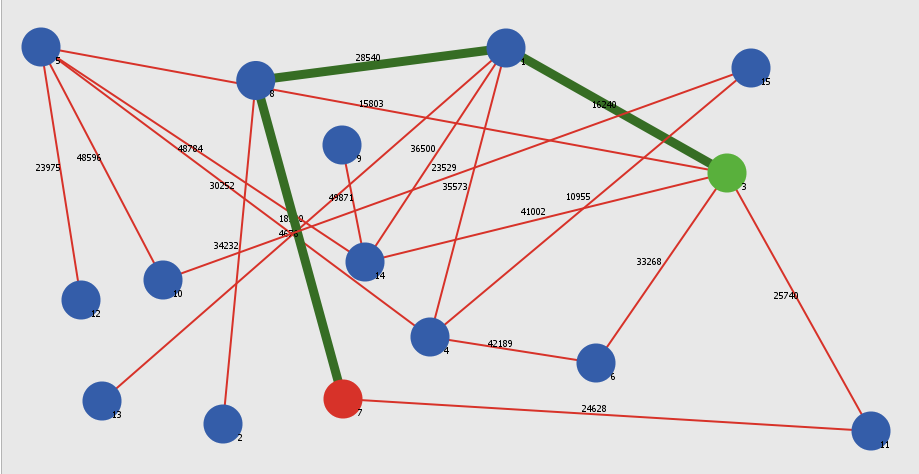
**SVEUČILIŠTE U SPLITU**

**PRIRODOSLOVNO-MATEMATIČKI FAKULTET**

**Mate Ćorić**

**PROBLEM NAJKRAĆEG PUTA U**

**JEDNOSTAVNIM TEŽINSKIM GRAFOVIMA**

**UVOD U UMJETNU INTELIGENCIJU**

**Split, siječanj 2018.**

**Sadržaj**

[**1. Opis problema** 1](#_Toc505413970)

[**2. Realizacija projekta, svijet i agenti** 2](#_Toc505413971)

[**3. Algoritmi pretrage** 4](#_Toc505413972)

[**3.1. Pretraga po dubini** 4](#_Toc505413973)

[**3.2. Pretraga po širini** 5](#_Toc505413974)

[**3.3. Pohlepno pretraživanje** 6](#_Toc505413975)

[**3.4. Tabu pretraga** 6](#_Toc505413976)

[**3.5. Algoritam jednolikog troška** 8](#_Toc505413977)

[**4. Usporedba u Behavior space-u za potpuni graf** 10](#_Toc505413978)

[**5. Usporedba u Behavior space-u za random graf** 13](#_Toc505413979)

[**6. Zaključak** 14](#_Toc505413980)

[7. Literatura 15](#_Toc505413981)

# **1. Opis problema**

U ovom projektu napravljena je simulacija u alatu Net-Logo u kojoj se radi o problemu najkraćeg puta između 2 vrha u jednostavnim težinskim grafovima.

Ideju za projekt dobio sam proučavajući postojeće modele u Net-Logu (Networks). Projekt je zamišljen tako da je u njemu iskorištena većina algoritama pretrage obrađenih u okviru kolegija Uvod u umjetnu inteligenciju uz jedan dodatni algoritam. Dani problem najkraćeg puta riješen je po principima umjetne inteligencije, odnosno uz pomoć 4 komponente: početno stanje (početni vrh), opis mogućih akcija u algoritmima pretrage, test na ciljno stanje i cijena puta između stanja.

Dani problem: Za jednostavni težinski graf s n vrhova treba pronaći put između njegova 2 odabrana vrha tako da ukupna suma težina bridova koji se nalaze u putu bude minimalna.

Najprije ćemo uvesti osnovne pojmove teorije grafova nužne za razumijevanje navedene teme:

**JEDNOSTAVAN GRAF** – graf u kojemu nema višestrukih bridova između vrhova i nema petlji.

**TEŽINSKI GRAF** – graf u kojemu je svakom bridu pridružen ne-negativan realni broj. Taj broj se naziva težina brida. Težina grafa je zbroj svih težina bridova u grafu.

**PUT IZMEĐU 2 VRHA GRAFA** – konačan niz vrhova i bridova grafa u kojemu su svi vrhovi i svi bridovi međusobno različiti.

**POTPUNI GRAF** – jednostavan graf u kojemu su svaka 2 vrha povezana bridom.

**LIST GRAFA** – vrh grafa koji ima samo jednog susjeda (vrh stupnja 1).

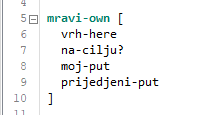
**SUSJEDNI VRH** (nekog vrha grafa) – vrh koji je s tim vrhom grafa povezan nekim bridom. Skup takvih vrhova naziva se susjedstvo danog vrha.

# **2. Realizacija projekta, svijet i agenti**

Za početak je potrebno implementirati pojam grafa u Net-Logu. Vrhovi grafa su predstavljeni vrstom (breed) zbog jednostavnosti nazvanom “vrhovi”. Taj breed ima određena svojstva koja jednoznačno određuju svaki pojedini vrh. Osnovno svojstvo koje je jedinstveno za svaki vrh je “vrijednost” koju predstavlja niz znakova. Sljedeće važno svojstvo je svojstvo “susjedi”, tj. lista vrhova koji su povezani bridom s danim vrhom. Zbog implementacije algoritama pretrage potrebno je dodatno svojstvo za svaki vrh, a to je svojstvo “posjećen?” koje je bool tipa i koje služi kao oznaka pri posjećivanju vrhova tijekom algoritma pretrage.

Bridovi u grafu su reprezentirani postojećom vrstom u Net-Logu zvanom “links”. Bridovima je dodano svojstvo “težina” koje je realan broj i predstavlja težinu brida između neka 2 vrha.

Pomoćna vrsta za pretrage implementirana u Net-Logu su “mravi”. To je vrsta koja reprezentira reaktivne agente koji služe za više-agentske pretrage na grafu.



Slika 1. Implementacija pomoćne vrste „mravi“ u Net-Logu

Mravi znaju trenutni vrh na kojem se nalaze te pamte sve vrhove kojim su prošli u jednom pretraživanju u listu “moj-put”. Svojstvo “prijeđeni-put” je ukupna suma težina bridova kojima je prošao mrav u nekom pretraživanju.

Za većinu pretraživanja svi mravi koriste identičan osnovni korak pretrage, čiji je pseudokod:

pitaj svakog mrava koji nije došao do cilja

za svaki susjedni vrh od trenutnog vrha na kojem se nalazi

ako mrav nije bio na tom vrhu

napravi svoju kopiju

pomakni kopiju na taj susjedni vrh

die (brišemo originalnog mrava iz simulacije)

Dani korak-pretrage se koristi ovisno o logici pretrage. Primjerice, korak pretrage možemo pozivati na zadnjem stvorenom mravu (pretraga po dubini), ili na svakom mravu u jednom koraku (pretraga po širini).

# **3. Algoritmi pretrage**

U ovom dijelu govorit će se o logici algoritama pretrage te njihovoj implementaciji u ovom projektu i na kraju analizi njihove učinkovitosti.

Sve pretrage koje su implementirane u projektu su temeljene na pravom algoritmu, primjenjene na dani problem uz poneku promjenu koja se tiče optimizacije algoritma. Algoritmi koji su analizirani imaju više verzija koje ovise o problemu koji se rješava. U usporedbi će biti obrazloženo zašto je korištena neka konkretna verzija.

Nadalje će biti analizirane klasične pretrage po dubini i širini te prikazani primjeri pohlepnih i informiranih usmjerenih te slijepih pretraživanja. Neki algoritmi su implementirani tako da nisu općenito potpuni, iz razloga da se istaknu problemi koji se mogu javiti kod takvog načina pretraživanja, kad se u obzir uzme neki veliki graf koji je potpun ili skoro potpun, te neki graf s malo bridova.

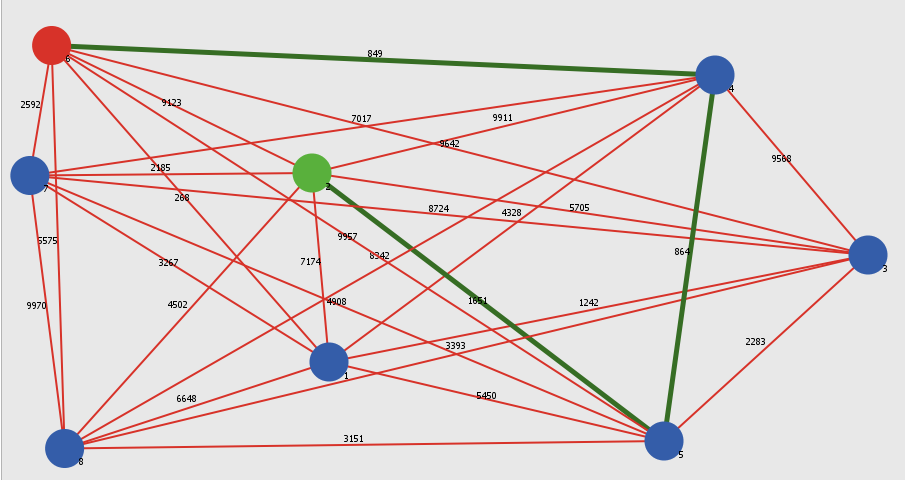
Glavni elementi usporedbe algoritama su **potpunost** (pronalazi li algoritam uvijek rješenje?), **optimalnost** (ako algoritam pronađe rješenje, je li to najbolje rješenje?) te **ukupan broj** **korištenih agenata u pretrazi** (zauzeće memorije).

## **3.1. Pretraga po dubini**

Pretraživanje po dubini je jedno od najpoznatijih primjera slijepog pretraživanja. Slijepo pretraživanje je nazvano tako jer nam prije rješavanja problema nisu poznate nikakve informacije o problemu, osim konačnog ciljnog stanja.

Kod pretrage po dubini krećemo iz početnog vrha i nastavljamo ispitivati njegovu djecu vrhove u svakom sljedećem koraku, sve dok ne dođemo do nekog lista ili cilja. Ako dođemo do lista, algoritam se vraća na posljednji vrh kojemu nisu ispitana sva djeca vrhovi.

Glavna mana ovog pretraživanja je njegova vremenska složenost, i prostorna kompleksnost koja dolazi do izražaja kada stablo pretrage ima veliku dubinu. Algoritam je potpun. Na slici 2 prikazan je najkraći put dobiven pretragom po dubini za potpuni graf s 8 vrhova.



Slika 2. Najkraći put dobiven pretragom po dubini

Pretraga je iscrpna,”mravi” istražuju sve moguće putove s optimizacijskim dodatkom: ako su već neki mravi stigli do kraja puta, onda iz simulacije brišemo sve mrave koji imaju prijeđeni put veći od dotad najboljeg nađenog puta. Rješenje dobiveno ovim algoritmom je optimalno jer imamo konačno vrhova i izbjegavamo cikluse.

## **3.2. Pretraga po širini**

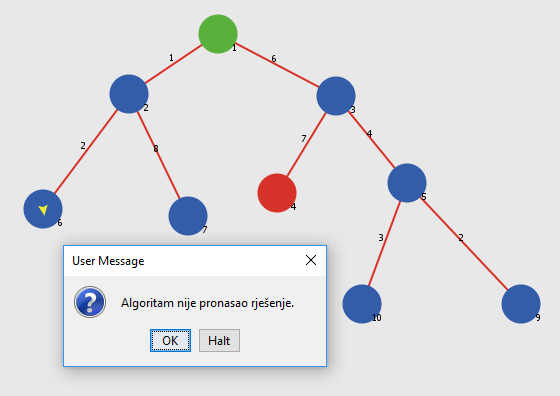
Pretraživanje po širini je još jedno pretraživanje iz klase slijepih pretraživanja. Logika pretrage je jednostavna. Ispituju se svi vrhovi na nekoj razini, nakon čega se prelazi na sljedeću. Algoritam pretrage je potpun. Algoritam u je u Net-Logu implementiran kao i pretraga po dubini, uz pomoć agenata (mravi). U svakom koraku algoritma koristimo osnovni korak pretrage. Svaki postojeći agent širi se na susjedne vrhove na kojima još nije bio, nakon čega uklanjamo originalnog agenta. Pretraga je implementirana kao iscrpna jer istražuje sve moguće putove, uz optimizaciju uklanjanja agenata koji u nekom trenutku imaju dulji ukupni prijeđeni put od nekog mrava koji je već stigao do cilja. To je osnovna razlika između ove verzije pretrage po širini i originalno zamišljene pretrage po širini koja ponekad u nekim slučajevima istražuje svaki vrh grafa. Najveća prednost ovog pretraživanja je njegova potpunost i činjenica da ne moramo unaprijed znati gdje se nalazi rješenje, dok najveći nedostatak ove pretrage dolazi do izražaja kada se rješenje nalazi na velikoj dubini u grafu jer se tada za svaku razinu grafa radi pretraga svakog vrha na toj razini.

## **3.3. Pohlepno pretraživanje**

Pohlepno pretraživanje (greedy search) pripada u heuristička pretraživanja, točnije u klasu najboljeg prvog pretraživanja (best first search). Postoji nekoliko varijacija ovog algoritma. U projektu je implementirana najjednostavnija od njih, poznatija pod nazivom **algoritam najbližeg susjeda**. U ovom algoritmu pretraga je jedno-agentska. Heuristička funkcija je vrijednost težine brida, tako možemo oponašati neke prirodne pojave (zračna udaljenost između 2 grada). U svakom koraku pretrage vrhova, mrav bira od svih susjednih vrhova na kojima nije bio, onog do kojeg je najmanja težina brida. Očito ta „pohlepnost“ ne mora uvijek dovesti do rješenja, točnije glavni nedostatak pretrage je nepotpunost algoritma. Postoje razne verzije algoritma u kojima se optimizira pretraga načinom da se agent, ako je došao do kraja puta koji ne vodi prema cilju (došao je do lista grafa koji nije cilj), vraća natrag na zadnje posjećeno polje, da pokuša pronaći novi put. Te verzije su primjerenije jer ne ovise toliko o odnosu broja vrhova i bridova. Ovdje je prikazana varijanta koja ne koristi tu optimizaciju kako bi bili vidljivi osnovni problemi pohlepnih pretraživanja koji se javljaju. Pohlepno pretraživanje je potpuno samo ako je stanje problema predstavljeno potpunim grafom, jer tada mrav ne može završiti u „slijepoj ulici“ budući da su svaka 2 vrha povezana bridom. Ovakav pristup traženju najkraćeg puta je inspiriran ponašanjem ljudi i životinja u konfliktnim situacijama gdje imaju na raspolaganju malo informacija (u ovom slučaju svaki mrav vidi samo susjedne vrhove i cijenu puta). Zaključak o ovoj pretrazi je da bi se trebala upotrijebiti kada imamo stanje s „puno“ bridova (graf je povezan ili „blizu“ povezanog) jer tada postoji veća vjerojatnost da algoritam pronađe neko rješenje. Algoritam može koristiti i neke druge heuristike ili kombinaciju više heurističkih funkcija.

## **3.4. Tabu pretraga**

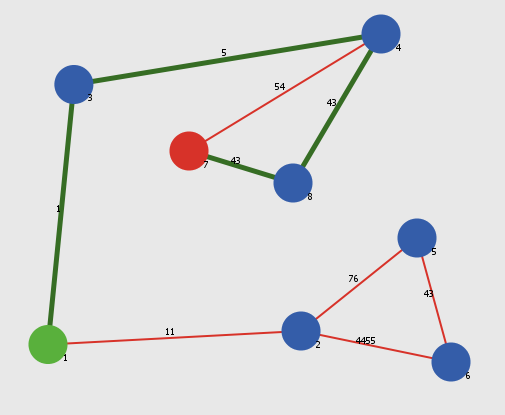
Tabu pretraga, kao i već opisani pohlepni algoritam, pripada klasi best first search pretraživanja. Osnovno svojstvo ove pretrage je korištenje „kratkotrajne memorije“ agenta koji pretražuje prostor stanja. U tu svrhu, koristi se tabu lista u kojoj agent (mrav) sprema posjećene vrhove u nekoliko zadnjih koraka, s ciljem izbjegavanja ciklusa u grafu. Lista je ograničena i kada je popunjena, novi posjećeni vrh se stavlja na kraj, a iz nje se izbacuje prvi element (FIFO princip). Na ovaj način se izbjegava alternacija dvaju rješenja, dakle ciklusi u grafu. Kao i za pohlepni algoritam prikazana je implementacija najjednostavnije verzije algoritma koja nije potpuna, odnosno ne daje u svakoj situaciji rješenje (prikazano na slici 3).



Slika 3. Algoritam „Tabu pretraga“ – najjednostavnija verzija koja nije potpuna

Kako se vidi na slici 3 agent na početku se nalazi na vrhu obojanom u zeleno i cilj je pronaći najkraći put do vrha s brojem 4 (crvena boja). Agent u prvom koraku bira vrh broj 2 jer on ima najbolju vrijednost heurističke funkcije. Zatim bira vrh broj 6 jer u drugom koraku on ima najmanju vrijednost heurističke funkcije. Tabu lista je sada {1,2,6} i mrav nema više na raspolaganju slobodnih vrhova koji se ne nalaze u tabu listi. Dakle došao je do uvjeta zaustavljanja algoritma. Ovisno o broju vrhova n, ovdje je prikazana duljina tabu liste ovisno o tome koliki postotak želimo od broja n : 25 %, 50% ili 75%. Algoritam je potpun za potpune grafove jer tada su svaka 2 vrha povezana bridom. Za duljinu tabu liste koja iznosi 75% od broja vrhova n, algoritam je jako sličan pohlepnom algoritmu. Ovisno o problemu i verziji ovog algoritma slijedi njegova potpunost, koja u ovom slučaju općenito nije ispunjena.

Na slici 4 je prikazan primjer uspješne tabu pretrage koja zbog relativno malog broja vrhova sliči na pohlepni algoritam, uz dužinu tabu liste od 25% broja vrhova.



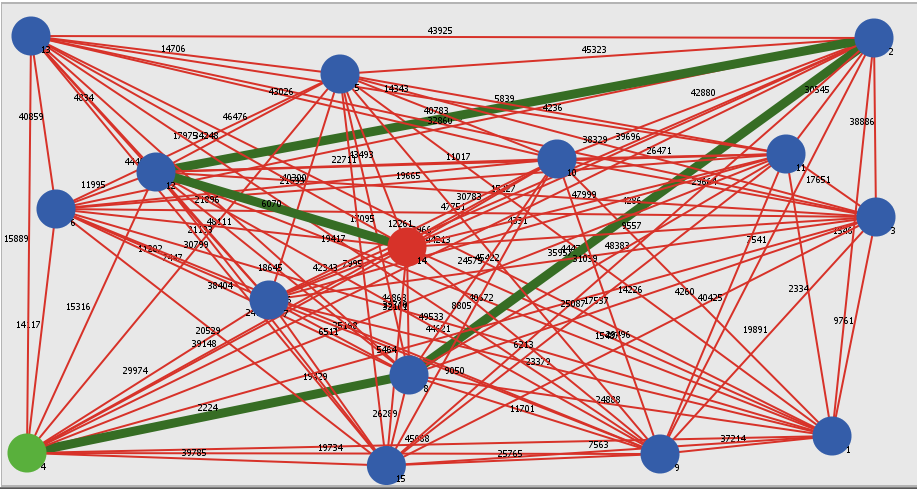
Slika 4. Uspješna tabu pretraga

## **3.5. Algoritam jednolikog troška**

Ovaj algoritam je primjer optimalnih usmjerenih pretraživanja. Cilj ove vrste pretraga je da uvijek daju optimalno rješenje problema. Ideja primjene ovog algoritma na problem najkraćeg puta je višeagentski pristup rješavanju problema, gdje u svakom koraku agent koji je prešao najkraći put (suma cijena puta kojeg je prešao) radi sljedeći korak. Agenti pamte prijeđeni put i sve vrhove koje su prešli. U svakom koraku, agent može posjetiti susjedni vrh ako se on ne nalazi u njegovoj listi posjećenih vrhova, bez obzira što je možda neki drugi agent posjetio taj isti vrh. Ova pretraga je iscrpna, jer svaki put između početnog vrha i cilja će biti istražen. Algoritmu je dodan dio gdje se provjerava je li neki agent stigao do cilja, te ako je, svi oni agenti koji već imaju veći prijeđeni put od tog koji je na cilju, brišu se iz pretrage, jer neće dati najbolje rješenje.

Na prvi pogled ova pretraga je jako slična pretrazi po širini jer ako su težine bridova u nekom malom intervalu, agenti se kreću u širinu budući da su male razlike između njihovih prijeđenih putova. Razlika je u tome što je ovo primjer optimalnog pretraživanja za razliku od klasičnog po širini.

Loša strana ovog algoritma je to što prije same pretrage nemamo informaciju o tome gdje se naš cilj nalazi, te zbog toga agenti su prisiljeni istraživati u svim smjerovima. Dobre strane su potpunost i optimalnost koja je posljedica logike po kojoj se agenti kreću. Stoga, u svakom koraku, ako neki agent stigne do cilja, to ce biti onaj, s trenutnim najkraćim putem. Na slici 5 je prikazan primjer rješenja dobivenog s algoritmom jednolikog troška na potpunom grafu s 15 vrhova. Sve težine tog grafa su iz intervala [1000,50 000].



Slika 5. Primjer rješenja dobivenog s algoritmom jednolikog troška

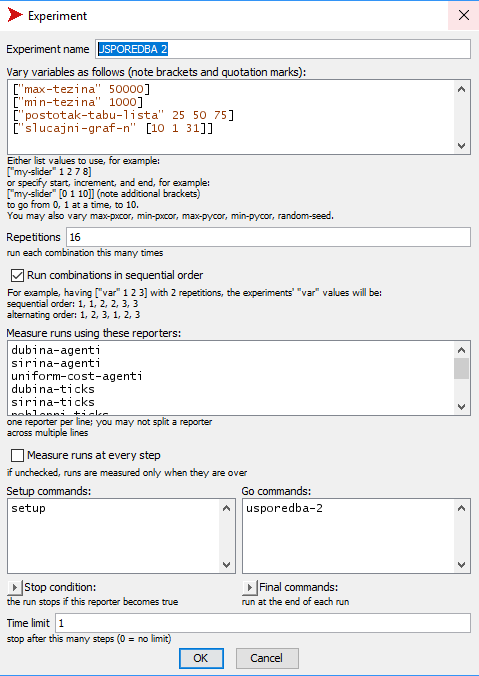
Algoritam jednolikog troška je osnova za A\* algoritam, koji uz informaciju o prijeđenom putu koristi i heuristiku „udaljenosti“ od cilja. Zbroj te dvije funkcije daje informaciju o kvaliteti trenutnog stanja koje agent pretražuje. A\* je dakle poboljšanje ovog algoritma jer koristi dodatne informacije o tome u kojem se „smjeru“ rješenje nalazi u grafu i time izbjegava neke putove.

# **4. Usporedba u Behavior space-u za potpuni graf**

U ovom dijelu ću usporediti navedene algoritme korištenjem Behavior space-a u Net-logu te alata Orange za obradu podataka. Osnovni parametri koji su važni u ovoj usporedbi su potpunost algoritma, optimalnost algoritma te ukupan broj agenata korištenih u jednoj pretrazi za svaki algoritma. Napravio sam 2 eksperimenta od kojih je jedan za slučajni graf, a drugi za potpuni graf, oba s istim parametrima. U jednoj iteraciji simulacije odlučio sam pokrenuti sve algoritme na jednom slučajnom grafu (koji ne mora biti potpun) za svaki od n vrhova za n iz intervala [10,30]. U svakoj iteraciji, za neki n, tabu pretraga će se izvršiti 3 puta, po jednom za svaku moguću duljinu tabu liste: 25%, 50% i 75% od n vrhova. Bridovima grafa će biti pridružena slučajna težina iz intervala [1000,50 000].

Kao što se može vidjeti na slici dolje, gdje su prikazani parametri eksperimenta, odlučio sam mjeriti određene podatke tijekom algoritama na istom grafu (jedna iteracija eksperimenta). Najvažniji podaci su bili broj agenata za svaki od algoritama više-agentske pretrage (dubina, širina i algoritam jednolikog troška) i broj koraka u svakoj pretrazi. Svaki korak predstavlja pomicanje jednog agenta ili njegovo širenje (hatch). To je predstavljeno brojčanim varijablama:dubina-ticks, širina-ticks, uniform-cost-ticks i ostali. Ti podaci su važni zbog vremenske procjene složenosti algoritma, a ukupan broj agenata zbog troška memorije. Pohlepni algoritam i tabu pretraga nisu uvijek potpuni algoritmi u ovom projektu kako je već spomenuto, pa je iz tog razloga važno pratiti u kojim slučajevima jesu, dakle koliki je postotak uspješnosti. Iz tog razloga, 2 dodatna bool parametra tabu-rješenje? i pohlepni-rješenje? služe za praćenje tih svojstava, odnosno omjera uspješnosti ova 2 algoritma.

Svaka od kombinacija ovih parametara je izvedena 16 puta. Ukupno imamo više od 1000 simulacija rješavanja ovog problema za svaki eksperiment.



Slika 6. Prikaz parametara eksperimenta

U PDF prilogu uz dokumentaciju se može vidjeti vizualna usporedba svakog od danih algoritama, koja se tiče broja agenata, broja koraka u kojima se izveo algoritam , te potpunosti rješenja (pohlepni i tabu pretraga).

Za usporedbu na potpunom slučajnom grafu, tabu pretraga i pohlepni algoritam su potpuni algoritmi, jer iz činjenice da između svaka 2 vrha postoji brid, agent uvijek ima na raspolaganju vrh koji nije posjećen, te tako uvijek pronalazi neko rješenje. Heuristika korištena u pohlepnom i tabu algoritmu je heuristika najbliže udaljenosti koja je optimistična jer nikad ne precjenjuje rješenje u svakom koraku. Kako je i očekivano za potpuni graf, najboljim se pokazao algoritam jednolikog troška koji od naša 2 optimalna algoritma (širina, dubina, jednoliki trošak) ima najmanje koraka za broj vrhova veći od 20, najčešće oko 2000 koraka, s rijetkim iznimkama, dok za isti broj vrhova pretraga po širini ima za nijansu veći broj, dok se pretraga po dubini pokazala kao najgora po tom pitanju jer nerijetko imamo više od 20 tisuća koraka.

Što se tiče ukupnog broja agenata korištenih u jednoj pretrazi opet prednjači algoritam jednolikog troška te odmah iza s manjom razlikom pretraga po širini, te kao najgori se ponovo pokazao algoritam pretrage po dubini, gdje za n veći od 25 imamo 20-ak tisuća agenata. Rezultati bi bili gori da nije dodana optimizacija kod zaustavljanja nekih agenata za koje je jasno da neće donijeti najbolji put. Ovo svojstvo nije imalo smisla ispitivati kod pohlepnog algoritma i tabu pretrage jer imamo samo jednog agenta. Kod tih pretraga se broj koraka pokazao istovjetan, što je očekivano s obzirom na broj vrhova n(<= 30), te s obzirom na to da se u nekim slučajevima kao duljina tabu liste koristi 50 % ili čak 75% od broja vrhova n. Kao što sam komentirao već prije, uspješnost pronalaženja rješenja za pohlepni algoritam i tabu pretragu kod potpunih grafova je 100%.

# **5. Usporedba u Behavior space-u za random graf**

Podaci koji su mjereni u Behavior space-u su isti kao i za eksperiment s potpunim grafom. Razlika je jedino to da je broj bridova različit. U PDF prilogu su prikazani identični mjereni podaci za grafove od n vrhova, za svaki n iz intervala [10,30]. Što se tiče zauzeća memorije od 3 potpuna algoritma koji su implementirani, podjednako dobri su se pokazali algoritam jednolikog troška i pretraga po širini, dok je najgori pretraga po dubini koji ponekad zahtjeva po 2000 do 4000 agenata, dakle duplo više nego 2 prethodno navedena. Broj agenata stvorenih u jednoj pretrazi kod malih broja vrhova i bridova (a takve smatramo sve ispod 30-ak vrhova) je često u korespondenciji s brojem koraka algoritma (ticks) jer graf nije „gust“(relativno malo bridova u odnosu na broj vrhova). Kao i za potpune grafove i ovdje se najbržim pokazao algoritam jednolikog troška , neznatno brži od pretrage po širini, ali s boljim informacijama o trenutno dobivenom rješenju.

Što se tiče tabu pretrage i pohlepnog algoritma, oba su u uspješnosti pronalažena rješenja od oko 50%, neovisno o broju vrhova.

# **6. Zaključak**

Cilj ovog projekta bio je prikazati znanje koje sam usvojio na ovom kolegiju primjenjeno na popularan problem najkraćeg puta u grafovima. Svaki od opisanih algoritama ima više verzija koje se implementiraju ovisno o danom problemu. Svaki od njih ima svoje dobre i loše strane koje su opisane i prikazane u usporedbi rezultata pretraga na različitim grafovima. Korištena je tehnika više-agentskog pretraživanja koje je u današnje vrijeme jako popularno i korisno kod ovog ili sličnih problema. Problem je riješen po principima umjetne inteligencije, rastavom na 4 komponente i već spomenutim agentskim pristupom. Moj zaključak je da od implementiranih verzija u ovom projektu se najboljim rješenjem pokazao algoritam jednolikog troška zbog svoje potpunosti i optimalnosti te činjenice da od ovih optimalnih algoritama ima najmanju vremensku kompleksnost. Vidi se kako je njegovo poboljšanje (A\* algoritam), još bolje u smislu da imamo dodatnu korisnu informaciju o „udaljenosti“ od ciljnog stanja, koja nas vodi u tom smjeru.

Pretrage po širini i dubini su klasične pretrage koje su temelj svih drugih i na kojima se temelji logika drugih algoritama. Grafovi na kojima sam analizirao pretrage imali su zbog jednostavnosti najviše 30 vrhova za koje su se pretraga po širini i dubini pokazale još uvijek optimalno rješenje. Za neki veliki broj vrhova ova 2 algoritma su malo sporija od algoritma jednolikog troška i posebno od pohlepnog algoritma što jasno ističe njihovu najveću manu.

Za pohlepni algoritam sam odlučio koristiti jednostavnu verziju najboljeg prvog pretraživanja (najbliži susjed) iz razloga da se jasno pokaže koje su najveće mane ovakvih pretraživanja. U ovom slučaju je to nepotpunost te verzije algoritma.

Od algoritama koje nismo koristili na vježbama, implementirana je tabu pretraga, čija verzija također kao ni za pohlepni algoritam, iz istog razloga nije potpuna.

Jasno je da bi primjena ovih algoritama na neki labirint koji je rješavan na vježbama ipak bila malo drugačija i neki algoritmi bi se malo drugačije ponašali. Zaključujem da je stvaranje nekog inteligentnog sustava s ciljem rješavanja nekog problema jako zahtjevno te da je potrebno zaista razumjeti algoritme pretrage te moći prepoznati kako i koji od njih implementirati na dani problem.

# 7. Literatura

1. Prirodoslovno-matematički fakultet Split ,Saša Mladenović ,Goran Zaharija

Uvod u umjetnu inteligenciju,bilješke s predavanja(2012-2013)

1. A tutorial on tabu search (Heltz,Taillard,Werra). Preuzeto 15. siječnja 2018. s linka <http://www.dei.unipd.it/~fisch/ricop/tabu_search_tutorial>
2. Geeks for geeks , Best first search(informed search) . Preuzeto s linka

<https://www.geeksforgeeks.org/best-first-search-informed-search/> datuma 15. siječnja 2018.