Primjena teorije grafova u umjetnoj inteligenciji

Licul, Manola

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:196:397542

Sveučilište u Rijeci
Fakultet za matematiku

Preddiplomski studij Matematika

Manola Licul

Primjena teorije grafova u umjetnoj inteligenciji

Završni rad

Rijeka, rujan 2022.

Sveučilište u Rijeci
Fakultet za matematiku

Preddiplomski studij Matematika

Manola Licul

Primjena teorije grafova u umjetnoj inteligenciji

Mentor: dr.sc. Andrea Švob

Završni rad

Rijeka, rujan 2022.

Sadržaj

Sažetak

U ovom radu upoznat ćemo se s nekim oblicima umjetne inteligencije te vidjeti povezanost s teorijom grafova. Uvest ćemo pojam umjetne inteligencije te kroz nekoliko primjera vidjeti kako je već koristimo u svakodnevnom životu, a da toga možda nismo ni svjesni. Zatim ćemo se upoznati s nekim algoritmima pretraživanja prostora, te na primjerima pokazati njihove razlike. Kroz teoriju grafova objasnit ćemo pretraživanje u širinu, pretraživanje s jednolikom težinom te pretraživanje u dubinu. Na primjeru putovanja kroz Istru provest ćemo svaki od navedenih algoritama te vidjeti razlike u dobivenim rezultatima.

1 Uvod

Umjetna inteligencija jedno je od novijih područja znanosti čiji je naziv osmišljen 1956. godine. Obuhvaća različita područja: opće područje, ekspertni sustavi, dedukcija i dokazivanje, automatsko programiranje, strojno učenje, rješavanje problema, metode upravljanja, metode pretraživanja, robotika, računalni vid, itd. Neki od primjera umjetne inteligencije su:

- Autonomni automobili - potrebna je kombinacija više tehnika umjetne inteligencije; pretraživanje i planiranje puta, računalni vid, sposobnost donošenja odluka kako bi se izbjegle neželjene posljedice. Cilj je poboljšanje sigurnosti na cestama te učinkovitost logističkih lanaca u prijevozu robe, pa bi u tom slučaju čovjek preuzeo ulogu nadzora dok bi za vožnju bili zaduženi automobili. Ovakve tehnike koriste i autonomni brodovi, bespilotne letjelice, roboti dostavljači i slični autonomni sustavi.

- Preporuka sadržaja - algoritmi posebno osmišljeni kako bi svaki korisnik dobio personalizirane preporuke sadržaja. Naime, na bilo kojoj društvenoj mreži ili internetskom pretraživaču algoritmi su zaslužni za oglase, preporuke glazbe, filmova i sličnih sadržaja.

- Obrada slika i videozapisa - sustav za prepoznavanje lica ima sve veće primjene (korisničke, administrativne i poslovne). Automatsko označavanje osoba na društvenim mrežama, grupiranje albuma prema osobama, provjera putovnica samo su neke od mogućnosti koje pruža prepoznavanje lica. Posljedice unaprjeđenja tehnika su izrada lažnih videozapisa gdje će biti nemoguće razlikovati ih od stvarnih videozapisa.

1.1 Povijest umjetne inteligencije

Za razvoj umjetne inteligencije zaslužne su različite grane znanosti; filozofija, matematika, ekonomija, neuroznanost, psihologija, računarstvo. Prvi rad priznat kao umjetna inteligencija napisali su Warren McCulloch i Walter Pitts 1943. godine. Predložili su model umjetnih neurona i načine na koji definirane mreže mogu učiti. Zatim je Donald Hebb 1949. godine nadogradio model za modificiranje snage veze između neurona koje danas zovemo Hebijansko učenje. Marvin Minsky i Dean Edmonds zaslužni su za prvo računalo neuronske mreže SNARC. Bilo je i ranijih radova koji se mogu

okarakterizirati kao umjetna inteligencija. Jedan od značajnijih je 1947. predavanje Alana Turinga Londonskom matematičkom društvu gdje je prezentirao svoj rad "Računalni strojevi i inteligencija". Uveo je Turingov test, strojno učenje te genetski algoritam. Rođenjem umjetne inteligencije smatra se 1956. godina kada je na Darthmouth College-u organizirana dvomjesečna radionica na kojoj su sudjelovali znanstvenici iz područja automatike, neuronskih mreža i znanstvenici koji se bave proučavanjem inteligencije kako bi pronašli način kako natjerati strojeve da rješavaju probleme. Istaknuli su se Allen Newell i Herbert Simon koji su razvili Logic Theorist, program za automatsko rasuđivanje koji sam može izvoditi logičke teoreme. John McCarthy 1958. definira programski jezik Lisp koji postaje dominantan sljedećih 30 godina. McCarthy je objavio članak "Programi sa zdravim razumom" kojim je opisao Advance Tracker, hipotetski program koji se može smatrati prvim kompletnim UI programom. U narednim godinama dolazi do mnogih uspjeha na ovom području; računalo koje je u šahu pobijedilo svjetskog prvaka, programi za dokazivanje matematičkih teorema, programa koji simulira terapeuta u konverzaciji s klijentom i slično. 80-ih godina 20.st. UI se naglo razvija kroz ekspertne sustave, sustave vizije, robote, softvere i hardvere specijalizirane za unaprjeđenje umjetne inteligencije. Od 90-ih godina vidljiv je veliki napredak u robotici, računalnom vidu, strojnom učenju i reprezentaciji znanja. Danas je sve više aplikacija koje svakodnevno koristimo, a iza kojih stoji upravo umjetna inteligencija.

1.2 Definicija umjetne inteligencije

Svakodnevno se nesvjesno susrećemo s umjetnom inteligencijom, ali se rijetko zapitamo što je to zapravo. Neki smatraju da je umjetna inteligencija umjetan oblik života koji će nadmašiti ljudsku inteligenciju, dok drugi smatraju da je to svaka vrsta tehnologije za obradu podataka.

"Umjetna inteligencija je naziv za znanstvenu disciplinu koja se bavi izgradnjom računalnih sustava čije se ponašanje može tumačiti kao inteligentno." John McCarthy (1956.)

"Proučavanje postupaka koji mogućim čine percipiranje, rasuđivanje i reagiranje." Patrick H. Winston

"Znanost o tome kako postići da strojevi izvode zadatke koji bi, kada bi

ih radio čovjek, iziskivali inteligenciju." Marvin Minsky

"Umjetna inteligencija bavi se izučavanjem kako računalo učiniti sposobnim da obavlja poslove koje u ovom trenutku ljudi obavljaju bolje." Elain Rich

Nema opće prihvaćene definicije umjetne inteligencije. Razlog tome je što se određene teme prestaju svrstavati u područje istraživanja umjetne inteligencije, dok se nove pojavljuju. Također, definicije se mogu svrstati u 4 kategorije.

- Razmišljati ljudski
  Želimo li da računalo razmišlja kao čovjek, moramo najprije shvatiti kako ljudi misle. Postoje tri načina da shvatimo kako radi ljudski um: kroz introspekciju opažanjem vlastitih misli, kroz psihološke eksperimente i kroz slikanje mozga odnosno promatranje mozga dok obavlja neke zadatke. Zatim treba sve dobivene rezultate izrazit kao računalni program. Kognitivne znanosti spajaju računalne modele iz UI te eksperimentalne tehnike iz psihologije za konstruiranje ljudskog uma.

- Razmišljati racionalno
  Grčki filozof Aristotel bio je među prvima koji je pokušao definirati "ispravno mišljenje". Time je pokrenuta znanstvena disciplina zvana logika. Tendencija logike unutar umjetne inteligencije je graditi programe za stvaranje inteligentnih sustava. Dvije su glavne prepreke; moramo uzeti neformalno znanje i navesti ga u formalnim terminima koje zahtjeva logička notacija, te velika razlika između rješavanja problema u teoriji i u praksi.

- Ponašati se ljudski
  Turingov test zamišljen je kao postupak nakon kojega je jasno je li stroj inteligentan ili nije. Računalo prolazi pisani test na način da komunicira s čovjekom te ukoliko čovjek ne raspoznaje je li s druge strane čovjek ili računalo smatra se da je test uspješno savladan odnosno, da je računalo inteligentno. Da bi računalo prošlo test, trebalo bi posjedovati sljedeće mogućnosti: obrada prirodnog jezika kako bi se ostvarila uspješna komunikacija; predstavljanje znanja kako bi pohranio sve što čuje ili vidi; automatsko zaključivanje za korištenje odgovora na pitanje; strojno učenje za prilagodbu novim okolnostima. Turingov test je

izbjegavao fizičku interakciju ispitivača i računala jer je fizička simulacija nepotrebna za inteligenciju. Totalni Turingov test uključuje video signal kako bi se mogle testirati perceptivne sposobnosti i kretanje, pa je za to još potrebno računalni vid i robotika.

- Ponašati se racionalno
  Agent je bilo koji sustav koji promatra okolinu kroz osjetila (senzore) i raznim akcijama (efektorima) djeluje na okolinu. Racionalni agent je onaj koji djeluje tako da postigne najbolji ishod ili, kada postoji neizvjesnost, najbolji očekivani ishod. Racionalnost ovisi o sljedećim elementima: mjera uspješnosti koju definira kriterij uspjeha, agentovo poznavanje okoline, akcije koje agent može izvoditi i agentov opažajni niz. Racionalni agent je onaj koji za svaki opažajni niz bira onu akciju za koju očekuje da će maksimizirati mjeru uspješnosti s obzirom na podatke koji su dani opažajnim nizom i agentovim ugrađenim znanjem.

S umjetnom inteligencijom usko su povezana još neka područja. Strojno učenje je grana umjetne inteligencije koja se temelji na ideji da sustavi mogu učiti iz podataka, odnosno metoda analize podataka koja automatizira izradu analitičkog modela. Duboko učenje je grana strojnog učenja koja stvara modele za izvršavanje zadataka. Znanost o podacima obuhvaća strojno učenje i statistiku, ali i dio računarstva, uključujući izradu internetskih aplikacija, pohranu podataka i algoritme.

1.3 Osnovni pojmovi o grafovima

Definicija 1.1 *Graf G je uređena trojka G=(V(G),E(G),ψ_G) koja se sastoji od nepraznog skupa V(G), čiji su elementi vrhovi od G, skupa E(G), disjunktnog sa V(G) čiji su elementi bridovi od G i funkcije incidencije ψ_G koja svakom bridu od G pridružuje neuređeni par (ne nužno različitih) vrhova od G. Ako je $e \in E(G)$ i ako su $u, v \in V(G)$ te ako je $\psi_G(e) = uv$ kažemo da e spaja vrhove u i v i da su u i v krajevi od e.*

Definicija 1.2 *Šetnja u grafu G je niz $W = v_0e_1v_1e_2...e_kv_k$ čiji članovi su naizmjence vrhovi v_i i bridovi e_i, tako da su krajevi od e_i vrhovi v_{i-1} i v_i, $1 \leq i \leq k$. Šetnja W je zatvorena ako je $v_0 = v_k$. Ako su svi bridovi $e_1, e_2, ..., e_k$ šetnje W međusobno različiti, onda se W zove staza. Zatvorena staza pozitivne duljine čiji su vrhovi (osim krajeva) međusobno različiti zove se ciklus.*

Definicija 1.3 *Povezan acikličkи graf nazivamo stablo.*

Definicija 1.4 *Graf je povezan ako su svaka dva njegova vrha povezana nekim putom.*

U teoriji grafova važno je naći cikluse i stabla zbog primjene njihovih svojstava. Za pronalazak ciklusa i stabla u grafu služimo se različitim algoritmima, međutim u praksi je važna i brzina pronalaska, odnosno tražimo brze i efikasne algoritme. Složenost algoritma je vrijeme njegove provedbe. Samim time algoritme možemo podijeliti prema njihovoj složenosti pa imamo polinomske i eksponencijalne algoritme. Polinomske algoritme smatramo efikasnim jer rastom ulaznih podataka, vrijeme njegove provedbe raste relativno sporo, dok eksponencijalni algoritmi s malim brojem ulaznih podataka rade jako dugo.

Definicija 1.5 *Traženo stablo u povezanom grafu s n vrhova je razapinjuće stablo čiji su skupovi vrhova $\{v_0, v_1, ..., v_{n-1}\}$ i bridova $\{e_1, e_2, ..., e_{n-1}\}$ uređeni tako da je v_0 korijen, a za svako j, $1 \leq j \leq n-1$, podgraf sa skupom vrhova $\{v_0, v_1, ..., v_j\}$ i skupom bridova $\{e_1, e_2, ..., e_j\}$ čini stablo. Drugim riječima, traženo stablo je proces rasta razapinjućeg stabla, a ne samo završno stablo. Počinjemo s korijenom v_0, a stablo raste tako da u svakom koraku dodamo vrh v_j i brid $e_j = v_i v_j$. Pri tome se v_j zove sin od v_i, a v_i otac od v_j.*

Definicija 1.6 *Usmjereni graf ili digraf D je graf G u kojem svaki brid ima smjer od početka prema kraju. D se još zove orijentacija od G i pišemo $D = \overrightarrow{G}$. Brid s početkom u, a krajem v je uređeni par (u, v) i katkad pišemo $u \to v$. Kaže se još da je $a = uv$ luk od u prema v.*

2 Algoritmi pretraživanja prostora

Znanosti se zasnivaju na rješavanju određenih problema, pa samim time je cilj i umjetne inteligencije rješavanje određenih problema. Pretraživanjem prostora možemo riješiti niz analitičkih problema. Opisat ćemo neke od algoritama za pretraživanje prostora pomoću kojih možemo doći do korisnih informacija odnosno pokazat ćemo na primjerima u kojim situacijama se koriste takvi algoritmi. Zadatak Umjetne inteligencije kao znanosti (u daljnjem tekstu UI) je dizajnirati program koji će implementirati funkciju agenta, odnosno mi ćemo pretraživanjem prostora stanja odrediti redosljed izvođenja radnji.

Neka je S skup stanja tj. prostor stanja. Svako pretraživanje prostora stanja sastoji se od: $problem$=(s_0,$succ$,$goal$) gdje je s_0 početno stanje, $succ : S \to \rho(S)$ je funkcija sljedbenika koja definira prijelaze između stanja (može se definirati pomoću skupa operatora), $goal : S \to \{\top, \bot\}$ je ispitni predikat istinit samo za ciljna stanja (true, false). Rješenje problema je niz akcija koje vode od početnog do ciljnog stanja. Pretraživanje prostora stanja se svodi na pretraživanje usmjerenog grafa (digrafa), pri čemu su vrhovi grafa stanja, dok su lukovi prijelazi između stanja. Ukoliko su definirane vrijednosti prijelaza, odnosno težine, govorimo o usmjerenom težinskom grafu.

Svojstva algoritama: potpunost, optimalnost, vremenska složenost i prostorna složenost. Algoritme pretraživanja prostora možemo podijeliti na sljedeći način:

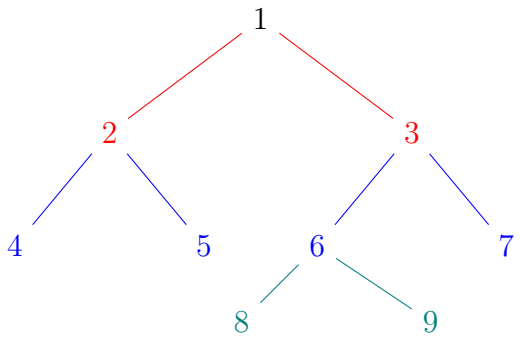
- Slijepo pretraživanje - metode koje nemaju dodatne informacije o stanjima osim onih navedenih u definiciji problema. Takve metode mogu samo generirati nove nasljednike i razlikovati ciljno od neciljnog stanja, a razlikuju se po redoslijed širenja čvorova.
  - Pretraživanje u širinu
  - Pretraživanje s jednolikom težinom
  - Pretraživanje u dubinu
  - Ograničeno pretraživanje u dubinu
  - Iterativno pretraživanje u dubinu
  - Dvosmjerno pretraživanje

- Usmjereno (ili heurističko) pretraživanje - Heuristička funkcija $h : S \to \mathbf{R}^+$ svakom stanju $s \in S$ pridružuje procjenu udaljenosti od tog stanja do ciljnog stanja. Što je vrijednost $h(s)$ manja, to je vrh s bliži ciljnom stanju. Ako je s ciljno stanje, $h(s) = 0$. Takve metode poboljšavaju učinkovitost pretraživanja odnosno smanjuju prostor pretraživanja.
  - Pretraživanje "najbolji prvi"
  - Algoritam A*
  - Pretraživanje usponom na vrh

U nastavku ćemo posebnu pažnju posvetiti pretraživanju u širinu, pretraživanju s jednolikom težinom i pretraživanju u dubinu. Ovim algoritmima pretražujemo prostor stanja te time dolazimo do stabla pretraživanja. Bitno je napomenuti da stablo pretraživanja može biti i beskonačno iako je prostor konačan, naime ako u grafu imamo cikluse stablo pretraživanja je beskonačno.

2.1 Pretraživanje u širinu

Dijkstrin algoritam je prvi dokumentirani algoritam, danas ga nazivamo pretraživanje u širinu (breadth-first search tree). U ovom je algoritmu traženo stablo takvo da za dane vrhove $\{v_0, v_1, ..., v_{n-1}\}$ i bridove $\{e_1, e_2, ..., e_{n-1}\}$, otac od v_j vrh v_i s najmanjim mogućim indeksom i gdje su v_j i e_j tako odabrani za svaki j, $1 \leq j \leq n-1$. Budući da je udaljenost između korijena i svakog vrha u stablu jednaka njihovoj udaljenosti u grafu, možemo odrediti najkraći put među svim parovima vrhova u grafu, a samim time i dijametar grafa. Ovaj algoritam izvodi se tako da se prvo proširuje korijenski vrh, a zatim sva njegova djeca, pa njihova djeca, itd. Svi vrhovi se proširuju na zadanoj razini prije nego se bilo koji vrh proširi na sljedećoj razini. Ukoliko je cijena prijelaza konstantna, ova metoda je optimalna. Veći problem kod pretraživanja u širinu je zahtjev za memorijom nego vrijeme izvršenja.

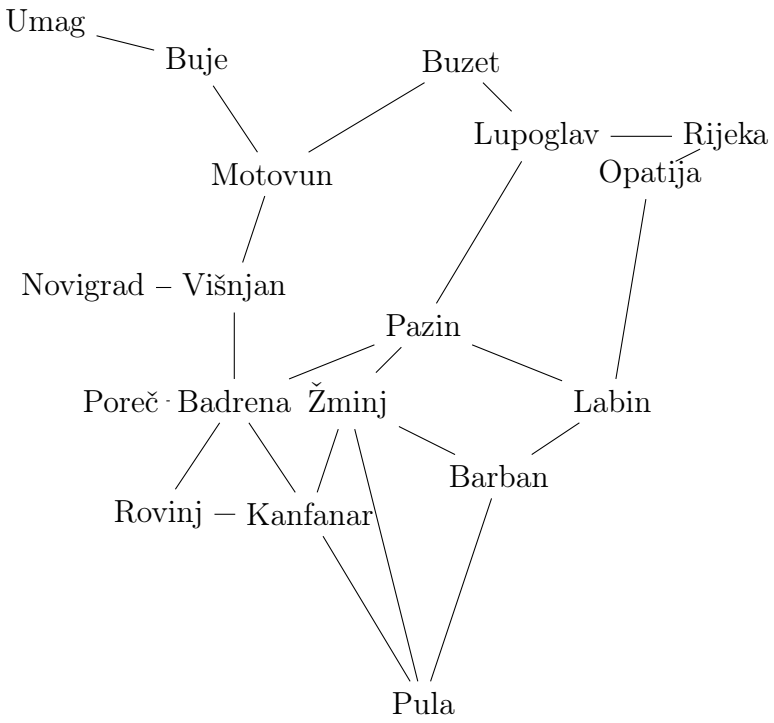


Slika 1: Algoritam pretraživanja u širinu

Pretraživanje u širinu koristimo kod traženja minimalnog razapinjućeg stabla i ciklusa. Također, ovaj algoritam ima primjenu u umrežavanju računala bez poslužitelja (peer to peer mreže). Pretraživači u tražilicama rade po principu pretraživanja u širinu tako da krećemo od izvorne stranice te pratimo linkove s kojima je povezana. Traženje osoba na društvenim mrežama odnosno prijedlozi koje dobivamo također se zasnivaju na ovakvom pretraživanju. Još jedna od primjena ovog algoritma je GPS odnosno sustav navigacije.

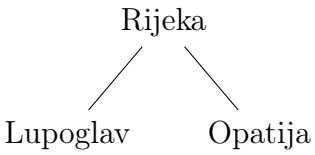
2.1.1 Primjer pretraživanja u širinu

Želimo od Rijeke doći do Rovinja. Tražimo najkraći put pomoću pretraživanja u širinu. Na grafu su nam vrhovi gradovi, a bridovi ceste kojima su gradovi povezani.



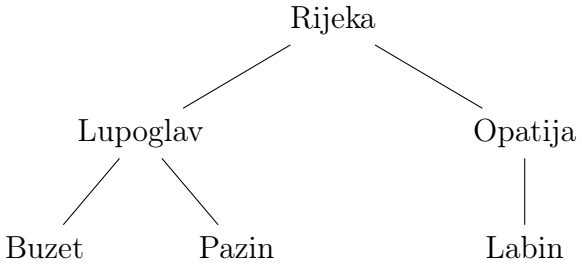
Slika 2: Putovanje kroz Istru

Za korijenski vrh uzet ćemo Rijeku te graditi stablo na sljedeći način. Djeca od korijenskog vrha su Lupoglav i Opatija.



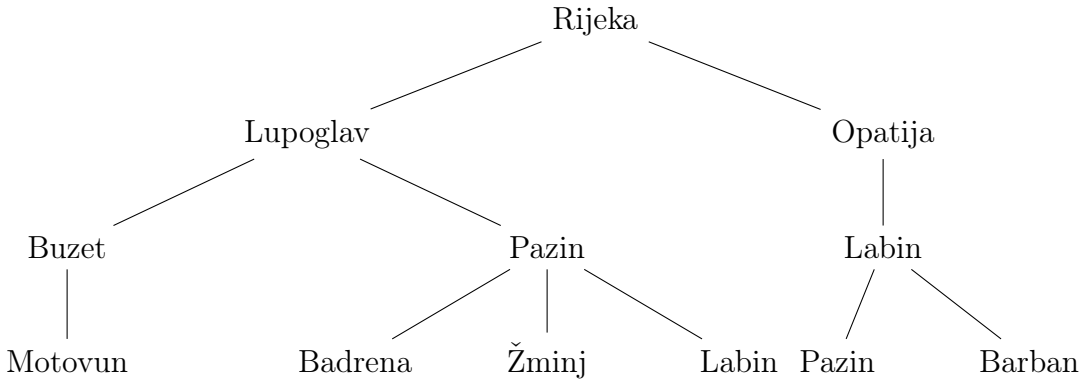
Slika 3: 1. korak pretraživanja u širinu

Kako nismo došli do ciljnog vrha u prvom koraku, nastavljamo tako da svaki dobiveni vrh proširimo na njihovu djecu odnosno, djeca od Lupoglava su Buzet i Pazin, a od Opatije samo Labin.



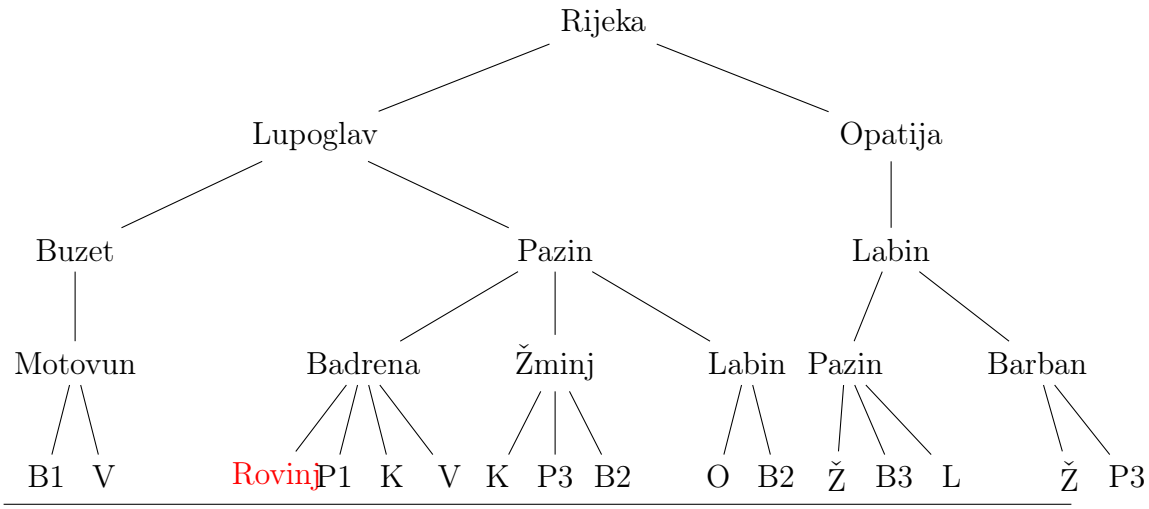
Slika 4: 2. korak pretraživanja u širinu

Budući da nismo došli do ciljnog vrha ni u drugom koraku, nastavljamo isti postupak; dijete od Buzeta je samo Motovun, od Pazina Baderna, Žminj i Barban, od Labina Pazin i Barban.



Slika 5: 3. korak pretraživanja u širinu

Zatim proširujemo stablo na njihovu djecu. Djeca od Motovuna su Buje i Višnjan, od Baderne Rovinj, Poreč, Kanfanar i Višnjan, od Žminja Kanfanar, Pula i Barban, od Labina Barban i Opatija, od Pazina Žminj, Baderna i Lupoglav, te od Barbana Žminj i Pula. Primijetimo da je moguće doći do ciklusa odnosno beskonačne petlje. Npr. djeca od Barbana su Žminj i Pula, zatim djeca od Pule su Žminj i Kanfanar, od Žminja Kanfanar, Barban i Pazin, nastavimo li s djecom od Barbana dolazimo do petlje Barban-Pula-Žminj. Vidimo da je dijete od Baderne Rovinj te smo time došli do ciljnog vrha u ovom koraku. Time smo našli najkraći put koji je Rijeka-Lupoglav-Pazin-Baderna-Rovinj.

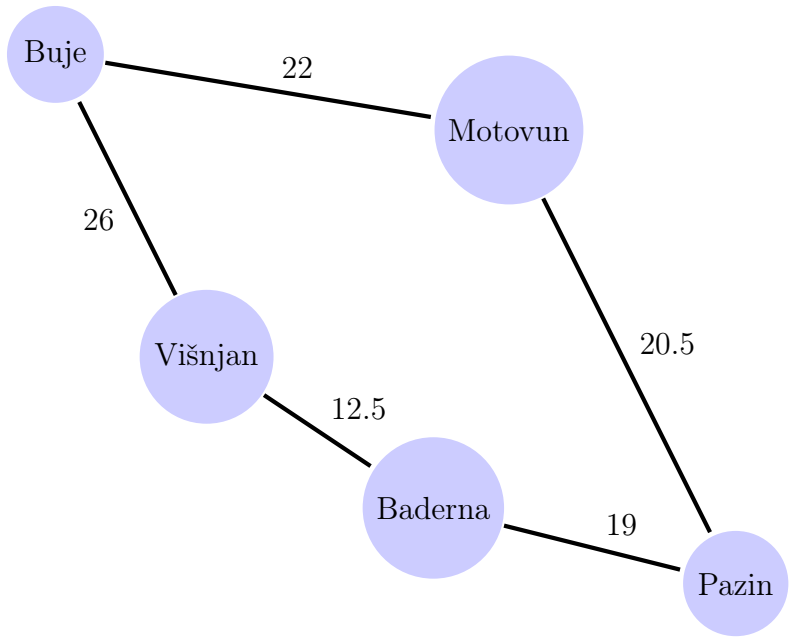


Slika 6: 4. korak pretraživanja u širinu

2.2 Pretraživanje s jednolikom težinom

Ukoliko bridovi u grafu nemaju težinu odnosno svi su jednake težine, tada je pretraživanje u širinu optimalno. Međutim, uzmemo li u obzir težine bridova tada dolazimo do algoritma koji nazivamo pretraživanje s jednolikom težinom. Izvodi se analogno kao pretraživanje u širinu tako da se proširuje brid s najnižom cijenom.

2.2.1 Primjer pretraživanja s jednolikom težinom



Slika 7: Graf udaljenosti između traženih gradova

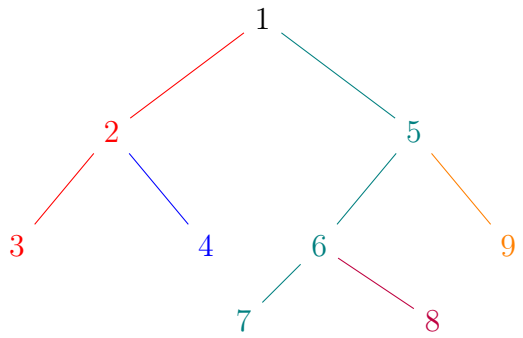
Želimo od Pazina doći do Buja. Gradovi su povezani cestama različitih udaljenosti. Vrhovi grafa bit će gradovi kojima moramo proći kako bi došli od Pazina do Buja, a udaljenost između gradova biti će nam težina brida u grafu. Krećemo od Pazina te uspoređujemo udaljenosti s gradovi s kojima je Pazin povezan, te uzimamo onaj grad koji ima manju udaljenosti. U našem slučaju Pazin je povezan s Motovunom i Badernom. Budući je udaljenost do Motovuna 20.5km, a do Baderne 19km, kraći put nam je do Baderne. Zatim od Baderne gledamo s kojim gradovima je ona povezana te uspoređujemo udaljenosti tako da od grada Pazina pribrojimo udaljenosti do novih gradova i uspoređujemo skupa s Motovunom. Budući je Baderna povezana s Višnjanom, te je udaljenost među njima 12.5km, zbrojimo udaljenosti te dobivamo 19+12.5=31.5km. Budući je $20.5 < 31.5$, sada gledamo udaljenost od Motovuna do Buja budući su oni direktno povezani. Pribojimo li udaljenost od Motovuna do Buja, koja iznosi 22km dobivamo da je tražena udaljenost od Pazina do Buja 42.5km. Međutim, algoritam nastavljamo kako bi usporedili udaljenosti drugim putem. Udaljenost od Višnjana do Buja je 26km, pribo-

jimo li to udaljenosti od Pazina do Višnjana dobivamo 31.5+26=57.5km što je više od 42.5, pa zaključujemo da je najkraći put Pazin-Motovun-Buje.

Primijetimo da su težine uvijek pozitivne vrijednosti, pa samim time putovi nikada ne postaju kraći. Time u svakom koraku nalazimo optimalan put do svakog od vrhova. Ovakvo pretraživanje ne vodi računa o broju koraka već samo o ukupnoj težini pa je moguće da se nađe u beskonačnoj petlji ukoliko postoji put s beskonačnim nizom jako male težine. Takvu situaciju možemo spriječiti ako zadamo da težina uvijek mora biti veća od zadanog ϵ.

2.3 Pretraživanje u dubinu

Pretraživanje u dubinu uveo je prvi Tarjan. U ovom algoritmu traženo stablo je takvo da za dane vrhove $\{v_0, v_1, ..., v_{n-1}\}$ i bridove $\{e_1, e_2, ..., e_{n-1}\}$, otac od v_j vrh v_i s najvećim mogućim indeksom i gdje su v_j i e_j tako odabrani za svaki j, $1 \leq j \leq n-1$. Pretraživanje u dubinu, kao i u širinu, primjenjivo je na konačnom prostoru. Ukoliko je prostor pretraživanja beskonačan može doći do beskonačne petlje pa takvo rješenje nije optimalno. Prednost u odnosu na pretraživanje u širinu je složenost prostora. Budući da zauzima puno manje prostora od ostalih algoritama, jedno je od osnovnih algoritama u mnogim područjima UI.



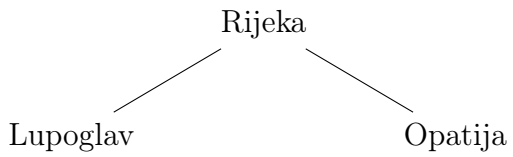
Slika 8: Algoritam pretraživanja u dubinu

Najveća primjena pretraživanja u dubinu je topološko sortiranje koje ima mnogo primjena. Topološkim sortiranjem možemo rasporediti poslove iz za-

danih ovisnosti među poslovima, redoslijed evaluacije formule kod ponovnog računanja proračunskih tablica ili određivanje redosljeda zadataka kompilacije za izvođenje kodova u programu. Također, ovaj algoritam koristi se u serijalizaciji podataka odnosno procesu generiranja i korištenja kodova ili serijskih brojeva.

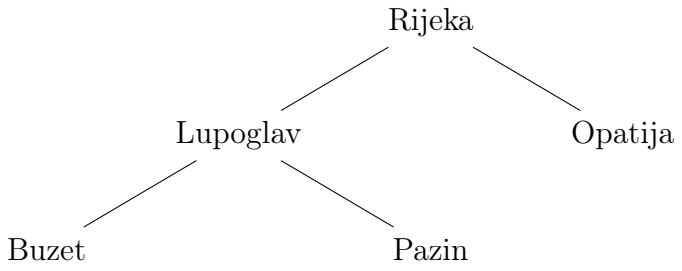
2.3.1 Primjer pretraživanja u dubinu

Za primjer pretraživanja u dubinu uzet ćemo isti primjer kao za pretraživanje u širinu kako bismo mogli usporedit ta dva algoritma. Dakle, cilj nam je od Rijeke stići do Rovinja, te su nam gradovi povezani kao na slici 2. Počinjemo od korijenskog vrha odnosno početne pozicije što nam je Rijeka. U prvom koraku stablo proširujemo na Lupoglav i Opatiju, odnosno djecu od korijenskog vrha.



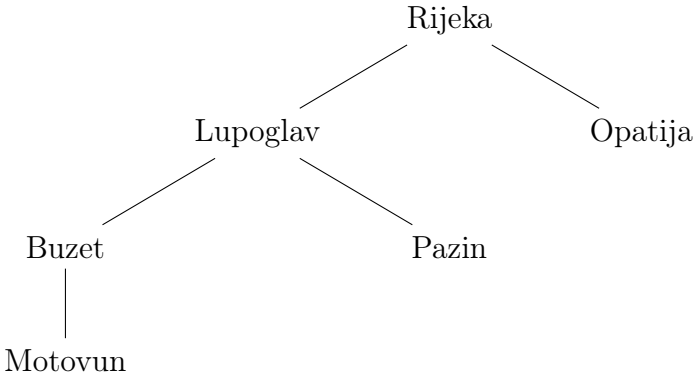
Slika 9: 1. korak pretraživanja u dubinu

Zatim čvor Lupoglav proširujemo na njegovu djecu odnosno Buzet i Pazin.



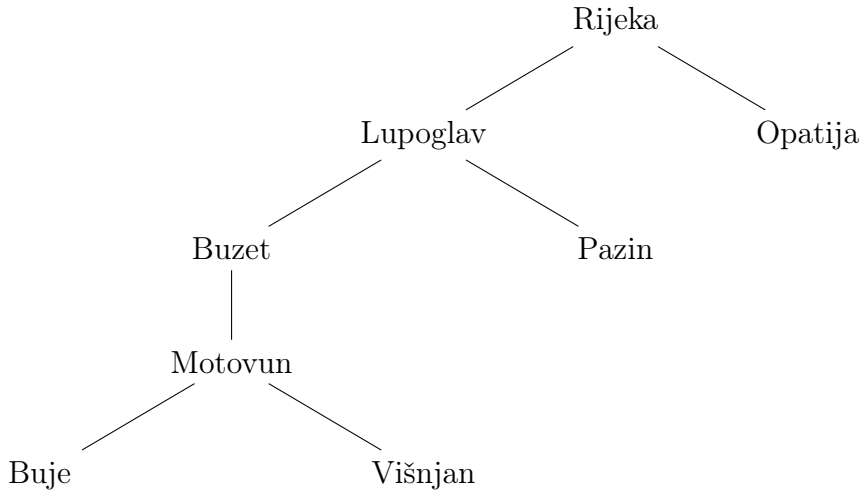
Slika 10: 2. korak pretraživanja u dubinu

Za razliku od pretraživanja u širinu, ne proširujemo vrh Opatija, već nastavljamo s vrhom Buzet. Budući da je jedino dijete od Buzeta Motovun, dolazimo do vrha Motovun te njega proširimo.



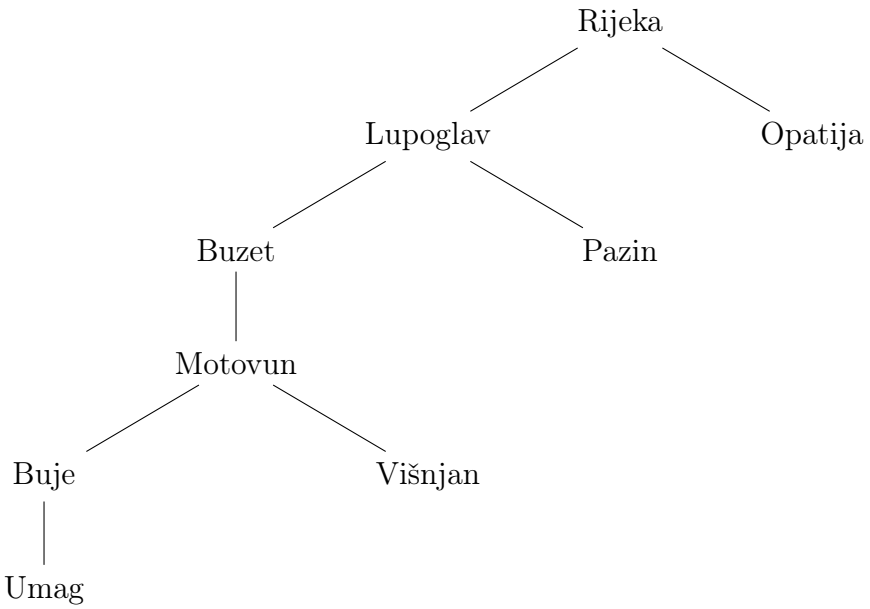
Slika 11: 3. korak pretraživanja u dubinu

Nastavljamo s vrh Motovun te proširujemo stablno na njegovu djecu odnosno, Buje i Višnjan.



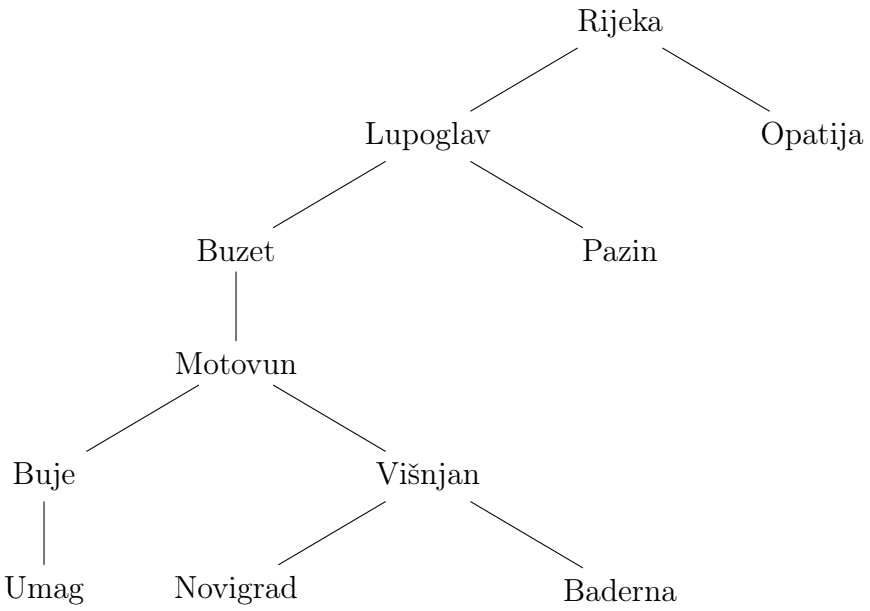
Slika 12: 4. korak pretraživanja u dubinu

U sljedećem koraku dodajemo vrh Umag, budući je to dijete od Buja, međutim kako Umag nema djece, prelazimo na vrh Višnjan.



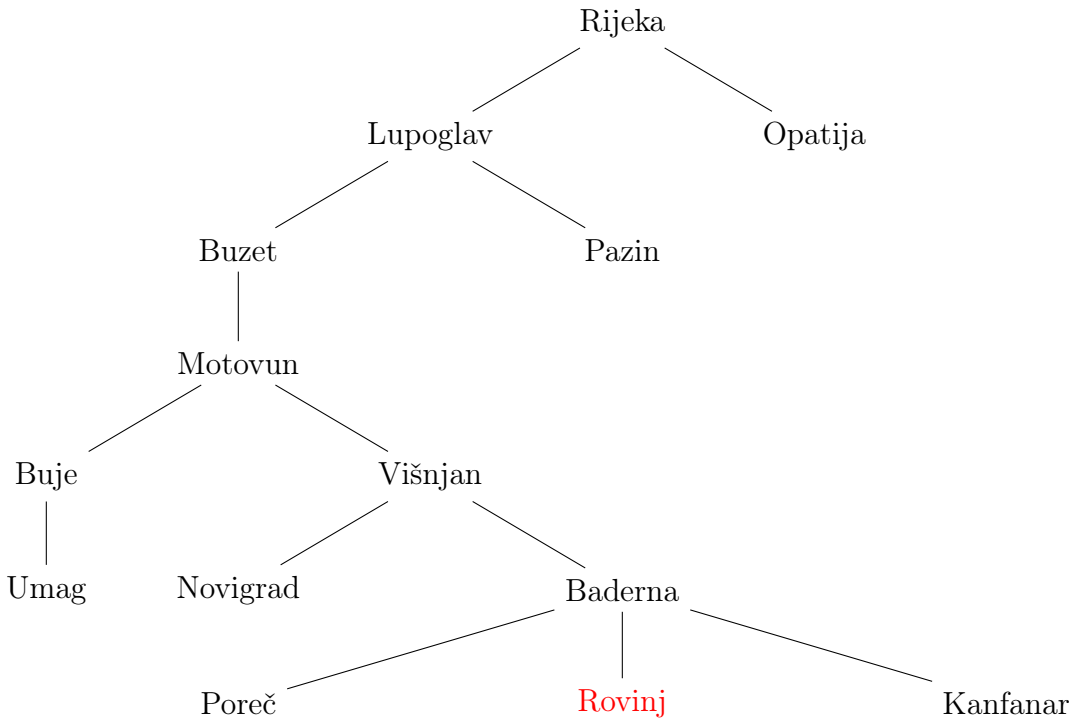
Slika 13: 5. korak pretraživanja u dubinu

Djeca od Višnjana su Novigrad i Baderna. Kako Novigrad nema djece, prelazimo na Badernu. Kao što vidimo ovaj algoritam nam je u ovom slučaju duži nego pretraživanje u širinu.



Slika 14: 6. korak pretraživanja u dubinu

U posljednjem koraku, proširujemo vrh Baderna na Poreč, Rovinj i Kanfanar. Kako je Rovinj bio traženi vrh, dobili smo put od Rijeke do Rovinja koji je: Rijeka-Lupoglav-Buzet-Motovun-Višnjan-Baderna-Rovinj.



Slika 15: 7. korak pretraživanja u dubinu

Kao što vidimo pretraživanje u dubinu nam je u ovom slučaju duži algoritam. U pretraživanju u dubinu lako dolazi do ciklusa ako ne ograničimo algoritam na ponovljena stanja, odnosno ako se grad već pojavljuje u stablu, ne ponavljamo ga. Dakle, ovakvo pretraživanje nije optimalno, ali može biti korisno u nekim situacijama.

3 Zaključak

U ovom radu opisali smo što je umjetna inteligencija te vidjeli kako iza mnogih oblika umjetne inteligencije stoji teorija grafova. Za određivanje redosljeda obavljanja radnji koristimo različite oblike pretraživanja. Svaki od tih pretraživanja svodi se na nalaženje traženog stabla. U radu su posebno opisani algoritmi pretraživanja u širinu, pretraživanja s jednolikom težinom i pretraživanja u dubinu. Na primjeru putovanja kroz Istru usporedili smo navedene algoritme te vidjeli da dolazimo do različitih rješenja. Primjena pretraživanja javlja se u različitim oblicima umjetne inteligencije. U računarstvu su česti problemi traženja optimalnog rješenja, pa se neki od problema mogu prikazat u obliku grafa te pomoću različitih algoritama dolazimo do optimalnog rješanja tako da uspoređujemo brzinu ili složenost izvedbe.

Popis slika

Literatura

[1] Bilješke s predavanja kolegija Umjetna inteligencija, Fakultet za matematiku, Svučilište u Rijeci, Akademska godina 2021./2022.

[2] Elements od AI, URL: https://course.elementsofai.com/hr/ (25.03.2022.)

[3] GeeksforGeeks: Applications of Breadth First Traversal, URL: https://www.geeksforgeeks.org/applications-of-breadth-first-traversal/?ref=rp (25.03.2022.)

[4] GeeksforGeeks: Applications of Depth First Search, URL: https://www.geeksforgeeks.org/applications-of-depth-first-search/?ref=rp (25.03.2022.)

[5] Russel Stuart - Norvig, Peter: *Artificial intelligence : a modern approach*, Pearson, Edinburg, 2016.

[6] Veljak, Darko: *Kombinatorna i diskretna matematika*, Algoritam, Zagreb, 2001.