SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Primjena strojnog učenja u kombinatornoj optimizaciji za upravljanje lancem opskrbe

Jana Juroš

Zagreb, lipanj 2021.

Ovaj rad izrađen je na Zavodu za primijenjeno računarstvo pod vodstvom doc. dr. sc. Marija Brčića i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2020./2021.

SADRŽAJ

1. Uvod

Kombinatorni optimizacijski problem je problem pronalaženja optimalnog objekta iz skupa svih objekata. Drugim riječima, cilj takvih problema je pronaći najbolje odnosno optimalno rješenje u diskretnim prostorima gdje je iscrpno pretraživanje najčešće netraktabilno. Takav problem definiran je funkcijom cilja i funkcijama ograničenja.

Kombinatorna optimizacija (engl. *Combinatorial optimization - CO*) je proces pronalaženja optimalnog rješenja gore navedenih problema i svodi se na pronalaženje ekstrema (minimuma ili maksimuma) *funkcije cilja* uz poštovanje zadanih *ograničenja*. CO ima primjene u elektronici, transportu, proizvodnji, maloprodaji, a u ovom radu ju primjenjujemo na optimizacijski problem upravljanja lancem opskrbe.

Takve (CO) probleme može biti izuzetno teško riješiti, a zapravo su većina klasičnih NP-teških problema primjeri kombinatorne optimizacije. S obzirom na težinu ovih problema, suvremeni algoritmi oslanjaju se na ručno izrađene heuristike za donošenje odluka koje su inače preskupe za izračunavanje ili matematički nedovoljno definirane. Međutim pokazalo se da je takve heuristike vrlo teško konstruirati, a njihovo je izračunavanje sporo. S druge strane, strojno učenje ima moć statističkog učenja iz podataka pa izgleda kao prirodan kandidat za donošenje takvih odluka na principijalniji i optimizirani način[2].

Predstavit ćemo još nekoliko intuitivnih razloga za korištenje strojnog učenja u rješavanju CO problema. Naime, suvremeni egzaktni algoritmi koji koriste spomenute heuristike će problem rješavati jednako uspješno neovisno o tome koliko su puta riješili problem iz iste klase. Drugim riječima, spomenuti algoritmi ne koriste priliku da uče iz već riješenih instanci istog problema. Međutim, u praksi je uobičajeno u više navrata rješavati slične probleme kombinatorne optimizacije (npr. problem upravljanja lancem opskrbe), koji se mogu značajno razlikovati od skupa slučajeva na kojima se tipično procjenjuju navedeni algoritmi. Tada ima smisla koristiti statističko učenje za automatsko podešavanje algoritama za željenu klasu problema. Tako možemo ostvariti specijaliziranije i uspješnije algoritme za pojedine klase problema. Nadalje, duboko učenje krasi i mogućnost učenja značajki skupa podataka (engl. *dataseta*) bez

značajnih intervencija (poput konstruiranja novih značajki). Zato bi buduće održavanje i razvoj *rješavača* (engl. *solvera*) bilo jednostavnije i uspješnije uz strojno učenje. U radu ćemo pokazati kako možemo primijeniti statističku snagu strojnog učenja na mehanizam rješavanja CO problema.

U ovom radu provest ćemo integraciju kombinatorne optimizacije i strojnog učenja replicirajući postojeći rad[3] iz 2019. godine te ga primijeniti na drugi problem, uz potrebne preinake da se postigne čim bolji rezultat. Konkretno, kombinirat ćemo suvremeni rješavač SCIP i grafovsku konvolucijsku neuronsku mrežu kako bismo rješavali instance problema upravljanja lancem opskrbe. Konkretne instance problema su stvarni problemi raspoređivanja opskrbe 420 Konzumovih dućana, uzimajući u obzir balansiranje količine rada u skladištima i balansiranje opterećenja transporta kroz tjedan.

2. Opći i specifični ciljevi rada

2.1. Kombinatorna optimizacija

Kombinatorna optimizacija sastoji se od pronalaska optimalnog rješenja iz konačnog skupa rješenja u problemima gdje iscrpna pretraga najčešće nije traktabilna. Takvim problemima pripadaju i *21 Karpovih NP potpunih problema*[3] od kojih je najpoznatiji *problem trgovačkog putnika*.

Bez gubitka općenitosti, CO problem može se formulirati min-optimizacijski problem s ograničenjima. Takav problem sastoji se od *funkcija ograničenja* (engl. *constraints*) koje rješenje mora zadovoljavati i *ciljne funkcije* (engl. *objective function*) koju rješenje treba minimizirati. Takva formulacija problema u kojoj ciljnu funkciju želimo minimizirati nas ne ograničava. Naime, ako je priroda našeg problema da želimo maksimizirati ciljnu funkciju, onda ćemo problem formulirati tako da negiramo ciljnu funkciju i nastojimo ju minimizirati.

Općenito, ograničenja modeliraju prirodna ili nametnuta ograničenja problema, varijable definiraju odluke koje treba donijeti, dok ciljna funkcija, općenito trošak koji se minimizira, definira mjeru kvalitete svake izvedive dodjele vrijednosti varijablama. Skup točaka koje zadovoljavaju funkcije ograničenja nazivamo *ostvarivo područje* (eng. *feasible region*). Svaka točka u tom skupu (koju nazivamo *ostvarivo rješenje*) daje gornju granicu vrijednosti optimalnog rješenja.

2.2. Problemi linearnog programiranja i mješovitog cjelobrojnog linearnog programiranja

Ako su funkcije cilja i ograničenja linearne, problem se naziva problem linearnog programiranja (engl. *Linnear programming - LP*). Ako su, uz to, neke varijable također ograničene samo na cjelobrojne vrijednosti, takav problem nazivamo problem mješovitog cjelobrojnog linearnog programiranja (engl. *Mixed Integer Linear Programming*

- *MILP*). Ako funkcije ograničenja podijelimo na funkcije nejednakosti i granice (engl. *bounds*) na varijable definiciju MILP-a izražavamo na sljedeći način:

$$\begin{aligned} &\textit{ciljna funkcija:} && minimiziraj \quad c^T x \\ &\textit{funkcije ograničenja} && Ax \leq b, \quad l \leq x \leq u \\ &\textit{funkcije ograničenja (MILP)} && x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

LP i MILP mogu modelirati širok spektar problema te postoje pouzdani algoritmi i softverski alati koji rješavaju takve probleme. Zato ćemo se usredotočiti samo na te vrste problema, iako LP i MILP ne predstavljaju sve probleme iz CO. S obzirom na složenost i metode rješavanja, LP je polinomski problem rješiv pomoću *simplex* algoritma (simplex u najgorem slučaju ima eksponencijalnu složenost, ali se to praktično ne događa) ili *metode unutarnjih točaka* koje ovdje nećemo razmatrati. S druge strane, MILP je NP-težak problem. Složenost MILP-a povezana je sa zahtjevom cjelobrojnosti (nekih) varijabli, što čini područje ostvarivosti nekonveksnim[2]. Međutim, odustajanje od zahtjeva cjelobrojnosti definira pravilnu relaksaciju MILP-a (tj. problem optimizacije čije područje ostvarivosti sadrži područje ostvarivosti MILP-a), što je LP, tj. polinomski rješivo. Tu činjenicu iskorištava i *algoritam grananja i ograđivanja* (engl. *branch-and-bound - B&B*).

2.3. Branch-and-bound (B&B) algoritam

Branch-and-bound implementira algoritam "podijeli pa vladaj" koji se može predstaviti pomoću stabla pretraživanja u kojem se na svakom čvoru izračunava LP relaksacija zadanog MILP problema. Ako relaksacija nema rješenje ili ako relaksacija ima rješenje koje poštuje cjelobrojna ograničenja za zadane varijable, tada taj čvor ne treba dalje proširivati. Inače, postoji barem jedna varijabla, među onima za koje se pretpostavlja da su cjelobrojne, koja ima necjelobrojnu vrijednost u rješenju LP-a i ta se varijabla može odabrati za grananje. Takvih varijabla može biti više pa onda biramo po kojoj ćemo varijabli dalje granati.

Grananje provodimo na način da odaberemo jednu od gore navedenih varijabli te izvedemo dvije grane tako da u jednoj uzmemo sva prethodna ograničenja i dodamo ograničenje da odabrana varijabla mora biti manja ili jednaka "stropu" trenutnog decimalnog rješenja LP-a. Drugu granu izvodimo na način da uzmemo sva prethodna ograničenja i dodamo novo ograničenje da odabrana varijabla mora biti veća ili jednaka "podu" trenutnog decimalnog rješenja LP-a. Tim dvama novim ograničenjima

dobili smo dvije grane čiji listovi (još neproširени čvorovi) imaju disjunktna područja izvedivosti te niti jedno ne sadrži rješenje prethodne LP relaksacije. Ako naiđemo na čvor gdje su sve varijable koje trebaju biti cjelobrojne, tada je to moguće rješenje (tražimo optimalno, odnosno najbolje takvo rješenje). Algoritam izvodimo dok postoji čvor koji možemo (i ima smisla) proširiti.

2.3.1. Odluke u B&B algoritmu

Iterativni postupak algoritma zahtijeva sekvencijalno donošenje dviju vrsti odluka:

1. odabir sljedećeg čvora za procjenu
2. odabir varijable po kojoj će se particionirati prostor pretraživanja čvora - biranje varijable po kojoj će se stablo dalje granati (npr. x ili y)

Ovaj postupak donošenja odluka tradicionalno slijedi niz fiksno napisanih (engl. *hard-coded*) heuristika, pažljivo osmišljenih od strane stručnjaka kako bi se smanjilo prosječno vrijeme rješavanja na reprezentativnom nizu MILP slučajeva [3]. Međutim, u praksi je uobičajeno u više navrata rješavati slične probleme kombinatorne optimizacije (npr. *problem raspoređivanja poslova na strojeve*), koji se mogu značajno razlikovati od skupa slučajeva na kojima se tipično procjenjuju navedeni algoritmi. Tada ima smisla koristiti statističko učenje za automatsko podešavanje algoritama za željenu klasu problema. Tako možemo ostvariti specijaliziranije i uspješnije algoritme za pojedine klase problema.

Međutim, ova vrsta rada postavlja dva izazova. Prvo, nije očito kako kodirati stanje procesa odlučivanja u algoritmu B&B, pogotovo jer i stabla pretraživanja i MILP mogu imati promjenjivu strukturu i veličinu. Drugo, nije jasno kako formulirati model koji vodi do pravila koja se mogu generalizirati, barem na slične slučajeve, ali idealno i na slučajeve veće od viđenih tijekom treninga [3].

2.4. Primjena grafovske konvolucijske neuronske mreže u B&B algoritmu

U ovom radu ćemo kao model koristiti grafovsku konvolucijsku neuronsku mrežu kako bismo ubrzali B&B algoritam u solveru SCIP-u. Preciznije, usredotočit ćemo se na odabir varijabli, poznat i kao problem grananja, koji leži u srži B&B paradigme. Pokušat ćemo usvojiti strategiju imitacijskog učenja kako bismo naučili brzu aproksimaciju

snažnog grananja (engl. *strong branching*) koje je često presporo da bi se koristilo u rješavanju velikih problema.

Snažno grananje je visokokvalitetno, ali skupo (dugo se izračunava) pravilo grananja koje rezultira najmanjim B&B stablom[3]. To čini izračunavanjem očekivanog poboljšanja gornje granice za svaku varijablu koja je kandidat za grananje u čvoru, što nažalost zahtijeva rješenja dvaju LP-a za svakog kandidata. U praksi se snažno grananje ne koristi na svakom čvoru, nego se umjesto toga primjenjuje hibridno grananje koje upotrebljava snažno grananje na početku procesa rješavanja i postupno prelazi na jednostavnije heuristike kao što su: rezultat sukoba (eng. conflict score), pseudo-trošak (eng. pseudo-cost) ili kombinacija navedenih.

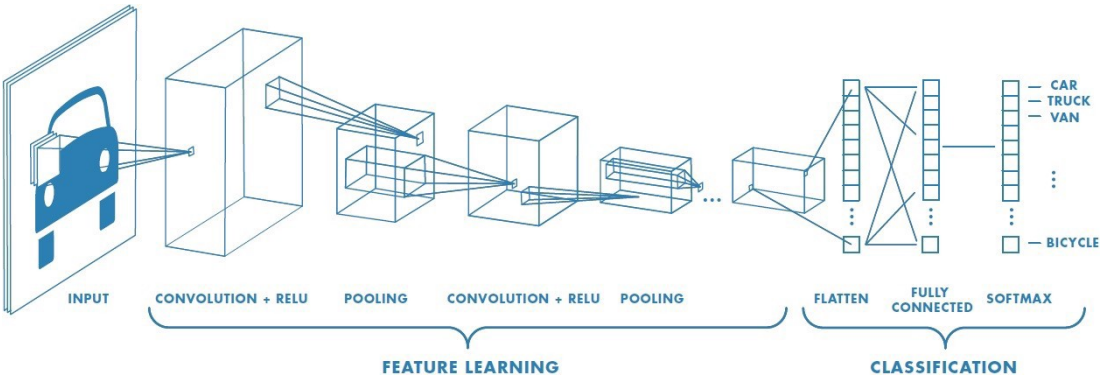
Naš je cilj napraviti model (grafovsku konvolucijsku neuronsku mrežu) koji imitira snažno grananje u SCIP-u za određenu klasu problema. To ćemo postići pokretanjem rješavanja mnogo problema iz iste klase u SCIP-u i "snimanjem" parova *(stanje, akcija)*. Odnosno, za neko *stanje* u algoritmu vidjet ćemo koju će varijablu politika snažnog grananja odabrati za proširivanje *(akcija)*. Mrežu ćemo trenirati i testirati nad dobivenim parovima te ćemo probati postići da model što bolje imitira spomenutu politiku. Nakon treninga mreže, integrirat ćemo ju u SCIP kao novu politiku grananja, koja radi približno dobro kao i snažno grananje, no značajno brže. Tako ćemo statističkim predmemoriranjem (engl. *caching*), interpolacijama i ekstrapolacijama u neuralnim mrežama pokušati ubrzati cijeli algoritam rješavanja.

2.5. Konvolucijska neuronska mreža

Umjetna neuronska mreža je skup međusobno povezanih jednostavnih procesnih elemenata (neurona) čija se funkcionalnost temelji na biološkom neuronu i koji služe distribuiranoj paralelnoj obradi podataka[4]. Omogućavaju robusnu obradu podataka te su sposobne učiti iz podataka. Možemo ih koristiti za probleme klasifikacije i regresije.

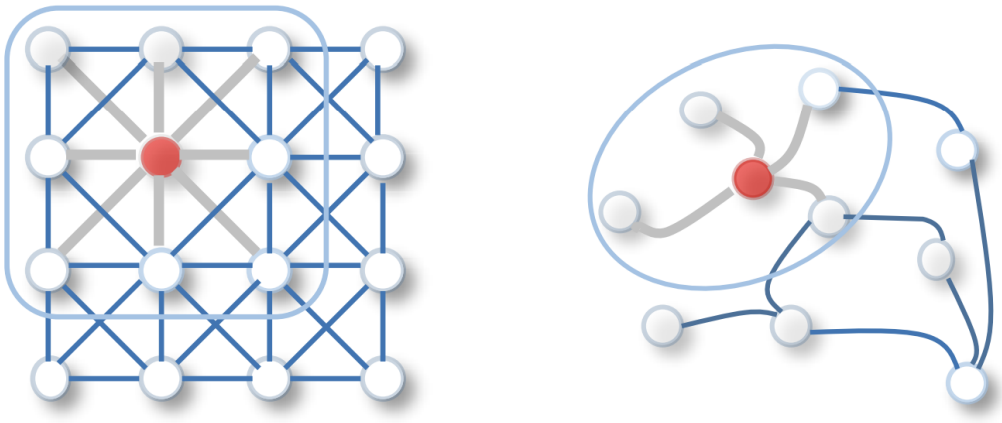
Konvolucijska neuronska mreža (CNN) je specifična vrsta neuronskih mreža koja daje izuzetno dobre rezultate pri obradi slikovnih podataka te je glavna karakteristika takvih mreža da model uči značajke pregledavajući susjedne čvorove. Konvolucijska neuronska mreža se, kao i obična, sastoji od jednog ulaznog, jednog izlaznog i jednog ili više skrivenih slojeva. Kod konvolucijskih neuronskih mreža specifični su konvolucijski slojevi i slojevi sažimanja. Konvolucijske neuronske mreže na početku najčešće imaju jedan ili više konvolucijskih slojeva, zatim slijedi sloj sažimanja, pa ponovo konvolucijski sloj i tako nekoliko puta. Mreža najčešće završava s jednim ili više potpuno

povezanih slojeva[7].



Slika 2.1: Struktura konvolucijskih neuronskih mreža - izmjena slojeva konvolucije i prožimanja i na kraju potpuno povezani slojevi za klasifikaciju[9]

U našem modelu, koristili smo grafovsku konvolucijsku neuronsku mrežu (GCN). Grafovske konvolucijske neuronske mreže proširenje su konvolucijskih neuronskih mreža koje osim regularno strukturiranih podataka (poput slika) mogu obrađivati i proizvoljne grafove. GCN-ovi izvode slične operacije kao i CNN gdje model uči značajke pregledavajući susjedne čvorove. Glavna razlika između CNN-a i GNN-a je u tome što su CNN-ovi posebno izrađeni za rad na redovnim (euklidskim) strukturiranim podacima, dok su GNN-ovi generalizirana verzija CNN-a gdje broj veza čvorova varira, a raspored čvorova nije uređen (nepravilni ili neeuklidski strukturirani podaci)[10].



Slika 2.2: Ilustracija konvolucijske i grafovske konvolucijske umjetne neuronske mreže[10]

2.6. Problem upravljanja lancem opskrbe

Instance problema nad kojima ćemo trenirati, testirati i evaluirati mrežu stvarni su primjerci problema, prikupljeni iz podataka o potrebama opskrbe više Konzumovih pos-

lovnica. Konkretno, radi se o opskrbi 420 Konzumovih poslovnica, uzimajući u obzir kapacitet skladišta i utilizaciju transporta te rok trajanja proizvoda. Rješenje problema čine tjedni obrasci isporuke tj. osnovni raspored koji se uzima kao baza za rasporede u svim tjednima u nekom periodu. Potencijalne reaktivne promjene, temeljene na specifičnostima određenih tjedana (na primjer blagdani), provode se kao lokalna pretraga osnovnog rasporeda. Model koji koristimo za formulaciju problema kao MILP nije tema ovog rada pa ga ovdje nećemo dalje opisivati.

3. Plan rada

U ovom poglavlju detaljno ćemo opisati metodologiju rada. Opisat ćemo stvaranje skupa podataka, parametrizaciju stanja B&B algoritma u solveru, izvlačenje podataka iz solvera (SCIP-a) te stvaranje nove politike grananja imitiranjem politike "snažnog grananja". Zatim ćemo opisati naš model - grafovsku konvolucijsku neuronsku mrežu - te njezino treniranje i testiranje. Konačno opisujemo evaluaciju rezultata.

Radi lakšeg razumijevanja navodimo algoritam cijelog postupka:

1. **Stvaranje uzoraka** (engl. *sample*) - pokretanje rješavanja instanci MILP-a u SCIP-u i snimanje parametriziranih parova *(stanje, akcija)* koje dobivamo dolje navedenim postupkom i koji predstavljaju politiku *strong branching*

2. **Treniranje mreže** - pokretanje treniranja na uzorcima za treniranje i validaciju u kojem naš model (grafovska konvolucijska neuronska mreža) uči imitirati politiku jakog grananja

3. **Testiranje mreže** - pokretanje utreniranog modela na novim uzorcima za testiranje kako bismo procijenili točnost modela

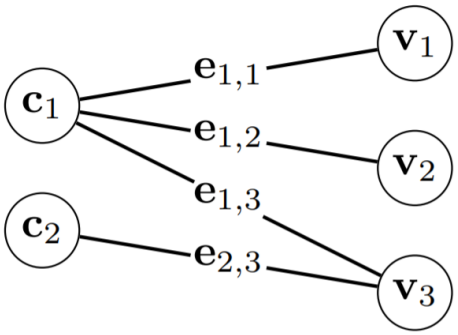
4. **Evaluacija mreže** - pokretanje rješavanja novih MILP instanci u SCIP-u s našim utreniranim modelom u ulozi politike grananja i usporedba brzine rješavanja s "običnim" SCIP-om

3.1. Parametrizacija stanja B&B algoritma

Budući da se MILP problemi sastoje od varijabli i ograničenja koja se odnose na te varijable, prirodno je stanje prikazati u obliku bipartitnog grafa. Tako s jedne strane imamo čvorove (engl. *nodes*) ograničenja (koji nisu međusobno povezani) te s druge strane čvorove varijabli (koji također nisu međusobno povezani). Ako se ograničenje odnosi na pojedinu varijablu, ta dva čvora (varijable i ograničenja) povezani su

bridom (engl. *edge*). Nadalje, pretpostavimo da u stanju rješavača (B&B algoritma) postoji m ograničenja, od kojih svako ima e značajki, te n varijabli, od kojih svaka ima d značajki. Tada je čvorovima ograničenja pridružena matrica značajki ograničenja $\mathbf{C} \in \mathbb{R}^{m \times c}$, a čvorovima varijabli matrica značajki varijabli $\mathbf{V} \in \mathbb{R}^{n \times d}$. Konačno, bridovima je pridružena matrica značajki bridova $\mathbf{E} \in \mathbb{R}^{m \times n \times e}$. Dakle, parametriziramo stanje s_t B&B procesa u trenutku t kao bipartitni graf sa vrijednostima značajki čvorova i bridova ($\mathcal{G}$, $\mathbf{C}$, $\mathbf{E}$, $\mathbf{V}$). Točne značajke varijabli, ograničenja i bridova su navedene u tablici ispod. Nadalje, budući da u B&B algoritmu iterativno dodajemo nova ograničenja, moramo ih dodati i u bipartitnu reprezentaciju stanja. Za svako novo ograničenje dodat ćemo u graf novi čvor ograničenja s dodijeljenim značajkama i bridove (sa značajkama) koji taj čvor ograničenja povezuju s čvorovima varijabli koje se nalaze u samom ograničenju. Također, treba spomenuti da u stanje pohranjujemo i kandidate za grananje koje kao i ostale možemo pročitati iz stanja rješavača. Primijetimo da naše stanje čini samo podskup stanja rješavača. Budući da je u spomenutom radu koji repliciramo [3] već napisan kod za preuzimanje vrijednosti značajki stanja iz SCIP-a tijekom rješavanja, iskoristili smo postojeći kod za snimanje uzoraka. Taj kod izvlači podatke iz SCIP-a u Khalilovo stanje[16]. Iskoristili smo navedeni kod da rješava instance problema upravljanja lancem opskrbe te pri rješavanju povremeno radi sljedeće:

1. čita stanje SCIP-a
2. izvlači iz njega značajke koje su mu potrebne
3. stvara bipartitnu reprezentaciju stanja koja se sastoji od:
   (a) varijabli i njihovih vrijednosti značajki
   (b) ograničenja i njihovih vrijednosti značajki
   (c) dvaju vektora incidencije usmjerenih bridova (jedan vektor bridova iz varijabli u ograničenja i drugi obrnuti) i značajki tih bridova
4. pokreće politiku snažnog grananja
5. stvara uzorke (engl. *sample*) pohranjujući bipartitnu reprezentaciju stanja, kandidate za grananje te rezultate (engl. *score*) kandidata dobivene primjenjivanjem politike grananja (politika snažnog grananja na kraju izračunavanja dobiva rezultate (engl. *score*) i na temelju njih odlučuje koja se varijabla bira za daljnje grananje - ona s najvećim rezultatom.



Slika 3.1: Bipartitna reprezentacija stanja s_t = ($\mathcal{G}$, **C**, **E**, **V**) s $n = 3$ varijable i $m = 2$ ograničenja[3]

3.2. Stvaranje skupa podataka

Od Konzuma smo dobili 36 tjednih problema raspoređivanja opskrbe u 420 poslovnica. Od tih 36 instanci (svaka za jedan tjedan) proizveli smo 334 potproblema po regijama (regije se sastoje od 25-50 trgovina). Dobivene potprobleme razdvojili smo u tri skupa: skup instanci za učenje (240 instanci), validaciju (60 instanci) i testiranje (34 instanci) kako bismo mogli provesti cjelokupni trening mreže s testiranjem.

3.3. Parametrizacija politike odabira varijable u B&B algoritmu

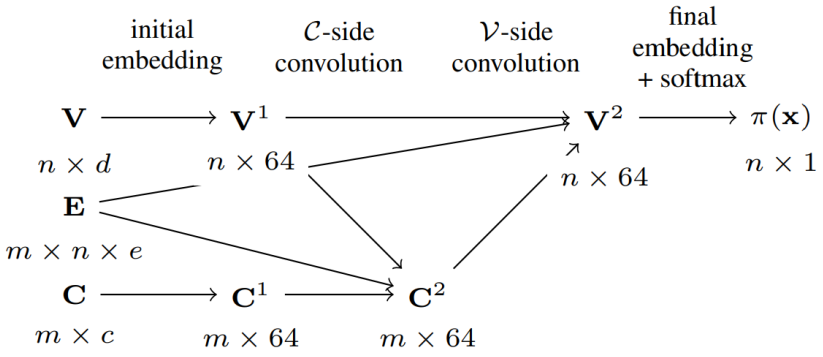
Politiku odabira varijabli $\pi_0(\mathbf{a}|\mathbf{s}_t)$ parametriziramo kao grafovsku konvolucijsku neuronsku mrežu. Model i trening mreže napisali smo u Pytorchu[15], *open source* okviru za duboko učenje koji je stvorio Facebook. Napisali smo ih po uzoru na rad[2] gdje su mreža i trening napisani u Tensorflowu[14], nešto starijem okviru za duboko učenje koji je stvorio Google. Njihov programski kod prebacili smo u Pytorch te ga doradili i prilagodili našem problemu. Naš model uzima kao ulaz spomenutu bipartitnu reprezentaciju stanja s_t=($\mathcal{G}$, **C**, **E**, **V**) i izvodi jednu grafovsku konvoluciju, u obliku dvije isprepletene polukonvolucije. Naime, zbog bipartitne strukture ulaznog grafa, naša se konvolucija grafa može podijeliti u dva uzastopna prolaza. Budući da nas zanima izbor varijabli, u prvom prolazu dodijeljujemo nove vrijednosti čvorovima ograničenja, a u drugom prolazu dodijeljujemo nove vrijednosti čvorovima varijabli, uzimajući u obzir nove vrijednosti čvorova ograničenja (kako bi na izlazu iz mreže bile

varijable). Navedeni prolazi imaju oblik:

$$\mathbf{c}_i \leftarrow \mathbf{f}_{\mathcal{C}}(\mathbf{c}_i \sum_{j}^{(i,j)\in\mathcal{E}} \mathbf{g}_{\mathcal{C}}(\mathbf{c}_i, \mathbf{v}_j, \mathbf{e}_{i,j}), \qquad \mathbf{v}_j \leftarrow \mathbf{f}_{\mathcal{V}}(\mathbf{v}_j \sum_{i}^{(i,j)\in\mathcal{E}} \mathbf{g}_{\mathcal{V}}(\mathbf{c}_i, \mathbf{v}_j, \mathbf{e}_{i,j})$$

za svaki $i \in \mathcal{C}, j \in \mathcal{V}$ gdje su $\mathbf{f}_{\mathcal{C}}, \mathbf{g}_{\mathcal{C}}, \mathbf{f}_{\mathcal{V}}, \mathbf{g}_{\mathcal{V}}$ 2-slojni perceptroni s *relu* aktivacijskim funkcijama[3]. Malo ćemo pojasniti gornje izraze, uzmimo prvi za objašnjenje. U prvom izrazu prolazimo po čvorovima ograničenja. Budući da u grafovskoj konvolucijskoj mreži gledamo susjede u grafu, za svaki čvor ograničenja imamo sumu koja prolazi po njegovim susjedima u grafu uzimajući u obzir značajke čvora ograničenja $\mathbf{c}_i$ koji trenutno procijenjujemo, trenutni čvor varijable u sumi $\mathbf{v}_j$ te značajke brida koji spaja ta dva čvora. Nakon prolaska po čvorovima ograničenja, obavljamo drugu polukonvoluciju prolazeći po čvorovima varijable, ali uvrštavajući novoizračunate čvorove ograničenja. Tako na optimirani način odradimo cijelu konvoluciju. U teoriji, mogli smo koristiti kanonsku graf konvolucijsku mrežu koju koristimo kod općenitih grafova. Ovdje koristimo ručno napisanu bipartitnu konvoluciju radi računalne efikasnosti gdje smo iskoristili unaprijed poznatu strukturu grafa.

Prolaskom kroz grafovski konvolucijski sloj, dobivamo bipartitni graf s istom topologijom kao i ulaz, ali s potencijalno različitim značajkama čvora, tako da svaki čvor sada sadrži akumulirane informacije svojih susjeda. Tražene politike biranja varijable dobivamo odbacivanjem čvorova ograničenja i primjenom završnog 2-slojnog perceptrona na čvorovima varijabli, u kombinaciji s maskiranom softmax aktivacijskom funkcijom kako bismo dobili vjerojatnosnu raspodjelu po varijablama koje su kandidati za grananje. Zatim filtriramo kandidate za grananje (koje smo dobili iz stanja SCIP-a) te funkciju gubitka osvježavamo ovisno o tome poklapa li se izbor naše mreže s izborom politike snažnog grananja.



Slika 3.2: Prikaz bipartitne reprezentacije grafovske konvolucijske neuronske mreže korištene za parametrizaciju politike grananja $\pi_0(\mathbf{a}|\mathbf{s}_t)$[3].

3.4. Postupak treniranja modela

Započinjemo treniranje grupiranjem podataka u minigrupe (engl. *mini-batches*). Minigrupe stvaramo čitajući uzorke i spajanjem ih u jedan veliki bipartitni graf (uz kandidate i rezultate).

U trening petlji iteriramo po stvorenim minigrupama i ubacujemo ih u model. Izlaz modela čine rezultati za sve varijable koje se nalaze u minigrupi. Filtriramo varijable kandidate te u funkciju gubitka unosimo rezultate kandidata i indekse varijabli koje je odabrala politika snažnog grananja (po jedan indeks za svaki podatkovni redak iz minigrupe).

Kao funkciju gubitka odabrali smo *funkciju gubitka unakrsne entropije* (engl. *Cross-Entropy Loss*), funkciju koja raste kako izračunate vjerojatnosti odudaraju od stvarnih oznaka. Zatim provodimo algoritam propagacije pogreške unazad. Algoritam propagacije pogreške unazad je optimizacijski postupak koji se temelji na izračunu gradijenata (parcijalnih derivacija funkcije pogreške s obzirom na svaku težinu i prag)[4]. Algoritmom ciljano mijenjamo težine u filterima konvolucijskih slojeva s ciljem smanjivanja vrijednosti funkcije gubitka.

U našoj trening rutini koristimo Adam optimizator. Adam je algoritam za optimizaciju koji se može koristiti umjesto klasičnog postupka stohastičkog gradijentnog spusta za ažuriranje mrežnih težina na temelju podataka treninga te ga karakterizira učinkovitost, mali zahtjevi za memorijom te pogodnost za probleme koji su veliki u pogledu podataka i / ili parametara. Koristili smo početni parametar stope učenja 10^{-3} te se smanjivao multiplikativno sa konstantom 0.2 na svaku detekciju platoa koju bismo prepoznali nakon 5 vremenskih koraka nepoboljšanja performanse na validacijskom setu.

Nakon korekcije težina prelazimo na iduću minigrupu i ponovno provodimo isti postupak. Jedan prolazak kroz sve podatke za treniranje (100 000 uzoraka) nazivamo epoha. Naš program će izvesti najviše 312 epoha prije zaustavljanja, ako prije ne dođe do prekida. Do prekida treniranja dolazi ako nema poboljšanja (ako se funkcija gubitka ne smanji) dvadeset epoha zaredom na uzorcima za validaciju (20 000 uzoraka).

Treba spomenuti da na kraju svake epohe provjeravamo je li funkcija gubitka (na podacima za validaciju) najmanja do sada te ako je, spremamo parametre modela kao najbolje dosad. Također, u odnosu na polazni rad morali smo koristiti podrezivanje gradijenta (engl. *gradient clipping*) na raspon $[-5, 5]$ jer bismo inače imali probleme usred procesa treninga koji bi nam narušavali učenje. U polaznom radu je sve funkcioniralo na odabranom skupu problema i bez te promjene.

3.5. Testiranje i evaluacija modela

Za konačnu evaluaciju efektivnosti našeg pristupa sačuvali smo 34 instanci problema koji su u vremenu iza svih instanci (podatci iz kasnijih tjedana) korištenih za trening i validaciju iz kojih radimo dva testa nad utreniranim modelom:

1. klasifikacijsku preciznost određivanja varijable za grananje nad testnim skupom podataka - uzorci generiranih iz testnih instanci problema

2. efektivnost u optimizaciji nad testnim instancama problema

Testiranje provodimo na uzorcima za testiranje (20 000 uzoraka) prikupljenim iz navedenih instanci. Testiranje provodimo iterirajući po minigrupama jedan put, unoseći minigrupe u model s najboljim parametrima dobivenim procesom treniranja. Tijekom procesa iteriranja ne izračunavamo funkciju gubitka niti upotrebljavamo optimizator, no osvježavamo točnost našeg modela te ju na kraju predočavanja svih podataka za testiranje ispisujemo.

Evaluaciju modela provodimo na instancama MILP problema koji nisu bili korišteni za stvaranje uzoraka za trening i testiranje mreže. Pokrećemo rješavanje problema u SCIP-u s našim modelom u ulozi politici grananja te zatim sa zadanom (engl. *default*) politikom grananja. Na kraju uspoređujemo rezultate tj. brzinu rješavanja problema.

4. Rezultati

4.1. Testiranje mreže

Nakon treniranja mreže, slijedi testiranje mreže. Testiranje smo proveli na način da smo uzeli najbolje parametre mreže dobivene tijekom treniranja (parametri za koje je funkcija gubitka na validacijskim uzorcima bila najmanja) te inicijalizirali mrežu s tim parametrima. Zatim smo jednom kroz mrežu proveli nove uzorke koji mreži nisu bili dostupni tijekom treniranja - uzorke za testiranje. Dobili smo sljedeće rezultate:

acc@1	acc@3	acc@5	acc@10
37.24	58.99	68.34	79.28

Svaki od stupaca označava stopu točnosti za broj iza "@" znaka - nazovimo taj broj n. Tako bi acc@3 predstavljao stopu točnosti za $n = 3$. Stopu točnosti računamo tako da za svaki uzorak u minigrupi uzmemo n najboljih varijabli iz snimljenog uzorka i gledamo nalazi li varijabla koja ima najbolji rezultat u našoj mreži među tim varijablama. Ako se nalazi, taj uzorak označimo s jedan, inače s nulom. Nakon što označimo sve uzorke u minigrupi, nađemo srednju vrijednost tih oznaka i taj broj nam predstavlja stopu točnosti za tu minigrupu. Ukupna stopa točnosti je srednja vrijednost točnosti svih minigrupa. Na primjer, ako imamo $n = 3$, uzet ćemo tri kandidata koji prema politici snažnog grananja imaju najbolji rezultat i pogledat ćemo najbolju varijablu prema izlazu iz naše mreže. Ako se varijabla koja prema našoj mreži ima najbolji rezultat nalazi u najbolja tri kandidata politike snažnog grananja, označavamo uzorak s brojem jedan. Ako imamo minigrupe veličine 32 uzoraka, stopu točnosti za tu minigrupu izračunavamo kao $(broj\ uzoraka\ označenih\ brojem\ 1)/32$. Postupak ponavljamo za svaku minigrupu i u tablici dajemo ukupnu stopu točnosti - srednju vrijednost točnosti minigrupa. Stopu točnosti smo izračunali za $n = \{1, 3, 5, 10\}$ i napisali u tablici. Tablicu interpretiramo na sljedeći način:

- naš model je u 37.24% slučaja odabrao istog najboljeg kandidata kao i politika snažnog grananja

- naš model je u 58.99% slučaja odabrao najboljeg kandidata koji se nalazi među 3 najbolja kandidata politike snažnog grananja
- naš model je u 68.34% slučaja odabrao najboljeg kandidata koji se nalazi među 5 najbolja kandidata politike snažnog grananja
- naš model je u 79.28% slučaja odabrao najboljeg kandidata koji se nalazi među 10 najbolja kandidata politike snažnog grananja

Možemo vidjeti da je model djelomično naučio imitirati politiku snažnog grananja. Ipak, također možemo uočiti da se mreža nije značajno približila politici snažnog grananja, no preostaje nam vidjeti hoće li uspjeti ubrzati rješavanje instanci problema.

4.2. Evaluacija modela

Evaluirali smo model (grafička konvolucijska neuronska mreža - GKNN) na 34 instanci problema upravljanja lancem opskrbe. Naš model uspoređivali smo sa zadanom politikom grananja u SCIP-u (hibridnom politikom RPB). Dopustili smo u SCIP-u frakcionalni raskorak (engl. *fractional gap*) odnosno odstupanje od optimalnog rješenja od 0.8% za rješavanje problema jer je to razina obično prihvatljiva u primjeni. Za svaku težinu naveli smo prosječno vrijeme izvođenja - *vrijeme*, broj instanci koje su se rješavale kraće koristeći odgovarajući model - *broj pobjeda*, broj stvorenih čvorova tijekom provođenja algoritma - *broj čvorova*. Svaku instancu smo pokrenuli na obje politike grananja, ograničavajući vrijeme rješavanja na sat vremena.

Tablica 4.1: Evaluacija mreže na instancama problema upravljanja lancem opskrbe

Politika grananja	Vrijeme	Broj pobjeda	Broj čvorova
RPB	14.99 $\pm 95.66\%$	17	638.88 $\pm 194.67\%$
GKNN	32.03 $\pm 206.99\%$	17	4785.66 $\pm 239.93\%$

Možemo uočiti da je unatoč jednakom broju pobjeda dvaju algoritama (17 na prema 17), onaj sa zadanom (RPB) politikom grananja nadmoćan u odnosu na algoritam koji koristi našu mrežu kao politiku grananja. Algoritam sa zadanom politikom rješava probleme u prosjeku brže te pri tome generira manje čvorova. Nadalje, možemo uočiti vrlo veliku varijancu u vremenu rješavanja i broju čvorova kod obadva algoritma. To se događa jer su problemi vrlo raznovrsni po težinama rješavanja. Također distribucije

broja čvorova i vremena rješavanja su zakrivljene (engl. *skewed*) udesno zbog malog broja problema koji se jako dugo se rješavaju u odnosu na ostale i značajno povećavaju prosjek i varijancu broja čvorova i vremena rješavanja.

5. Zaključak

U ovom radu, pokušali smo grafovskom konvolucijskom neuronskom mrežom ubrzati moderni rješavač SCIP pri rješavanju problema upravljanja lancem opskrbe.

Opisali smo probleme kombinatorne optimizacija - LP i MILP, algoritam grananja i ograđivanja (B&B) te konvolucijske neuronske mreže i grafovske konvolucijske neuronske mreže.

Zatim smo ušli u srž ovog rada - prikazali smo parametrizaciju stanja i politike grananja u B&B algoritmu, definirali grafovsku konvolucijsku neuronsku mrežu te opisali cjelokupnu metodologiju i postupak treniranja, testiranja i evaluacije našeg modela.

Procjenjivali smo rezultate testiranjem i evaluacijom. Testiranjem modela dobili smo točnost imitiranja politike snažnog grananja 37.24% pri biranju najboljeg kandidata za grananje. To se u evaluaciji nije pokazalo dovoljnim da bismo ubrzali rješavanje problema.

Međutim, možemo predložiti nekoliko poboljšanja. Za početak, trebalo bi pronaći točno koje značajke čvora stabla grananja iz SCIP-a su nam najznačajnije za ovaj konkretan problem te izmijeniti program koji generira uzorke skupljajući informacije iz SCIP-a. Nadalje, mjesta za napredak ima i u modelu grafovske konvolucijske neuronske mreže. Graf-transformerske neuronske mreže pokazale su se uspješne u učenju nad grafovima. S boljim modelom i izvedbom smatramo da bismo mogli postići mnogo bolji učinak te ubrzati i specijalizirati moderne rješavače poput SCIP-a korištenjem imitacijskog strojnog učenja. Ako bismo krenuli putem podržanog učenja, pogotovo *offline* varijante, postoje novi radovi koji postižu značajna poboljšanja rezultata u drugim domenama primjene. Primjer takvog rada je [17].

6. Zahvala

Zahvaljujem se mentoru, doc. dr. sc. Mariju Brčiću na svoj pomoći, usmjeravanju i potpori pri pisanju ovog rada. Također, zahvaljujem se Adrian Alajkoviću i Mihael Končiću iz Konzuma koji su nam donirali potrebne podatke za rad.

LITERATURA

[1] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, Jakob Witzig
The SCIP Optimization Suite 7.0 Available at Optimization Online and as ZIB-Report 20-10, March 2020

[2] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*, 2020. URL `https://arxiv.org/pdf/1811.06128.pdf`. Pristupano: 24. svibnja 2021.

[3] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, Andrea Lodi. *Exact Combinatorial Optimization with Graph Convolutional Neural Networks*, 2019. URL `https://arxiv.org/pdf/1906.01629.pdf`. Pristupano: 24. svibnja 2021.

[4] Jan Šnajder, Bojana Dalbelo Bašic, Marko Ćupić. *Umjetne neuronske mreže*. Zavod za elektroniku, mikroelektroniku i inteligentne sustave, Fakultet elektrotehnike i računarstva, 2008.

[5] Jan Šnajder, Bojana Dalbelo Bašic, Marko Ćupić. *Strojno učenje*. Zavod za elektroniku, mikroelektroniku i inteligentne sustave, Fakultet elektrotehnike i računarstva, 2008.

[6] Liu, Danqing. *'A Practical Guide to ReLU'*. *Medium*, 30.11.2017, `https://medium.com/@danqing/`

a-practical-guide-to-relu-b83ca804f1f7. Pristupano 25. svibnja 2021.

[7] Damir Kopljar. *Konvolucijske neuronske mreže*, ZAVRŠNI RAD br. 4580, 2016.

[8] *Neural Network Models in R* https://www.datacamp.com/community/tutorials/neural-network-models-r. Pristupano 25. svibnja 2021.

[9] Saha, Sumit. *'A Comprehensive Guide to Convolutional Neural Networks — the ELI5 Way'*. Medium, 17.12.2018, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the\-eli5-way-3bd2b1164a53. Pristupano 25. svibnja 2021.

[10] Hui, Jonathan. *'Graph Convolutional Networks (GCN) Pooling'*. Medium, 23. veljače 2021, https://jonathan-hui.medium.com/graph-convolutional-networks-gcn-pooling-839184205692. Pristupano 27. svibnja 2021.

[11] *Markovljevi lanci* https://www.grad.unizg.hr/_download/repository/Markovljevi_lanci.pdf Pristupano 31. svibnja 2021.

[12] *Duboko Učenje.* https://www.fer.unizg.hr/predmet/dubuce. Pristupano 3. lipnja 2021.

[13] Yun, Seongjun, et al. *"Graph Transformer Networks." Advances in Neural Information Processing Systems*, vol. 32, 2019. papers.nips.cc, https://papers.nips.cc/paper/2019/hash/9d63484abb477c97640154d40595a3bb-Abstract.html. Pristupano 28. lipnja 2021.

[14] "TensorFlow." TensorFlow,https://www.tensorflow.org/. Pristupano 30.lipnja 2021.

[15] PyTorch. https://www.pytorch.org. Pristupano 30. lipnja 2021.

[16] Elias B. Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. *Learning to branch in mixed integer programming. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[17] Chen, Lili, et al. *"Decision Transformer: Reinforcement Learning via Sequence Modeling."*, lipanj 2021. http://arxiv.org/abs/2106.01345. Pristupano 30. lipnja 2021.

Primjena strojnog učenja u kombinatornoj optimizaciji za upravljanje lancem opskrbe

Sažetak

Upravljanje lancem opskrbe težak je kombinatorni problem. Egzaktne općenite metode za rješavanje takvih problema dijele se na pristupe bazirane na matematičkom programiranju i na one bazirane na zadovoljenju ograničenja. U ovom radu koristit će se grafovska neuronska mreža za ubrzanje izvođenja izračuna u rješavaču baziranom na matematičkom programiranju. Mreža će se utrenirati imitacijskim učenjem nad odabranim skupom problema upravljanja lancem opskrbe.

Ključne riječi: kombinatorna optimizacija, grafovske konvolucijske neuronske mreže, strojno učenje, algoritam grananja i ograđivanja, linearno programiranje, mješovito cjelobrojno linearno programiranje, upravljanje lancem opskrbe

Towards applications of machine learning in combinatorial optimization for supply chain management

Abstract

Supply chain management is a difficult combinatorial problem. Exact general methods for solving such problems are divided between mathematical programming and those based on meeting constraints. In this paper, graph-convolution neural network will be used for performance acceleration of solver based on mathematical programming. The network will be trained by imitation learning over a selected set of instances of supply chain management problems.

Keywords: combinatorial optimization, graph-convolution neural network, machine learning, branch-and-bound algorithm, B&B, linear programming, mixed-integer linear programming, supply chain management