Univerzitet u Beogradu
Matematički fakultet

Longest simple path
(Najduži put u grafu)
Računarska inteligencija

Predrag Stefanović mi20280

1. Uvod

U savremenom računarstvu, problemi povezani sa grafovima igraju ključnu ulogu u raznim aplikacijama, na primer optimizacije mreže. Jedan od značajnijih problema u teoriji grafova je pronalaženje najdužeg puta u usmerenom ili neusmerenom grafu. Ovaj problem je izazovan zbog svoje kompleksnosti, posebno kada se radi o grafovima koji sadrže cikluse, gde pronalaženje najdužeg NP-težak problem.

Rešenje problema tražimo koristeći optimitzaciju mravlje kolonije i algoritma grube sile

Problem se smatra NP-težkim iz nekoliko razloga:

Eksponencijalni broj mogućih putanja: U opštem grafu, posebno onom koji može imati cikluse, postoji eksponencijalno mnogo mogućih putanja koje je potrebno istražiti. Kada graf ima nnn čvorova, broj različitih putanja može rasti brzo kako se dodaju čvorovi.

Težina provere rešenja: Čak i kada se predstavi potencijalno rešenje (npr. određena putanja), provera da li je to najduži put zahteva analizu svih mogućih putanja koje mogu biti duže. Ovo čini problem teškim za rešavanje u razumnom vremenskom okviru.

Sličnost s drugim NP-teškim problemima: Problem najdužeg puta je sličan drugim NP-teškim problemima, kao što je problem Hamiltonovog ciklusa. Mnoge tehnike za rešavanje NP-teških problema, poput povratne pretrage ili heurističkih pristupa, mogu se primeniti na problem najdužeg puta.

Odsustvo efikasnog algoritma: Trenutno ne postoji poznat polinomijalni algoritam koji bi mogao da reši sve instance ovog problema u razumnom vremenu. Ovo ga čini izazovnim za rešavanje, posebno u aplikacijama koje zahtevaju brze rezultate.

Zbog ovih faktora, problem najdužeg puta se klasifikuje kao NP-težak, što znači da ne postoji poznati efikasan način da se reši za sve moguće grafove.

2. Opis problema

Instanca: Graf G = (V,E)
Rešenje: Prost put u grafu G, tj, niz različitih čvorova V_1, V_2, . . ., V_m takav da, za bilo koji $1 \leq i \leq m - 1$, važi $(V_i, V_{i+1}) \in E$.
Merilo: Dužina puta, tj. broj ivica u putu.

Pronalaženje optimalnog rešenja za ovaj problem je računski zahtevno,
U praktičnim primenama se mogu koristiti
heurističke metode ili aproksimacije kako bi se došlo do skoro optimalnog rešenja sobzirom na teškoću dobijanja tačnog rešenja.

3. Rešavanje problema

Za pronalaženje najdužeg puta u grafu, se koristi algoritam mravlje kolonije, kao i jedan algoritam grube sile

3.1 Algoritam mravlje kolonije

Kratak pregled glavnih delova algoritma.

Klasa **ACO:**

- **'__init__'**: Inicijalizuje objekte koji su potrebni algoritmu za rad

  graph: Graf koji se koristi za pretragu, obično predstavljen kao skup čvorova i ivica.

  nodes_info: Informacije o čvorovima u grafu, što može uključivati dodatne podatke vezane za svaki čvor.

  adjacency_info: Informacije o povezanim čvorovima, koje definišu odnose između čvorova u grafu.

  ants: Broj mrava (agenta) koji će se koristiti u procesu optimizacije.

  alpha: Težinski faktor koji utiče na značaj feromona prilikom odabira puta.

beta: Težinski faktor koji utiče na značaj heurističkih informacija prilikom odabira puta.

rho: Faktor isparavanja feromona, koji određuje koliko feromona nestaje tokom svake iteracije.

q: Kontrolna konstantna koja se koristi za dodavanje feromona na ivice nakon što mravi završe svoj put.

iterations: Broj iteracija koje će algoritam izvršiti.

- **Funkcija run:** pokrece program
- **Funkcija generate_paths:** Generise sve dozvoljene puteve
- **Funkcija choose_next_node:** Bira sledeci cvor
- **Funkcija calculate_probability:** Racuna šansu za odabir sledeće grane
- **Funkcija update_pheromone:** Postavlja vrednost feromona na grane grafa
- **Funkcija update_best_path:** Postavlja vrednost najdužeg puta
- **Funkcija plot_graph:** Iscrtava graf na kome se program izvršava

3.2 Algoritam grube sile

- **Generisanje permutacija**:

Algoritam generiše sve moguće permutacije čvorova grafa i istražuje svaku permutaciju kako bi utvrdio da li čvorovi mogu formirati validan put.

- **Istraživanje permutacija**:

Za svaku permutaciju, algoritam stvara curr_path (trenutni put) i dodaje čvorove jedan po jedan.
Prvo dodaje prvi čvor iz permutacije.
Zatim proverava da li je sledeći čvor povezan sa poslednjim dodanim čvorom putem ivice.
Ako jeste, dodaje ga u curr_path. U suprotnom, prekida proces za tu permutaciju.

- **Praćenje Najdužeg Puta:**

Nakon prolaska kroz sve čvorove u permutaciji, algoritam proverava da li je trenutni put duži od do sada pronadjenog najdužeg puta, ako jeste ažurura najduži put

- **Praćenje progrsije**

Koristi se biblioteka **tqdm** koja omogucava vizuelno praćenje napredka programa

3.2.1 Performanse algoritma grube sile

- **Kompleksnost**:

O(n!), gde je n broj čvorova u grafu. Ovaj pristup može postati veoma spor za veće grafove zbog eksponencijalnog rasta broja permutacija.

- **Prostor**:

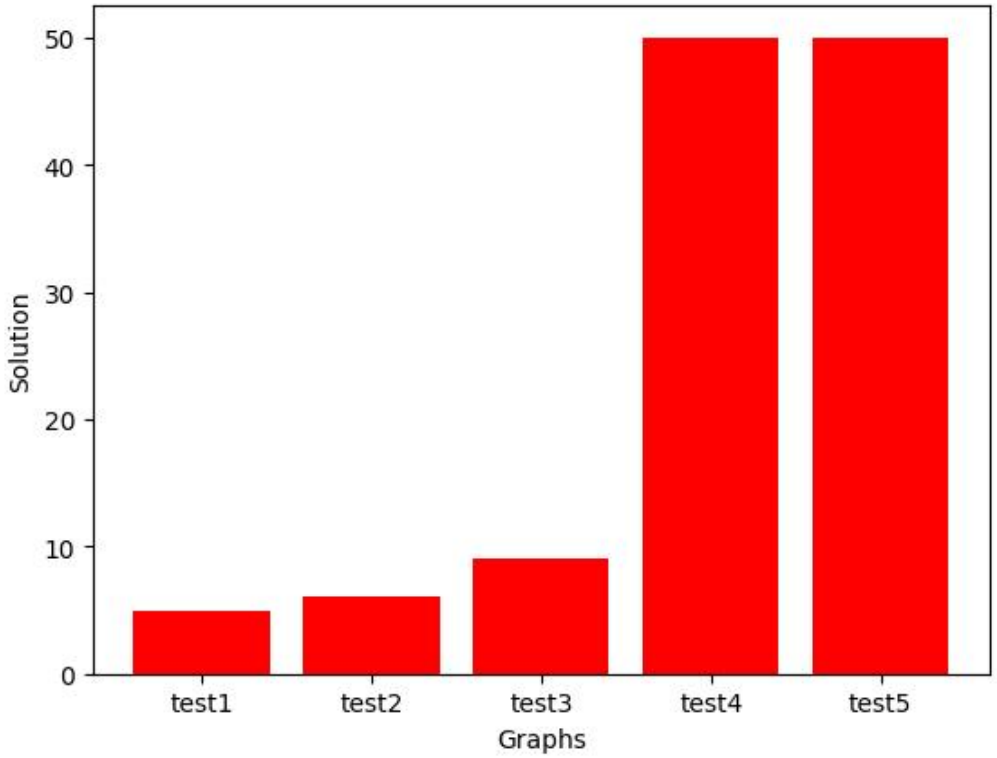
O(n) za skladištenje trenutnog puta i dodatne strukture podataka.

3.2.2 Performanse mravlje kolonije

- Mravlja kolonija koristi hjeuristicke metode i probabilistički pristup za pretragu rešenja, omogućavajući mu da efikasnije istražuje prostranstvo rešenja. U praksi, mravlja kolonija često pokazuje bolju performansu i može se koristiti za veće i složenije grafove.

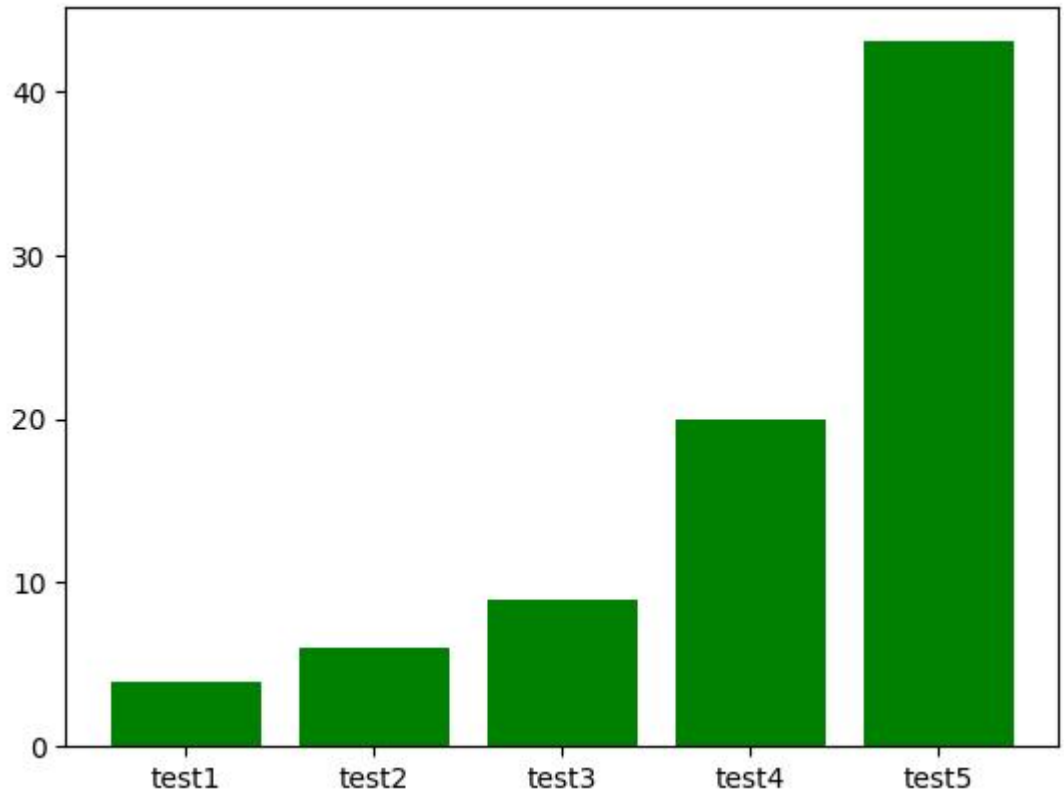
4. Analiza rezultata

4.1 Algoritam grube sile



Na grafikonu su prikazana rešenja do kojih je došao algoritam grube sile.
Za test primere 4 i 5 zbog faktorijelne složenosti algoritam nije uspeo da dodje do rešenja.
Ovaj grafik nam daje uvid u to kako složenost i veličina grafa direktno utiču na mogucnost pronalaženja rešenja.
Grafovi sa većim dimenzijama zahtevaju znatno više permutacija za generisanje i obradu, što je očekivano zbog prirode problema koji se rešava.

4.2 Optimizacija kolonijom mrava



Na grafikonu su prikazana rešenja do kojih je došao algoritam optimizacije mravljom kolonijom.
Posto je mravlja kolonija jedan od optimizacionih algoritama za manje test primere mora da pronadje tačno resenje, ovom slucaju to su sigurno test primeri 1, 2 , 3.
To možemo da zakljucimo uporedjivati vrednosti do kojih je došao algoritam grube sile, iz razloga što isti ukoliko vraća neko rešenje ono mora da bude tačno.
Za test primere 4 i 5 optimizacija kolonijom mrava je došla do nekih rešenja, pošto su problemi previše veliki za grubu silu ne možemo garantovati potpunu tačnost, već samo parcijalnu tačnost, tj, dovoljno dobro rešenje problema.

5. Zaključak

U ovoj analizi, smo upoređivali algoritam grube sile za pronalaženje najdužeg jednostavnog puta sa optimizacionim algoritmom kolonije mrava(**ACO**). Ova dva pristupa predstavljaju različite paradigme rešavanja problema optimizacije u grafovima, svaki sa svojim prednostima i nedostacima.

5.1. Efikasnost

- **Gruba sila**:

Ima eksponencijalnu složenost (O(n!)), što ga čini neizvodljivim za veće grafove zbog brzog rasta broja permutacija. Ovaj pristup je efikasan samo za manje grafove sa ograničenim brojem čvorova.

- **ACO:**

ACO koristi heurističke metode i probabilistički pristup za pretragu rešenja, omogućavajući mu da efikasnije istražuje prostranstvo rešenja. U praksi, ACO često pokazuje bolju performansu i može se koristiti za veće i složenije grafove.

5.2. Kvalitet Rešenja

- **Gruba sila**:

Garantuje pronalaženje optimalnog rešenja jer istražuje sve moguće puteve. Međutim, ovo dolazi po cenu vremena i resursa.

- **ACO:**

Iako ACO ne garantuje optimalno rešenje, često pronalazi veoma dobra rešenja u razumnom vremenskom okviru. Uz dodatne strategije kao što su adaptivne feromone i lokalna pretraživanja, ACO može poboljšati kvalitet rešenja.

5.3. Prilagodljivost i Primena

- **Gruba sila**:

Jednostavno implementirati, ali nije prilagodljivo za složenije probleme ili dinamičke grafove. Ograničeno na statičke grafove.

- **ACO:**

Visoko prilagodljiv i može se primeniti na različite probleme optimizacije, uključujući one sa promenljivim uslovima i dinamičkim promenama u grafu.

5.4. Pristup Problemu

Gruba sila:
Sistematski pristup koji osigurava da su svi putevi ispitani, ali je nedelotvoran u pogledu vremena.

ACO:
Inspirisan prirodom, koristi koncept kolonija mrava za pronalaženje puta do resursa, efikasno učeći iz prethodnih iteracija.

Dok je algoritam grube sile jednostavan i garantuje optimalna rešenja za manje grafove, njegov nedostatak efikasnosti ograničava njegovu primenu u realnim scenarijima sa većim grafovima. S druge strane, ACO nudi fleksibilniji i efikasniji pristup koji može brzo pronaći dobra rešenja, čak i za kompleksnije probleme. Stoga, izbor između ova dva pristupa zavisi od specifičnih zahteva problema, uključujući veličinu grafa, potrebu za optimalnošću i dostupne resurse za izvršenje.