je udržiavaná neupravovaná cena – využíva ju prostredie na výpočet hodnoty riešenia. V prvom riadku je normalizovaná cena hrany – predelená maximálnou cenou hrany v grafe. Druhý riadok obsahuje informáciu o zaradení hrany do riešenia – *1* ak je hrana zaradená do riešenia, *0* inak.

kde *n* je počet vrcholov a *e* je počet hrán v grafe. Ak bude pri niektorom experimente upravená reprezentácia, bude zmena popísaná tam.

Pri **reštartovaní riešenia** je teda potrebné vymazať iba informáciu o vybraných hranách, to docielime tak, že požiadame novú kópiu grafu od generátora grafov, v ktorom nie sú uložené informácie o riešení.

Ukončenie riešenia nastane práve vtedy, keď je vytvorená kostra riešeného grafu. Vypočíta sa odmena a agentovi sa spolu s odmenou pošle aj informácia o ukončení riešenia. Ukončenie riešenia tak, že agent dosiahne neprípustné riešenie nenastane, pretože akcie, ktoré by viedli k takýmto riešeniam budú maskované.

4.1.2 Určovanie odmien

Pri všetkých experimentoch boli odmeny udeľované na konci riešenia už spomínaným vzorcom $R = -\frac{U_f}{U_{opt}} + 2$, kde U_{opt} je hodnota optimálneho riešenia (keďže riešime exaktnú úlohu vieme ju presne vypočítať) a U_f je hodnota vytvoreného riešenia.

Pri jednom experimente budeme udeľovať odmeny aj podľa toho, či agent rieši úlohu podľa Kruskalovho algoritmu. To znamená, že agent dostane odmenu 1 práve vtedy, keď vyberie hranu, ktorá má najnižšiu cenu z hrán, ktoré môže agent vybrať do kostry.

4.1.3 Maskovanie akcií

Pre určovanie povolených akcií, resp. maskovanie nepovolených akcií, ukladáme v prostredí navyše jednu informáciu, ktorá urýchľuje výpočet masiek. Pri zisťovaní neprípustných akcií používame základy algoritmu union-find. Prostredie obsahuje navyše informáciu o časti grafu, v ktorej sa nachádzajú. S touto informáciou vieme zistiť, či niektorá hrana vytvára cyklus v riešení. Informácia v prostredí nadobúda podobu jednorozmerného poľa *P*, kde každý stĺpec prislúcha jednému z vrcholov v grafe a hodnota v poli určuje časť grafu, ku ktorému vrchol prislúcha. Ak existuje cesta medzi vrcholmi *u* a *v* tak sa hodnoty v poli *P* na indexoch *u* a *v* rovnajú. Z toho vyplýva, že ak by sme chceli pridať hranu z *u* do *v* a hodnoty v poli *P* sa rovnajú, tak by sme v riešení vytvorili cyklus. Túto masku nazývame **maska cyklov**.

Po prvom kroku vytvárame masku pre agenta kombináciou dvoch masiek – masky cyklov a masky, ktorá filtruje vrcholy do ktorých neexistuje hrana, alebo už hrana bola pridaná do riešenia.

Po druhom kroku agentovi poskytneme masku, ktorá povoľuje výber iba vrcholov, ktoré majú incidentné hrany, ktoré môžeme pridať do riešenia (ešte neboli vybrané a po pridaní nevytvoria cyklus). Masku vytvárame zistením všetkých možných prípustných hrán a odmaskujeme vrcholy, s ktorými nie je incidentná žiadna z týchto hrán.

Pri reštartovaní prostredia poskytujeme masku, ktorá neobmedzuje výber akcií.

4.1.4 Generovanie grafov

Pre generovanie grafov sme vytvorili dve metódy – metóda na vytvorenie úplného grafu a metóda na vytvorenie grafu s určitým počtom hrán. Vytvorené grafy budú mať reprezentáciu popísanú vyššie (zloženú z matice vrcholov, zoznamu hrán a matice vlastností hrán).

Vytváranie úplného grafu jednoducho vytvorí potrebné tensory a naplní ich potrebnými informáciami. Priradí indexy vrcholom od 0 po *n*, kde *n* je počet vrcholov vytváraného grafu a je metóde poslaný ako parameter (väčšinou používame *n=10*). Následne zadefinuje všetky hrany. Keďže vytvárame úplný graf, tak vytvorí hrany medzi všetkými vrcholmi grafu. Cena grafu sa určuje pomocou metódy randint z python knižnice random. Ceny hrán sa generujú z intervalu $<V_{min}, V_{max}>$, kde V_{min} a V_{max} sú parametre metódy (väčšinou používame $V_{min}=1$ a $V_{max}=10$). Po vytvorení všetkých hrán sa vypočítajú aj normalizované hodnoty vydelením maximálnou cenou hrany v grafe.

Pri vytváraní grafu s určitým počtom hrán postupujeme pri vrcholoch rovnako. Rozdiel nastáva pri vytváraní hrán. Najskôr vytvoríme základnú kostru (nemusí byť najlacnejšia), tzn. že pridávame hrany medzi vrcholy, medzi ktorými ešte neexistuje cesta. Po vytvorení kostry vytvárame náhodné hrany medzi vrcholy, medzi ktorými ešte nie je hrana až do špecifikovaného počtu. Maximálny počet hrán v grafe určuje parameter metódy. Ceny hrán sa určujú rovnako ako v predošlej metóde a taktiež sa po skončení vytvárania hrán znormalizujú vydelením maximálnej ceny v grafe.

Prostredie pracuje s triedou nazvanou „graph provider“, ktorá sa stará o volanie spomínaných metód a poskytovanie vygenerovaných grafov prostrediu. V triede je možné zafixovať graf – tzn. že trieda nebude generovať nové grafy ale poskytovať iba ten zafixovaný. Taktiež je možné požiadať o posledný vygenerovaný graf. Pri každej požiadavke o graf sa vytvorí kópia grafu, ktorý sa následne pošle prostrediu. Kópia sa vytvára preto, aby sme daný graf vedeli poskytnúť aj viackrát alebo viacerým prostrediam naraz.

V experimentoch budeme používať niekedy označenie G*x* (napr.: G10), čo určuje použitý graf uložený a prístupný na githube. [21]

4.2 Agent

Pri experimentoch používame implementáciu agenta z githubu. [22] Táto implementácia spĺňa väčšinu požiadaviek z návrhu riešenia a navyše urýchľuje tréning paralelným vykonávaním akcií na niekoľkých prostrediach naraz, čo urýchľuje vytváranie skúseností.

Danú implementáciu sme upravili, aby bola schopná pracovať s grafovými reprezentáciami, taktiež sme pridali ukladanie vypočítanej korelácie medzi logitmi a stupňami vrcholov.

Počas experimentov budeme upravovať tieto parametre agenta:

- Epsilon – medzi hodnotami 0.1, 0.2 a 0.3
- Coef_entropy – medzi hodnotami 0.001 a 0.002
- Learning rate – hodnoty z intervalu $<2.5e^{-4};2.5e^{-3}>$

4.3 Sieť

Vo väčšine experimentov používame popísanú grafovú neurónovú sieť GATConv z knižnice „pytorch_geometric“.

Prvá vrstva siete má počet vstupných kanálov rovný počtu vlastností vrcholov (vo väčšine experimentov rovný 2) a počet výstupných kanálov nastavený na 16. Nasledujúcich *x-1* vrstiev má počet vstupných aj výstupných kanálov rovný 16, kde *x* je nastaviteľná hodnota siete od 1 po 6 (najčastejšie používame hodnotu 3).

Sieť sa následne rozdelí na dve hlavy:

- Výsledky z prvej hlavy sú používané na výber akcií. Skladá sa z *p* GATConv vrstiev, kde *p* je nastaviteľná hodnota od 1 do 4 (v experimentoch používame hodnoty 1 a 2). Všetky vrstvy majú vstupný počet kanálov 16 a všetky vrstvy okrem poslednej aj výstupný počet kanálov 16. Posledná výstupná vrstva má výstupný iba jeden kanál.
- Druhá hlava sa používa na určovanie výhod. Pozostáva z *y* plne prepojených vrstiev (angl.: Linear), kde *y* je nastaviteľný parameter s hodnotou 1 alebo 2 (vo väčšine experimentov je nastavený na 2). Výsledkom tejto hlavy je jedna hodnota určujúca odhadovanú výhodu.

Akékoľvek úpravy vykonané pri špecifických experimentoch budú popísané pri daných experimentoch.

V experimentoch budeme niekedy používať značenie *x-p*GAT (napr.: 3-1GAT), čo znamená, že bola použitá sieť s *x* hlavnými vrstvami a *p* vrstvami v prvej hlave.

4.4 Tréning

Pri experimentoch sme trénovali agentov na 1000 iteráciách. V každej iterácii bolo vykonaných od 4 po 10 epoch s batch size hodnotou v počiatočných experimentoch 64

a pri neskorších experimentoch vypočítaná podľa veľkosti a počtu vlastností grafu (väčšinou hodnota 576).

Pri každom riešení grafu je vykonaných s krokov (step amount), kde s je hodnota vypočítaná pomocou rovnice $s = (n - 1) * 2$, kde n je počet vrcholov grafu.

Najčastejšie používané parametre grafu pri experimentoch sú:

- Počet vrcholov = 10
- Počet hrán – je hodnota vypočítaná z počtu hrán úplného grafu s daným počtom vrcholov vynásobená hodnotou 0.75

Ak budú pri niektorom z experimentov spomínané hodnoty upravené, tak to bude popísané pri danom experimente.

4.5 Testovanie

Na testovanie agentov sme si vytvorili triedu, ktorá je schopná vykonať vybraný počet testov na niekoľkých agentoch a následne vypísať výsledky alebo ich vyobraziť v stĺpcovom grafe.

Pri testovaní budeme natrénovaných agentov porovnávať s niekoľkými inými agentami a hodnotami. Prvá porovnávacia hodnota je hodnota optimálneho riešenia, keďže riešime exaktnú úlohu, tak ju vieme vypočítať presne. Následne budeme testovaných agentov porovnávať s dvomi náhodnými agentami. Prvý náhodný agent (Rnd výber) vyberá akcie zo zoznamu prípustných akcií pomocou funkcie random.choice z knižnice numpy. Druhý náhodný agent (Rnd sieť) vyberá akcie pomocou výsledkov z náhodne inicializovanej siete. Porovnania s týmito náhodnými agentmi nám ukážu, či sa testovaný agent niečo natrénoval alebo vytvára iba náhodné riešenia.

Pri niektorých experimentoch budeme porovnávať aj niektorých agentov medzi sebou. Ak napríklad trénujeme agenta s nejakou zmenou, tak ho porovnáme aj s agentom natrénovaným bez danej zmeny aby sme vedeli povedať, či daná zmena pomohla alebo nie.

5 POPIS EXPERIMENTOV

V tejto kapitole sa budeme venovať samotným experimentom vykonaným v tejto práci. Pri každom experimente bude popísaný úvod do experimentu – dôvod prečo sme daný experiment vykonali a dôležité zmeny vykonané v experimente. Následne budú popísané a zobrazené výsledky z daného experimentu. Zhrnutie hlavných výsledkov z experimentov bude zhrnuté v kapitole 7.

5.1 Počiatočné experimenty

Spočiatku boli experimenty hlavne zamerané na to aby sa agent natrénoval aspoň na jednom statickom grafe. Skúšali sme hlavne upravovať počet vrstiev v hlavnej časti sieti od 2 po 6 GATConv vrstiev. Následne sme upravovali parametre epsilon (0.1-0.3), learning rate (0.00005-0.0025).

Teraz pri väčšine experimentov používame takéto parametre:

- Epsilon = 0.3
- Lr = 0.00025
- počet hlavných GATConv vrstiev = 3

Pri počiatočných experimentoch sme vyskúšali aj vylepšenie vrstvy GATConv nazývanú GATv2Conv, ale výsledky neukázali, že by zlepšovala schopnosť agentov riešiť úlohy.

V implementácii vrstvy GATConv sa používa aj parameter edge_dim, ktorý určuje počet vlastností na hranách. Popri niektorých experimentoch sme vykonali aj testy, či zmena tohto parametru mení aj výsledky experimentov. Nastavovali sme parameter na hodnoty 2 alebo „None“ a z našich experimentov vyšlo, že to nepomáha agentom riešiť úlohy. Vo väčšine experimentov bol daný parameter nastavený na „None“.

Pri niektorých experimentoch boli vykonané pokusy o natrénovanie agentov na väčších úlohách, pri ktorých ale nedosahovali lepšie výsledky.

Taktiež sme skúšali experimenty na testovanie generalizácie, ale taktiež agenti neboli schopný riešiť úlohy.

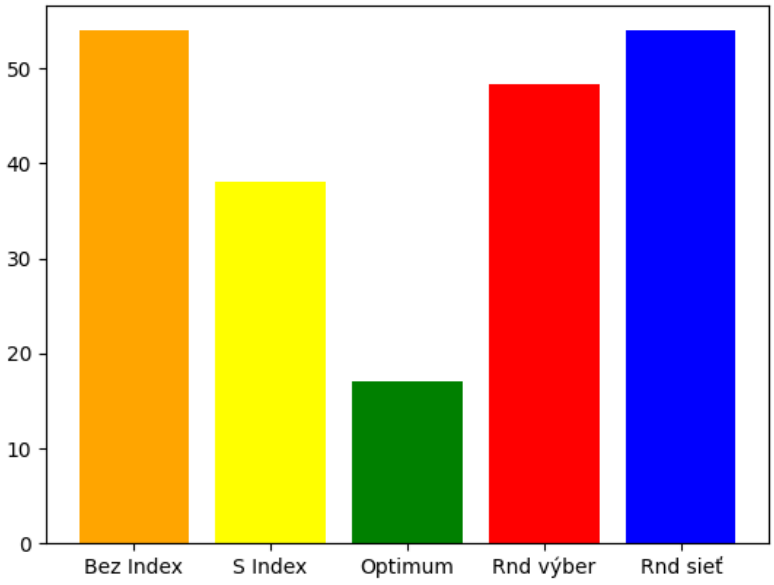
5.2 Odstránenie indexov z vrcholov

Z našich skúseností agenti pri tréningu statických úloh ignorujú statické informácie poskytované z prostredia. Chceli sme teda vyskúšať, či vyhodením statickej informácie z vrcholov dosiahneme lepšie výsledky.

Pri tréningu sme znížili počet vlastností vo vrcholoch z 2 na 1, takže vo vlastnostiach vrcholov je iba informácia, či bol daný vrchol vybraný v predošlom kroku alebo nie.

Výsledky:

Vo väčšine tréningoch sa nedokázal natrénovať ani jeden z agentov – dávali výsledky podobné náhodným agentom a pri niektorých tréningoch sa natrénoval niečo iba agent s informáciou o indexoch vo vrchole (Obrázok 9). Preto tvrdíme, že odstránenie indexov z vrcholu nepomohlo pri tréningoch a pri ďalších experimentoch budeme pokračovať s indexami vo vrcholoch.



Obrázok 9 Porovnanie agentov s a bez odstránenia indexov

5.3 Pridanie súradníc do vrcholov

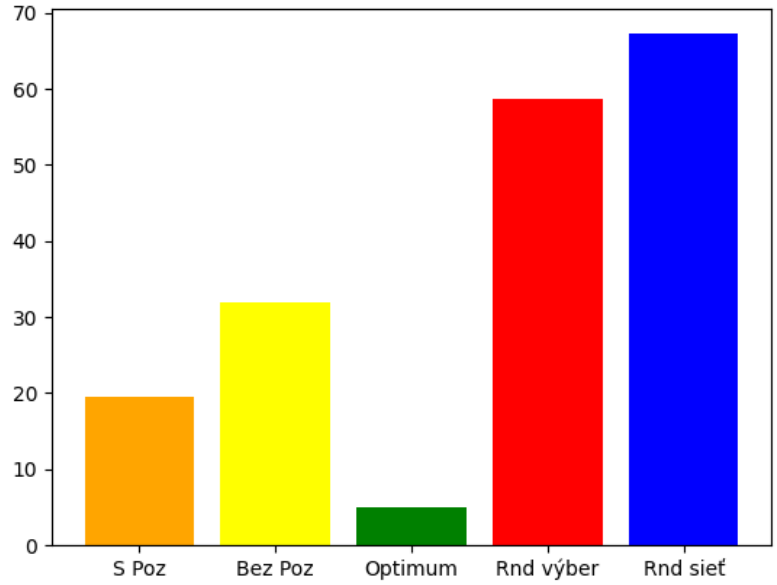
Dôvodom na vyskúšanie tohto experimentu, bol nápad pridať agentovi viac informácií, z ktorých môže zistiť vzdialenosť (cenu) hrán aj z iných informácií.

Pridali sme dve nové vlastnosti vrcholov súradnice *x* a *y*. Následne sme ceny hrán vypočítali Euclidovú vzdialenosť pomocou rovnice $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, kde x_1 a y_1 sú súradnice prvého vrcholu a x_2 a y_2 súradnice druhého vrcholu.

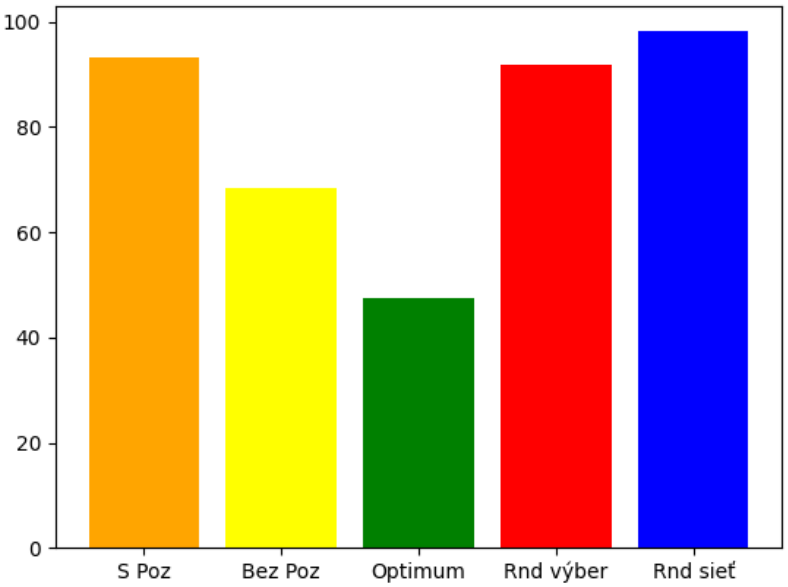
Súradnice boli generované ako celé čísla z intervalu <-10, 10>.

Výsledky:

Po vyhodnotení tréningov sme zistili, že sa agenti buď nič nenaučili, alebo ak sa niečo naučili tak boli výsledky jedného z agentov lepšie ako druhého. Bohužiaľ ani jeden s agentov nedokázal konzistentne poraziť druhého, takže nevieme presne určiť, či pridanie súradníc pomáha pri tréningu (Obrázok 10 a Obrázok 11).



Obrázok 10 Porovnanie agentov s a bez pozícii vo vrcholoch na grafe G32



Obrázok 11 Porovnanie agentov s a bez pozícii vo vrcholoch na grafe G33

5.4 Pozorovanie EDB

V predošlých experimentoch sme si všimli, že pri testovaní sa agenti zameriavajú na vybraný vrchol a vyberajú hlavne hrany z jeho okolia. Pri niektorých experimentoch to boli aj vrcholy s najväčším stupňom. Taktiež sme zistili, že schopnosť natrénovaného agenta riešiť úlohu najlacnejšej kostry na grafe záleží aj od štruktúry grafu.

Príklad 1:

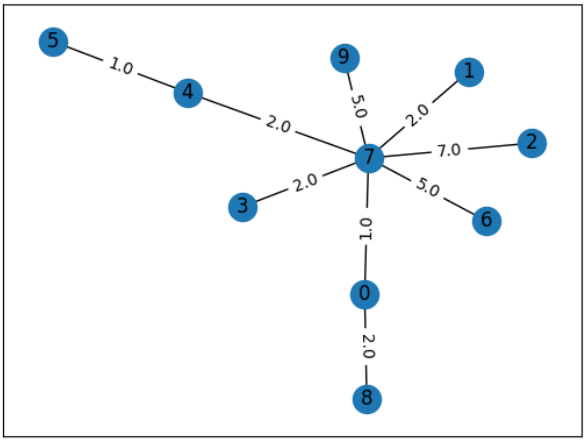
V tomto príklade môžeme vidieť, že agent sa naučil vyberať hrany z okolia vrcholu 7 (Obrázok 12) aj napriek tomu, že hrana (7, 2) má cenu 7 a z optimálneho riešenia (Obrázok 13) vidíme, že je lepšie spojiť vrchol 2 s vrcholov 8. Agent dosiahol cenu kostry 27 a optimálne riešenie je 14.

Tabuľka 2 Súčet hodnôt incidentných hrán s daným vrcholom v grafe v príklade 1

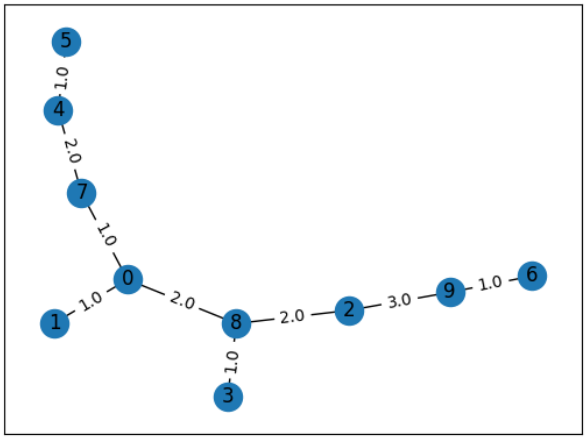
Index	5	7	4	0	3	8	2	1	9	6
Súčet	17	24	26	30	32	32	38	42	42	57

Tabuľka 3 Stupeň vrcholov v grafe v príklade 1

Index	5	4	0	7	8	1	2	3	9	6
Stupeň	4	6	7	7	7	8	8	8	8	9



Obrázok 12 Vytvorené riešenie na grafe v príklade 1



Obrázok 13 Najlacnejšia kostra na grafe v príklade 1

Príklad 2:

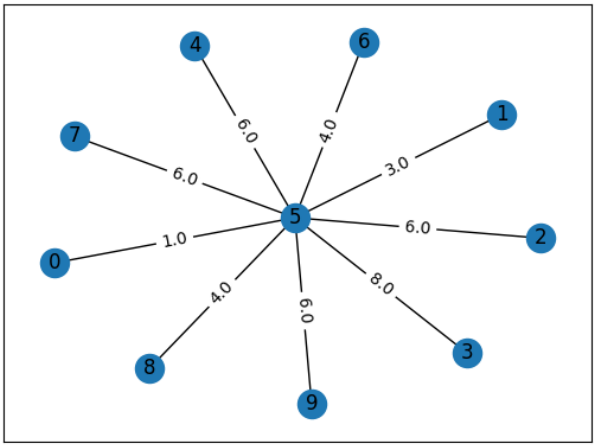
V tomto príklade vidíme podobný jav s tým rozdielom, že agent si vybral vrchol 5 (Obrázok 14), ktorý je zároveň aj vrchol s najväčším stupňom (Tabuľka 5). Z obrázka najlacnejšej kostry (Obrázok 15) môžeme vidieť, že agent vytvoril riešenie (s hodnotou 44) vzdialené od optimálneho (s hodnotou 32).

Tabuľka 4 Súčet okolia vrcholov v grafe v príklade 2

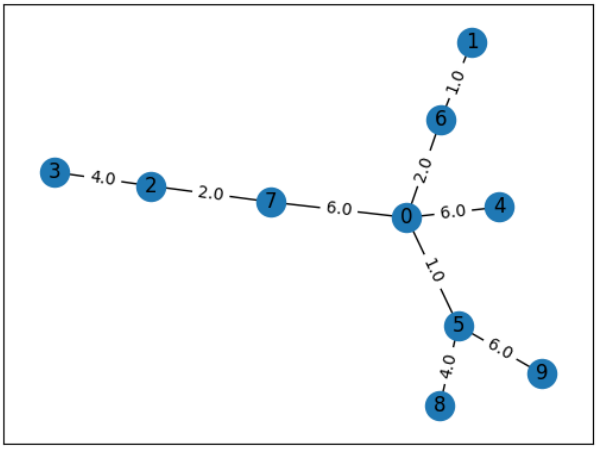
Index	8	2	6	5	1	3	7	4	0	9
Súčet	34	35	40	44	49	54	54	55	56	59

Tabuľka 5 Stupeň vrcholov v grafe v príklade 2

Index	8	2	1	3	4	6	9	7	0	5
Stupeň	5	6	7	7	7	7	7	8	9	9



Obrázok 14 Vytvorené riešenie na grafe v príklade 2

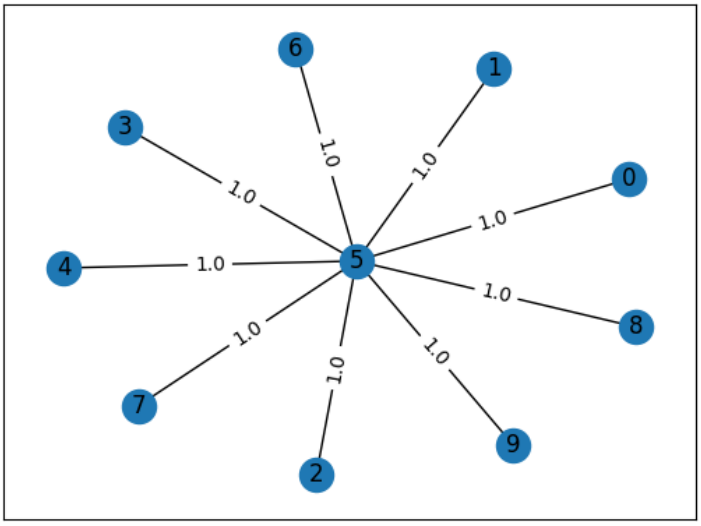


Obrázok 15 Najlacnejšia kostra na grafe v príklade 2

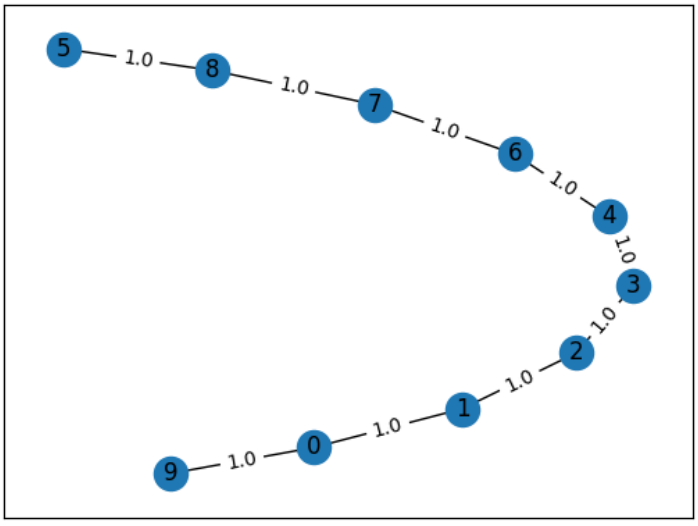
5.4.1 Vlastné grafy

Po zistení vyššie popísaného pozorovania sme sa rozhodli skúmať, či túto vlastnosť vieme posilniť alebo zoslabiť.

Na otestovanie sme vytvorili dva ručne zostavené grafy. Prvý graf bol navrhnutý tak, aby riešenie vyžadovalo výber vrcholov s vysokým stupňom — tento graf nazývame **EDB+ graf**, pretože vykazuje pozitívnu synergiu s metódou EDB. Druhý graf, nazývaný **EDB- graf**, bol vytvorený tak, aby optimálne riešenie vyžadovalo výber maximálne dvoch hrán z každého vrcholu, pričom vrchol s najvyšším stupňom vedie k najhoršiemu riešeniu, pretože v jeho okolí sa nachádzajú hrany s najvyššími cenami..



Obrázok 16 Najlacnejšia kostra EDB+ grafu (optimálne riešenie)



Obrázok 17 Najlacnejšia kostra EDB- grafu (optimálne riešenie)

EDB+ graf G11:

V tabuľke (Tabuľka 6) sú napísané súčty cien hrán vychádzajúcich z vrcholov zoradený od najmenšej po najväčšiu, prvý riadok je index vrcholu a druhý riadok je súčet cien. V tabuľke (Tabuľka 7) sú zobrazené stupne vrcholov zoradené od najmenšieho po najväčší a v obrázku (Obrázok 16) je zobrazená najlacnejšia kostra grafu.

Z obrázku môžeme vidieť, že najlacnejšia kostra je centrovaná na vrchol s indexom 5, ktorý má najväčší stupeň zo všetkých vrcholov grafu a má najnižší súčet cien hrán incidentných s týmto vrcholom.

Tabuľka 6 Súčty cien hrán v EDB+ grafe

Index	5	8	7	9	0	1	3	2	4	6
Súčet	9	20	25	26	26	32	32	35	36	37

Tabuľka 7 Stupne vrcholov v EDB+ grafe

Index	0	1	2	7	8	9	3	4	6	5
Stupeň	6	6	6	6	6	6	7	7	7	9

EDB- graf G10:

V tabuľke (Tabuľka 8) sú napísané súčty cien hrán vychádzajúcich z vrcholov zoradený od najmenšej po najväčšiu, prvý riadok je index vrcholu a druhý riadok je súčet cien. V tabuľke (Tabuľka 9) sú zobrazené stupne vrcholov zoradené od najmenšieho po najväčší a v obrázku (Obrázok 17) je zobrazená najlacnejšia kostra grafu.

Z obrázku môžeme vidieť, že najlacnejšia kostra musí byť tvorená presne proti zistenému pozorovaniu – nemôže vyberať iba okolie jedného vrcholu.

Tabuľka 8 Súčty cien hrán v EDB- grafe

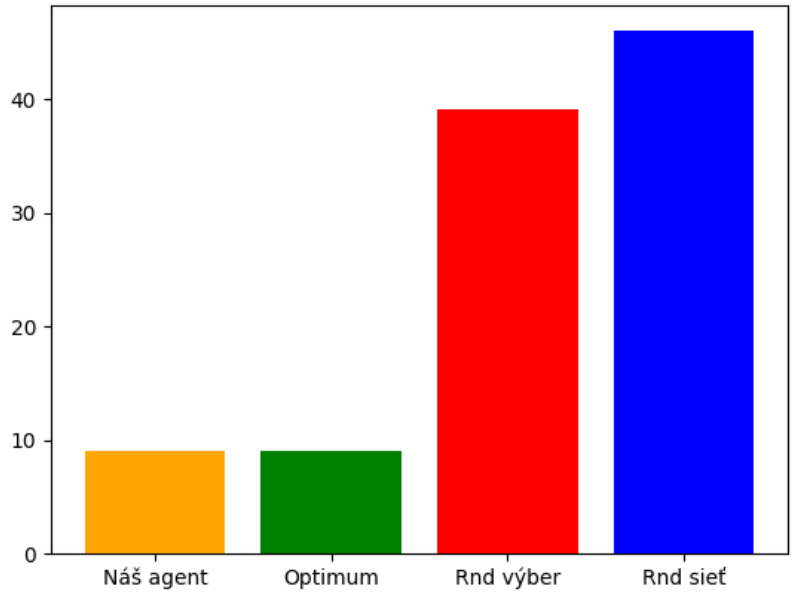
Index	8	0	9	7	2	3	1	6	4	5
Súčet	16	24	25	27	30	31	32	32	36	81

Tabuľka 9 Stupne vrcholov v EDB- grafe

Index	0	1	2	7	8	9	3	4	6	5
Stupeň	6	6	6	6	6	6	7	7	7	9

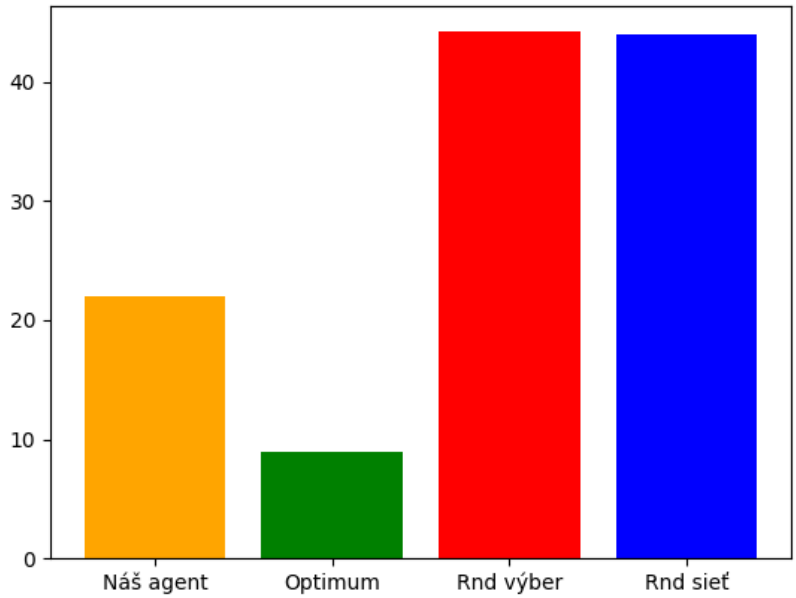
Výsledky:

Bolo vykonaných niekoľko tréningov s danými grafmi, pri niektorých sa agenti stále nič nenatrénovali. Keď sa agent natrénoval tak pri EDB+ grafe dosiahol optimálne riešenie ako sme očakávali. Na obrázku (Obrázok 18) je zobrazené porovnanie jedného z agentov, ktorý sa niečo natrénoval.

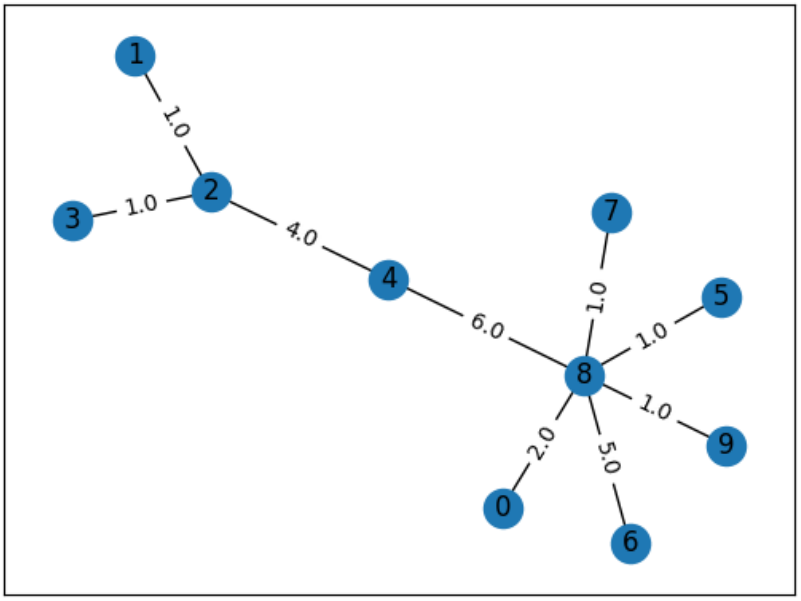


Obrázok 18 Porovnanie výsledkov na EDB+ grafe

Pri EDB- grafe sa dokázal natrénovať riešenie úlohy podobným spôsobom ako bol spomínaný spomínaným v kapitole 5.4. V obrázku môžeme vidieť, že sa agent naučil vyberať okolie vrcholu 8, ktoré má nízky stupeň a zároveň má aj najnižší súčet cien v okolí.



Obrázok 19 Porovnanie výsledkov na EDB- grafe



Obrázok 20 Riešenie vytvorené agentom na EDV- grafe

5.4.2 Záver

Z doposiaľ vykonaných experimentov sme zistili, že agenti preferujú výber vrcholov s vysokým stupňom. Návrhom vlastných grafov sme si chceli toto tvrdenie overiť a z výsledkov predošlého experimentu sme zistili, že keď vytvoríme graf, ktorý podporuje túto vlastnosť (EDB+ graf), tak sa agenti dokážu naučiť aj vytvárať optimálne riešenie.

Z výsledkov na EDB- grafe, sme zistili, že agenti sa avšak dokážu naučiť ignorovať vrcholy s najväčším stupňom, ale zamerajú sa na vrchol s najnižším súčtom cien hrán v okolí. Myslíme si, že to je preto, lebo pri tréningu sa otočí znamienko váh a tým pádom sa bude dávať väčší dôraz na vrcholy s najmenším stupňom a súčtom okolia.

Vyplýva nám z toho, že EDB má vplyv počas tréningu a pri riešení grafu, pri ktorom je potrebné potlačiť EDB, agenti zareagujú iba zmenou vyberaného vrcholu, čo vo väčšine grafoch nie je vhodná stratégia.

5.5 Nové architektúry siete

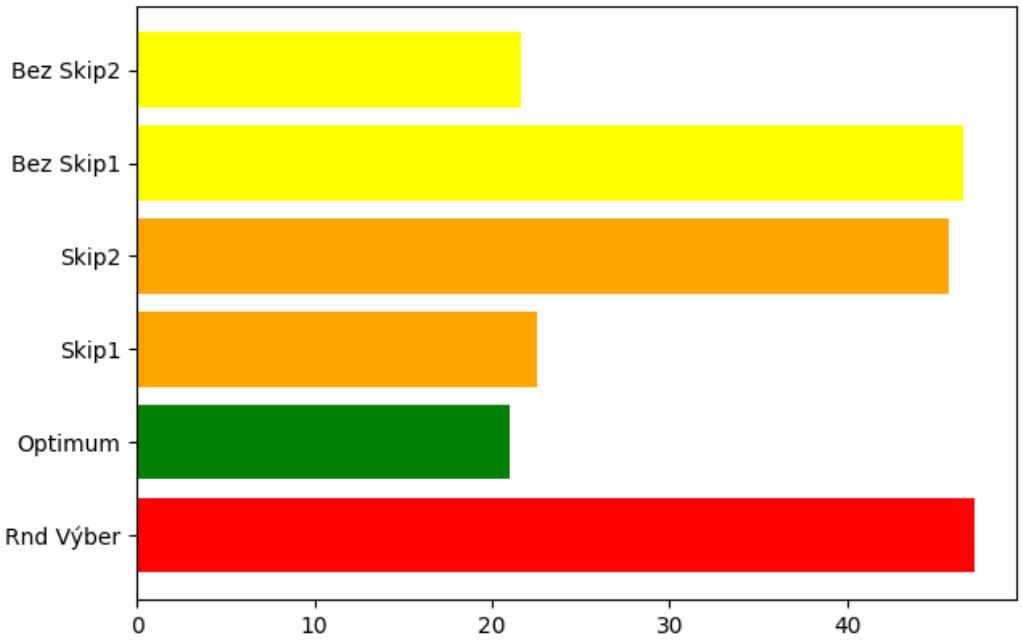
Po predošlých zisteniach a taktiež problému, že niekedy sa agenti nič nenaučia, sme sa rozhodli pokúsiť sa zmierniť vplyv EDB na tréning. Vyskúšali sme zmeniť architektúru siete dvoma rôznymi spôsobmi. Pri prvom experimente sme upravili doposiaľ používanú architektúru a pri druhom experimente sme vyskúšali úplne iný návrh siete.

5.5.1 Skip connection

Pri vykonávaní tohto experimentu sme agentovi v každej vrstve siete pripojili aj pôvodné informácie o vlastnostiach vrcholov (vykonávali sme tzv. preskakovanie spojení v sieti – angl.: skip connection), aby sme zosilnili informáciu o vlastnostiach samotných vrcholov.

Výsledky:

Z výsledkov nevieme presne určiť, či pridanie preskakovania pomáha prekonávať vplyv EDB. Pri niektorých tréningoch sa žiadny agent nič nenatrénoval (vytvára riešenia s rovnakými cenami ako náhodný agent). Pri niektorých tréningoch porazil agent s preskakovaním agenta bez preskakovania a pri niektorých naopak. Je teda ťažké povedať, či preskakovanie pomáha, keďže väčšina tréningov neposkytuje výsledky, z ktorých by sme to vedeli povedať.



Obrázok 21 Porovnanie vybraných agentov s a bez použitia preskočenia (skip)

5.5.2 FRI architektúra siete

V tomto experimente išlo o podobnú myšlienku ako v predošlom experimente. Použili sme na tréning sieť navrhnutú vo výskumnom tíme FRI (Fakulty Riadenia a Informatiky).

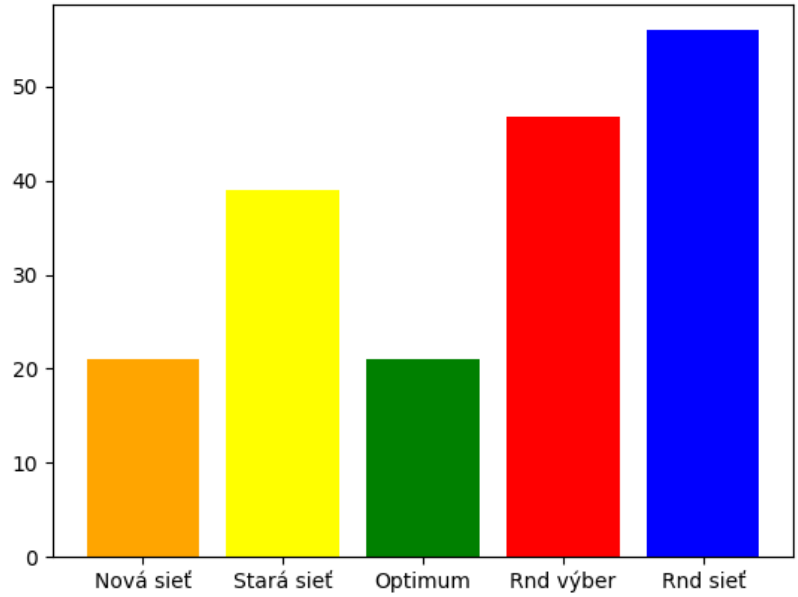
Sieť obsahuje 4 GATConv vrstvy a pri každej GATConv vrstve používa ešte jednu plne prepojenú vrstvu. Následne sa snaží vytvoriť informáciu o celkovom stave grafu a nakoniec pomocou ďalších plne prepojených vrstiev vytvorí logity pre výber akcie a taktiež hodnotu výhod.

Táto sieť si ale kvôli využitiu plne prepojených vrstiev vyžaduje, aby všetky riešené úlohy boli rovnakej veľkosti, čo nie je pri aktuálnych experimentoch problém, ale pri testovaní generalizácie by nebola možnosť natrénovať agenta, ktorý by bol schopný generalizovať nad rôznymi veľkosťami grafu.

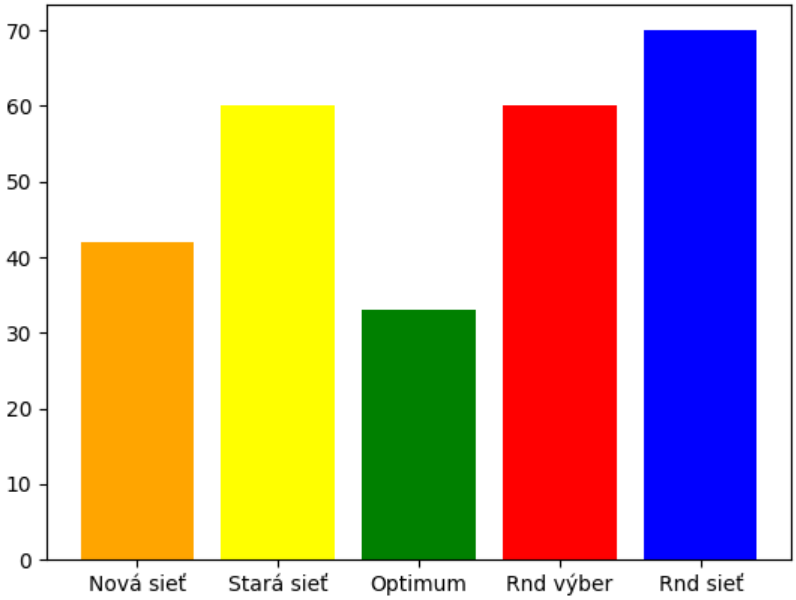
Výsledky:

Z výsledkov môžeme vidieť, že nová architektúra siete dáva lepšie výsledky ako pôvodná. Na grafe G24 sa jej dokonca podarilo postup optimálneho riešenia.

Po vykonaní týchto experimentov si myslíme, že EDB nie je jediná vlastnosť grafových neurónových sietí, ktorá bráni dosiahnutiu optimálnych riešení pri úlohe hľadania najlacnejšej kostry. To sme sa rozhodli otestovať v nasledujúcich experimentoch úpravou odmien a porovnaním s plne prepojenou sieťou.



Obrázok 22 Porovnanie výsledkov novej a starej siete na grafe G24



Obrázok 23 Porovnanie výsledkov novej a starej siete na grafe G22

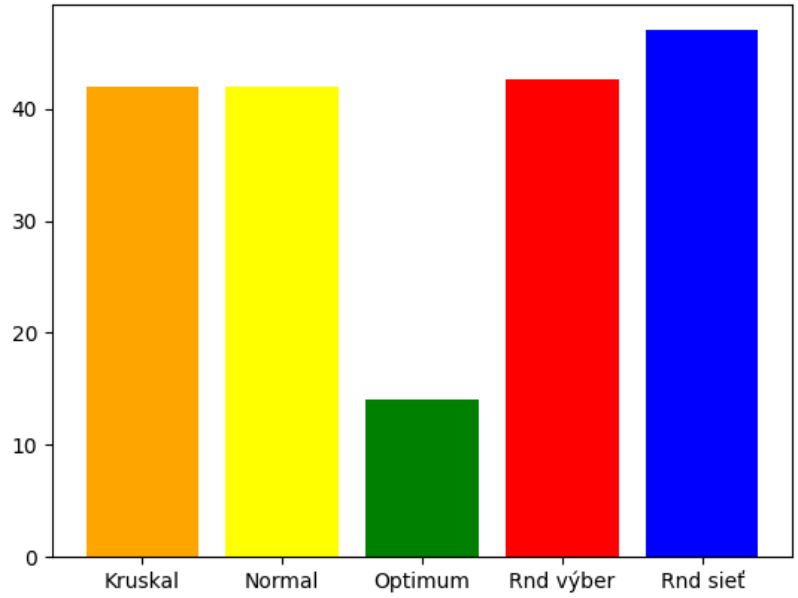
5.6 Trénovanie Kruskalovho algoritmu

Rozhodli sme sa vyskúšať, či sa agenti dokážu naučiť riešiť úlohu pomocou Kruskalovho algoritmu.

Ak agent pri tréningu vyberie hranu podľa postupu v algoritme – vyberie najlacnejšiu hranu, ktorá nevytvára cyklus s hranami vo vytváranom riešení, tak dostane odmenu *1*, inak dostane odmenu 0. V poslednom kroku riešenia stále dostáva odmenu podľa navrhnutého riešenia.

Výsledky:

Z výsledkov sme zistili, že pridané odmeny nepomáhajú agentom riešiť úlohu – dosahujú podobné výsledky ako bez pridaných odmien.



Obrázok 24 Porovnanie agentov trénovaných s a bez pridaných odmien

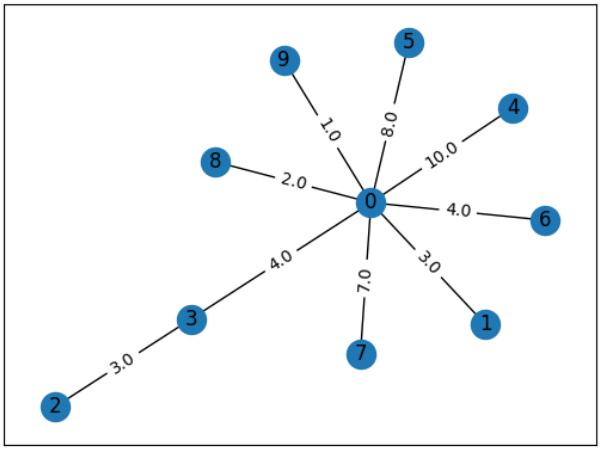
Taktiež môžeme vidieť už spomínané pozorovanie z kapitoly 5.4. Agent sa natrénoval tak, že vyberá okolie vrcholu 0 (Obrázok 25), ktorý je vrchol s najväčším stupňom v grafe a aj s vysokým súčtom okolia (Tabuľka 10 a Tabuľka 11). Optimálne riešenie má hodnotu 14 a vytvorené riešenie hodnotu 42.

Tabuľka 10 Súčty cien hrán v grafe G42

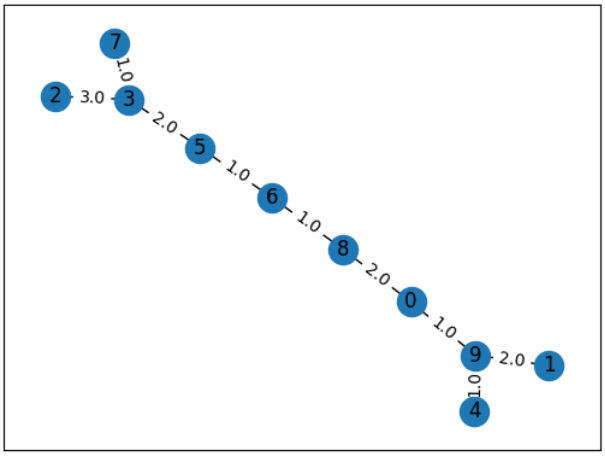
Index	1	3	9	6	8	7	5	2	0	4
Súčet	18	23	28	29	29	31	36	37	39	44

Tabuľka 11 Stupne vrcholov v grafe G42

Index	1	2	3	7	8	6	9	0	4	5
Stupeň	4	6	6	6	6	7	7	8	8	8



Obrázok 25 Vytvorené riešenie na grafe G42



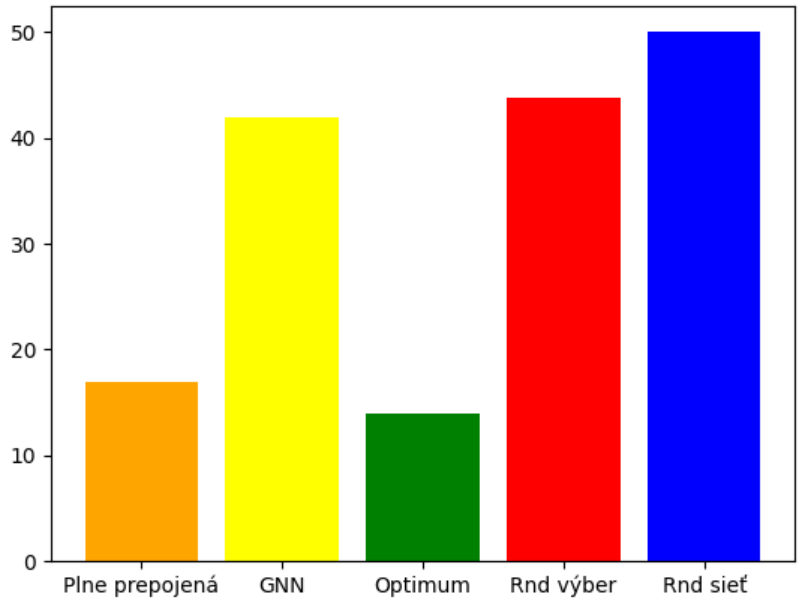
Obrázok 26 Najlacnejšia kostra v grafe G42

5.7 Porovnanie s plne prepojenou sieťou

V tomto experimente sme chceli porovnať výsledky tréningu vykonaného s grafovými neurónovými sieťami s výsledkami tréningu s plne prepojenými sieťami. Chceli sme zistiť, či sú plne prepojené siete schopné sa naučiť riešiť úlohu hľadania najlacnejšej kostry s odmenami ako pri experimente 5.6 alebo aj bez nich. Ak sa plne prepojené siete dokážu naučiť riešiť úlohu, znamená to, že problém bude v GNN. Myslíme si, že GNN nie sú schopné sa zamerať exaktne na jednu zmenu informácie práve na jednom mieste, čo si úloha hľadania najlacnejšej kostry vyžaduje.

Výsledky:

Pri testovaní natrénovaných agentov s plne prepojenou sieťou a s odmenami podľa Kruskalovho algoritmu sme zistili, že plne prepojené siete sú schopnejšie riešiť úlohu hľadania najlacnejšej kostry ako GNN.

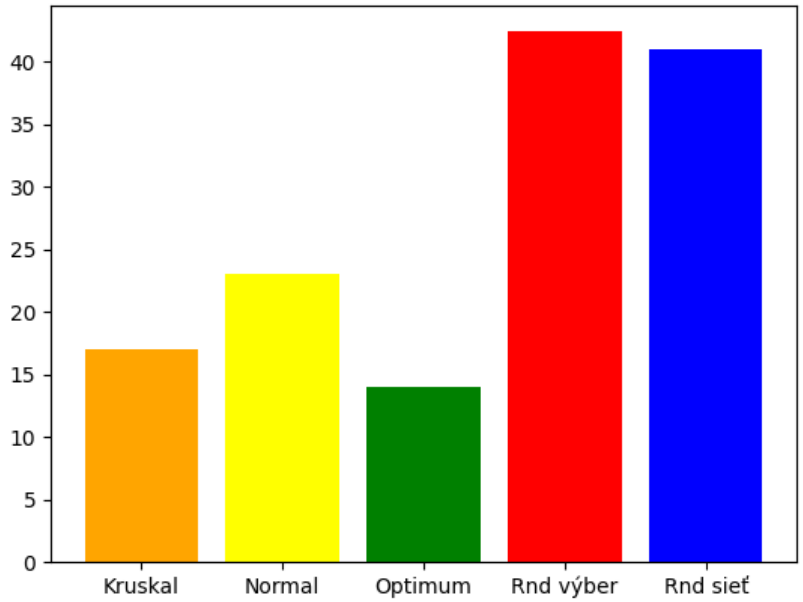


Obrázok 27 Porovnanie agentov s plne prepojenou sieťou a s GNN na grafe G42

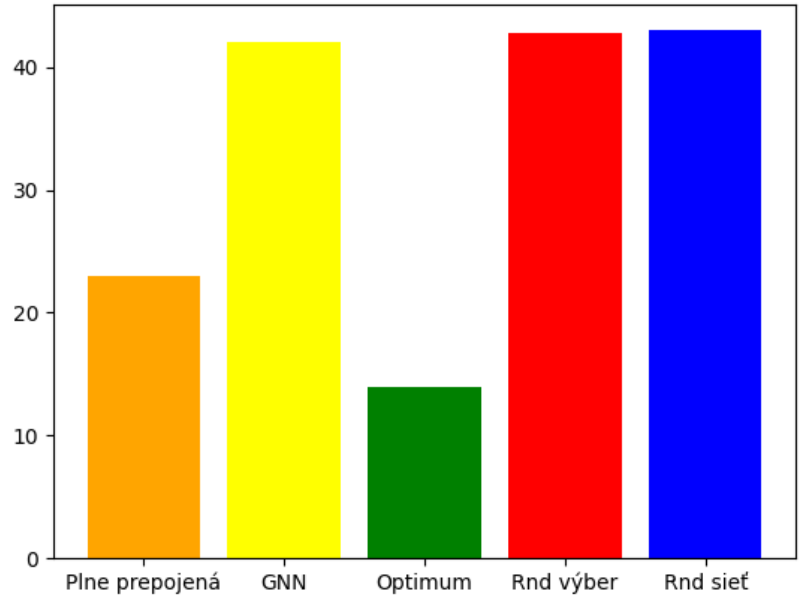
Taktiež sme otestovali, či sú plne prepojené siete schopné sa naučiť hľadať najlacnejšiu kostru aj bez odmien podľa Kruskalovho algoritmu.

Z tréningov nám vyšlo, že tieto siete sú toho schopnejšie ako GNN a znamená to, že je potrebné ďalší výskum ohľadom schopností GNN sa naučiť riešiť danú úlohu.

V prvom obrázku (Obrázok 28), je porovnanie dvoch agentov s plne prepojenými sieťami. Jeden z agentov bol trénovaný s odmenami podľa Kruskalovho algoritmu (Kruskal) a druhý bez (Normal). V druhom obrázku (Obrázok 29) je porovnanie agenta s plne prepojenou sieťou a agenta s GNN, kde obidvaja boli trénovaný bez pridaných odmien.



Obrázok 28 Porovnanie plne prepojených sietí s a bez pridaných odmien



Obrázok 29 Porovnanie plne prepojenej siete a GNN bez pridaných odmien

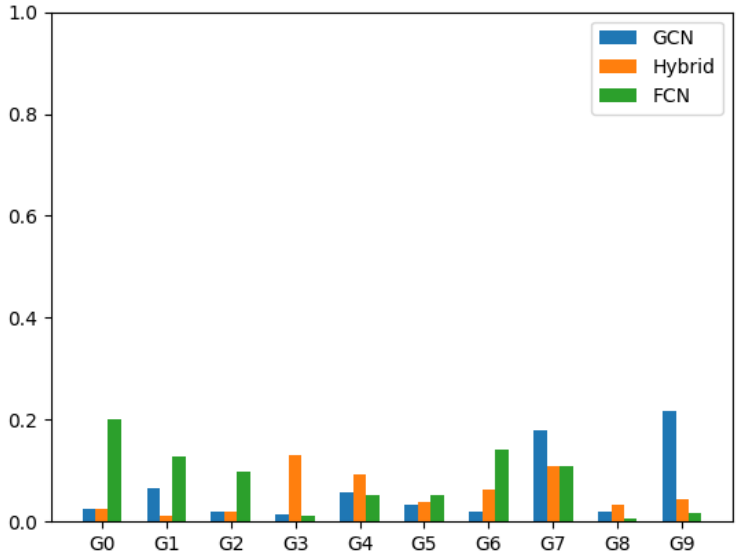
5.8 Sledovanie korelácie

V poslednom experimente práce sme chceli pozorovať koreláciu medzi logitmi a stupňom vrcholu počas tréningu a taktiež pri náhodne inicializovaných sieťach. Tým zistíme, či agenti musia už pri začatí tréningu prekonávať EDB. Testovanie prebieha na náhodne generovaných grafoch s počtom vrcholov 100 a počtom hrán 3712 (grafy G0 – G9 neboli ukladané – nie sú dostupné na githube [21]).

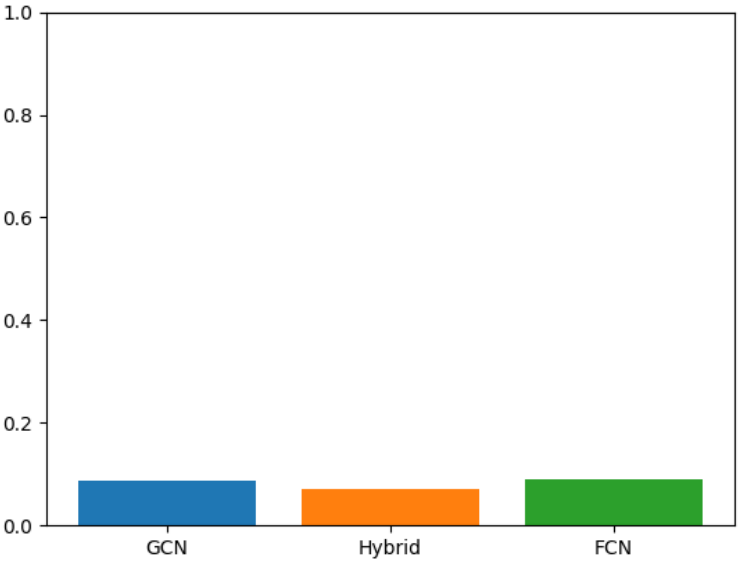
Taktiež sme chceli skúsiť pozorovať koreláciu počas tréningu medzi stupňom vrcholu a počtom, koľko krát bol daný vrchol vybraný. V každej iterácii je vykonávaných niekoľko riešení, pri všetkých riešenia v iterácii je pre každý vrchol spočítaná hodnota, koľko-krát bol počas riešenia označený. Pred vykonaním kroku učenia sme vypočítali koreláciu medzi stupňom vrcholu a vypočítanými súčtami.

Výsledky:

Pri zisťovaní korelácie v náhodne inicializovaných sieťach nám vyšlo, že korelácia pri väčšine grafov nepresiahne hodnotu 0.2 (Obrázok 30) a výsledky z našej grafovej siete (GCN) sú veľmi podobné plne prepojenej sieti (FCN) aj sieti z experimentu 5.5.2 (FRI). Priemerné hodnoty korelácií sú okolo 0.1 (Obrázok 31).

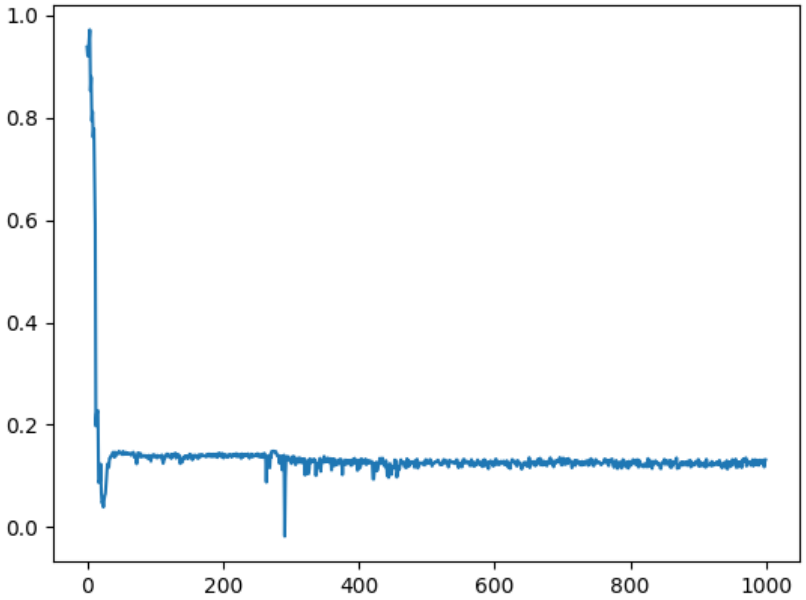


Obrázok 30 Porovnanie korelácií pri náhodne inicializovaných sieťach

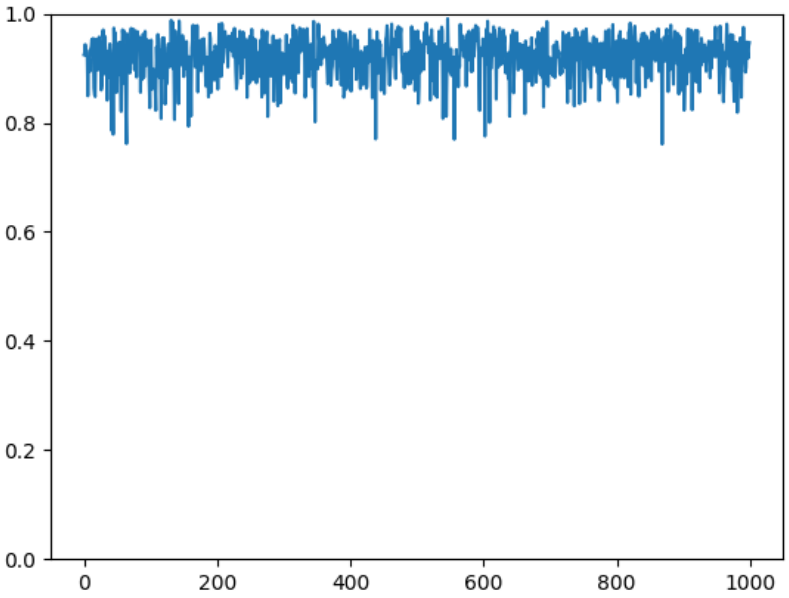


Obrázok 31 Priemerná korelácia pri náhodne inicializovaných sieťach

Pri zisťovaní korelácie medzi počtom výberov vrcholu a jeho stupňom, sme si všimli jednu zaujímavosť. Keď agent po tréningu dáva lepšie výsledky ako náhodný agent, tak v grafe korelácie (Obrázok 32) môžeme vidieť, že sa znížila počas tréningu a ustálila. Na druhej strane, keď sa agent nič nenaučí (dáva výsledky ako náhodný agent), tak z grafu (Obrázok 33) to vyzerá tak, že s EDB celý tréning bojoval a nedokázal ho poraziť.



Obrázok 32 Korelácia počas úspešného tréningu



Obrázok 33 Korelácia počas neúspešného tréningu

6 VYHODNOTENIE

Z vykonaných experimentov sme zistili a ukázali, že pri tréningu PPO agentov učením posilňovaním s použitím grafových neurónových sietí je v niektorých prípadoch výrazný vplyv EDB na tréning. Taktiež sme ale zistili, že výsledok tréningu veľmi závisí aj od štruktúry grafu, na ktorom je tréning vykonávaný.

V experimente 5.4.1 sme zistili, že vplyv EDB môžeme podporiť. Experiment nám taktiež ukázal, že sa agenti dokážu odnaučiť vyberať hrany incidentné s vrcholom s najvyšším stupňom a namiesto toho vyberať hrany incidentné s vrcholom s najnižším súčtom okolia. Myslíme si, že vhodným návrhom ďalších grafov sa môžu odhaliť nové vlastnosti grafov, ktoré budú podporovať resp. brániť tréningu.

Pomocou experimentu 5.6 sme zistili, že agenti s GNN sa nedokážu namemorovať výber kostry aj keď tento výber podporíme odmenou.

Pri porovnaní GNN s plne prepojenými sieťami sa ukázali tie plne prepojené ako schopnejšie riešiť vybranú úlohu. Myslíme si, že GNN v porovnaní s plne prepojenými sieťami nemajú schopnosť priamo zareagovať na jednu malú zmenu (napr.: značka, že hrana je už súčasťou riešenia), čo si úloha hľadania najlacnejšej kostry vyžaduje. Toto tvrdenie by bolo vhodné do budúcnosti viac preskúmať, napr. využitím metód vysvetliteľnosti.

Ako nám ukázali výsledky z experimentu, kde sme pracovali s FRI architektúrou siete (5.5.2), vhodným riešením môže byť aj spojenie GNN a plne prepojených sietí. Avšak takéto riešenie stráca schopnosť generalizovať na rôzne veľkosti úloh.

Budúce práce v oblasti by sa mohli venovať ďalším vlastným návrhom grafov a analýzou vplyvu rôznych vlastností. Ďalej si myslíme, že návrhom vhodnej siete, či už v kombinácii s iným typom siete alebo čisto GNN, môžu byť dosiahnuté lepšie výsledky. Tiež by bolo možno vhodné vyskúšať aj iné operácie na agregáciu vrcholov. Základnou operáciou je operácia sum (slov.: sčítavanie) a medzi možnosti patria operácie mean (slov.: priemer), min, max a mul (slov.: násobenie). Myslíme si, že operácia sčítania má veľký vplyv na výskyt EDB a preto by iné operácie mohli viesť k zníženiu vplyvu.

ZÁVER

V experimentoch sme popísali, že agenti mali problém s tréningom už pri jednoduchej úlohe – hľadanie najlacnejšej kostry. V práci sme sa preto nevenovali úlohe obchodného cestujúceho, ale analyzovali sme úlohu hľadania kostry do väčších detailov.

Použitím grafových neurónových sietí na riešenie úlohy hľadania najlacnejšej kostry sme vykonali niekoľko experimentov, pri ktorých sme ukázali, že EDB má vplyv počas tréningu. Nie je to ale jediný problém, ktorý je potrebné prekonať na úspešné riešenie daných úloh. Myslíme si, že agregácie v GNN spôsobujú rozostrenie dôležitých informácií potrebných na riešenie úloh najlacnejšej kostry a úloh jej podobným. V porovnaní s GNN sú plne prepojené siete schopné riešiť danú úlohu a nemajú problém s EDB a rozostrením informácií.

Odporúčame, aby výskum v oblasti vplyvu EDB na tréning aj ďalej pokračoval so zameraním hlavne na vlastnosti grafov, ktoré vplývajú na vytvárané riešenia a taktiež aj na spôsob tvorby riešenia.

Ďalší výskum by sa mohol zamerať aj na iné stratégie výberu akcií. Jednou z možností je priame vyberanie hrán, čo by si však vyžadovalo zavedenie umelých vrcholov do hrán (ako bolo spomenuté v kapitole 3.3.1). Ďalšou možnosťou je úprava reprezentácie stavu prostredia. Príkladom takejto úpravy môže byť pridanie tretieho atribútu k vrcholom, ktorý by uchovával globálnu informáciu o tom, v ktorom rozhodovacom kroku sa agent nachádza. Táto informácia by bola pevnejšie zakotvená v stave prostredia, čo by mohlo posilniť schopnosť agenta správne rozpoznať aktuálnu fázu výberu. Tiež by bolo vhodné otestovať aj iné GNN poskytované knižnicou „pytorch_geometric“, alebo vytvoriť vlastný návrh message passingu v GNN, ktorý by dokázal zmierniť obidva spomínané problémy.

Naša práca môže slúžiť ako podklady pre ďalší výskum v oblasti. Poskytuje rôzne zistenia, spôsoby riešenia a problémy, ktoré vznikajú nielen pri úlohe hľadania najlacnejšej kostry ale aj pri úlohách jej podobným.

Zoznam použitej literatúry

[1] P. Tarábek a D. Matis, „Exploration Degree Bias: The Hidden Influence of Node Degree in Graph Neural Network-Based Reinforcement Learning,“ IEEE, 2025.

[2] N. Mazyavkina , S. Sviridov , S. Ivanov a E. Burnaev, „Reinforcement learning for combinatorial optimization: A survey,“ Elsevier, 2021.

[3] R. S. Sutton a A. G. Barto, Reinforcement Learning: An Introduction, Cambridge: MIT Press, 2018.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford a O. Klimov, „Proximal Policy Optimization Algorithms,“ arXiv, 2017.

[5] „Best Practices when training with PPO,“ 29 December 2017. [Online]. Available: https://github.com/EmbersArc/PPO/blob/master/best-practices-ppo.md. [Cit. 2025].

[6] A. Goldie, A. Mirhoseini, M. Yazgan a e. al., „Addendum: A graph placement methodology for fast chip design.,“ Nature, 2024.

[7] D. Silver a e. al., „Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm,“ arXiv, 2017.

[8] C. Wang, C. Han, T. Guo a e. al., „Solving uncapacitated P-Median problem with reinforcement learning assisted by graph attention networks,“ Springer, 2022.

[9] S. Palúch, Algoritmická teória grafov, Žilina: EDIS Žilina, 2020.

[10] Sanchez-Lengeling a e. al., „A Gentle Introduction to Graph Neural Networks,“ Distill, 2021. [Online]. Available: https://distill.pub/2021/gnn-intro. [Cit. 2025].

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò a Y. Bengio, „GRAPH ATTENTION NETWORKS,“ arXiv, 2018.

[12] Z. Liu, T.-K. Nguyen a Y. Fang, „On generalized degree fairness in graph neural networks,“ AAAI, 2023.

[13] J. Wu, X. Wnag, F. Feng, X. He, L. Chen, J. Lian a X. Xie, „Self-supervised Graph Learning for Recommendation,“ arXiv, 2020.

[14] PyTorch Foundation, „Pytorch,“ 2025. [Online]. Available: https://pytorch.org/. [Cit. 2025].

[15] PyG Team, „PyG Documentation,“ 2025. [Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/. [Cit. 2025].

[16] The Matplotlib development team, „Matplotlib,“ 2024. [Online]. Available: https://matplotlib.org/. [Cit. 2025].

[17] NetworkX developers, „NetwrokX,“ 2024. [Online]. Available: https://networkx.org/. [Cit. 2025].

[18] NumPy team, „NumPy,“ 2025. [Online]. Available: https://numpy.org/. [Cit. 2025].

[19] pandas team, „pandas - Python Data Analysis Library,“ 2025. [Online]. Available: https://pandas.pydata.org/. [Cit. 2025].

[20] „MilosMurin/MyDiploma,“ 2025. [Online]. Available: https://github.com/MilosMurin/MyDiploma. [Cit. 2025].

[21] „MyDiploma/Dimploma/graphs,“ 2025. [Online]. Available: https://github.com/MilosMurin/MyDiploma/tree/master/Dimploma/graphs. [Cit. 2025].

[22] „PPO parallel,“ 2022. [Online]. Available: https://github.com/balaz94/FRI-RL-SC2/blob/main/agents/ppo_parallel.py. [Cit. 2025].

Prílohy

Zoznam príloh

Príloha A | Pamäťové médium

Médium obsahuje:

- Práca vo formáte pdf
- Zdrojové kódy a grafy použité pri práci (dostupné aj na https://github.com/MilosMurin/MyDiploma)