Istraživanje Tabu pretrage u kontekstu problema trgovačkog putnika

Sandra Petrović 84/2019

Iva Đorđević 267/2019

Januar 2024

1. Uvod

U realnom svetu se javljaju razni problemi optimizacije. Efikasno rešavanje ovih problema igra ključnu ulogu u poboljšanju performansi mnogih sistema. U cilju rada na ovim izazovima, razvijene su različite tehnike optimizacije, a tehnika koja je prikazana u ovom radu predstavlja Tabu pretragu u kontekstu problema trgovačkog putnika.

1.1 O radu

U ovom radu smo rešavali problem trgovačkog putnika (TSP) korišcenjem Tabu pretrage. Prikazali smo rešenja za 2 varijante ovog NP-teškog problema: kada je on simetričan i kada je asimetričan. Dobijeni rezultati su vizuelno prikazani. Projekat je implementiran u programskom jeziku Python, a izvršava se uz pomoć Jupyter Notebook-a.

1.2 Skup podataka

Za simetričan TSP:
Za potrebe testiranja rada algoritma na konkretnom problemu, kao ulazne podatke koristili smo koordinate gradova date u fajlu. Rastojanje između dva grada smo računali korišćenjem formule za Euklidsko rastojanje.
Za asimetričan TSP:
Podatke o rastojanjima između gradova čuvali smo u matrici.

2. TSP

Problem trgovačkog putnika, poznat i kao TSP, predstavlja NP težak problem u kombinatornoj optimizaciji. TSP je prvi put formulisan 1930. godine i jedan je od najintezivnije proučavanih problema u optimizaciji.
U osnovi, postavka problema je sledeća:
Trgovac želi da pronađe, počevši od svog rodnog grada, najkraći put kroz dati skup gradova kupaca i da se na kraju vrati u svoj grad. Pri tom je važno napomenuti da svaki grad treba da poseti samo jednom. Dakle, cilj je minimizovati ukupnu duzinu puta.
Formalno, ukoliko imamo n gradova, indeksiranih od 1 do n, rastojanja između svakog para gradova i tražimo permutaciju gradova P=(p1,p2,...,pn), gde p_i predstavlja redosled gradova, takvu da minimizuje izraz:

$$\mathrm{L(P)} = \sum i = 1\mathrm{n} - 1 d_{p_i,p_{i+1}} \; d_p + d_{p_n,p_1}$$

gde $d_{p_i,p_{i+1}}$ predstavlja rastojanje između gradova p_i i p_{i+1}, a d_{p_n,p_1} rastojanje izmedju poslednjeg i prvog grada.

Značaj rešavanja ovog problema ogleda se u tome što ima brojne primene u različitim oblastima kao sto su logistika, proizvodnja i distribucija, telekomunikacije, rutiranje, mikroelektornika, genetika…

Rešavanje ovog problema može biti izazovno jer se broj mogućih permutacija eksponencijalno povećava sa brojem gradova. Postoje različite metode za rešavanje TSP-a, uključujući tačne algoritme (poput grube sile ili dinamičkog programiranja) i heurističke algoritme (poput algoritma genetskog algoritma, simuliranog kaljenja, ili lokalne pretrage). Izbor metode zavisi od veličine problema i specifičnih zahteva. U našem radu mi smo se fokusirali na rešavanje ovog problema korišcenjem Tabu pretrage koja se pokazala kao efikasna metoda za rešavanje ovog problema, posebno za instance sa velikim brojem gradova.

U simetričnom TSP-u, rastojanje između dva grada je isto. Ova simetrija prepolovi broj mogućih rešenja. U asimetričnom TSP-u, putanje možda neće postojati u oba smera ili razdaljine mogu biti različite, formirajući usmereni graf. Saobraćajni sudari, jednosmerne ulice i cene avionskih karata za gradove sa različitim taksama odlaska i dolaska su primeri kako bi se ova simetrija mogla narušiti.

3. Tabu pretraga

Tabu pretraga predstavlja algoritam optimizacije koji se često koristi za rešavanje problema kombinatorne optimizacije, uključujući i klasičan problem trgovačkog putnika (TSP). Ovaj algoritam pruža efikasno rešenje za pronalaženje približnih optimalnih rešenja u prostorima pretrage velikih dimenzija.

Osnovna ideja Tabu pretrage je da pretražuje prostor rešenja u okolini trenutnog rešenja, izbegavajući ponavljanje prethodnih koraka kako bi se izbegli lokalni optimumi i istražio širi prostor rešenja. Možemo izdvojiti neke bitne elemente Tabu pretrage: generisanje susednih rešenja, tabu lista, evaluacija rešenja i odabir sledećeg rešenja, koji su opisani u nastavku.

Generisanje susednih rešenja: Tokom svake iteracije, algoritam generiše susedna rešenja u blizini trenutnog rešenja. Ovi susedi se stvaraju primenom različitih operacija, poput zamene gradova, invertovanja dela putanje ili dodavanje/uklanjanje gradova.

Tabu lista: Tabu lista je ključni element Tabu pretrage koji sprečava ponavljanje loših koraka. Ova lista čuva nedozvoljena rešenja, koja su nedavno posmatrana. Ideja je da se izbegnu ciklusi i diverzifikacija pretrage.

Evaluacija rešenja: Svako generisano susedno rešenje se evaluira kroz funkciju cilja koja meri kvalitet rešenja u kontekstu TSP-a. Tipično, ova funkcija meri ukupnu dužinu putanje trgovačkog putnika.

Odabir sledećeg rešenja: Na osnovu evaluacije susednih rešenja, algoritam bira sledeće rešenje za istraživanje. Uzimajući u obzir tabu listu, algoritam bira rešenja koja nisu nedavno razmatrana, čime se izbegavaju loši ciklusi i lokalni optimumi.

Izbor metode za računanje rastojanja je veoma bitan korak prilikom rešavanja problema trgovačkog putnika (TSP) jer utiče na tačnost i efikasnost algoritma. Mi smo ovde opredelili za Euklidsko rastojanje. U većini slučajeva, Euklidsko rastojanje se koristi kao metoda za izračunavanje udaljenosti između gradova u simetričnom TSP-u zbog svoje jednostavnosti i praktičnosti.

Ako je non_tabu_neighbors prazna lista, to znači da su svi susedi na tabu listi, tj. sva moguća susedna rešenja već postoje na tabu listi. U tom slučaju, nema potrebe dalje istraživati susede, jer su svi već razmatrani i nisu doneli poboljšanje u prethodnim iteracijama. U praksi, kada se svi susedi nalaze na tabu listi, to može značiti da algoritam trenutno zapinje u lokalnom optimumu ili da nije u stanju da generiše nove korisne susede. Ponovno pokretanje pretrage omogućava algoritmu da se izbavi iz ove situacije i nastavi sa istraživanjem novih suseda u nadi pronalaženja boljeg rešenja.

Rad algoritma se završava kada dostigne maksimalan broj iteracija (max_iterations). To znači da će se pretraga završiti nakon što algoritam izvrši određeni broj iteracija, bez obzira na to da li je postigao optimalno rešenje ili ne. Ovaj pristup omogućava da se algoritam zaustavi čak i ako nije moguće postići poboljšanje u rešenju ili ako pretraga ne konvergira.

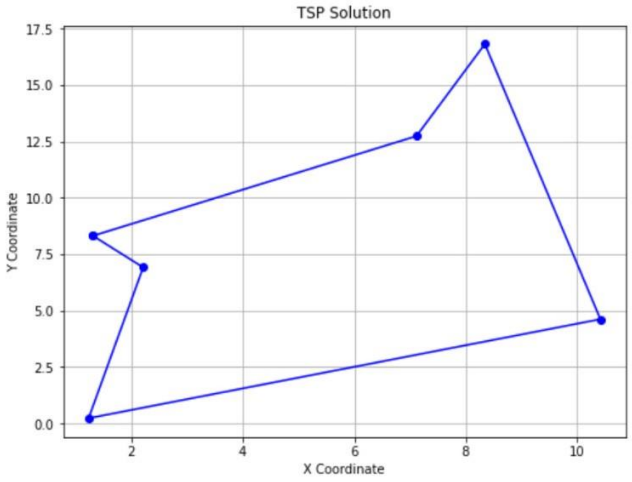
4. Postignuti rezultati

Za testiranje rezultata koristili smo sledeće parametre:

- broj iteracija: 100
- veličina Tabu liste: 10

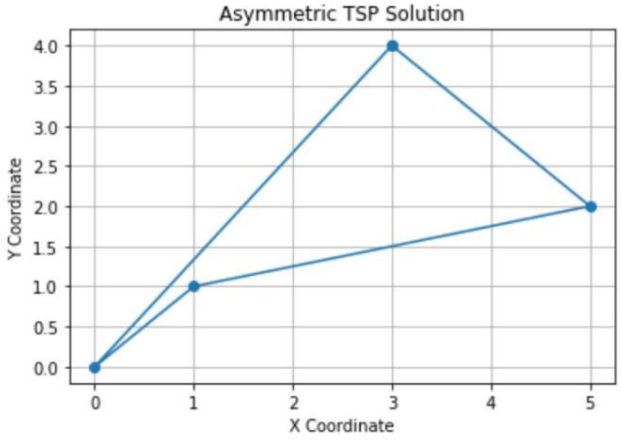
Postavljanjem drugačijih parametara postigli bismo drugačije rezultate, o čemu će biti reči kasnije.

4.2 Simetričan TSP



Nakon izvršavanja Tabu pretrage za rešavanje simetričnog TSP-a nad skupom gradova datih koordinatama, dobili smo optimalnu putanju koju je trgovac trebao da pređe, kao i ukupnu dužinu puta. Optimalna putanja kojom trgovac treba da prođe je definisana redosledom poseta gradovima. Kao što prikazuje sam grafik, jedna putanja može biti: 3 2 0 1 5 4. Ovaj redosled ukazuje na redosled poseta gradovima, pri čemu trgovac počinje od grada 3, zatim posećuje grad 2, pa grad 0 i tako redom, sve dok ne završi u gradu 4 i ponovo se vrati u početni grad. Ukupna dužina puta koju trgovac treba da pređe iznosi 42.522. Ova vrednost predstavlja zbir svih rastojanja između gradova duž putanje koju je trgovac prošao.

4.3 Asimetričan TSP



Nalik simetričnom TSP-u, i kod asimetričnog TSP-a smo takođe dobili optimalnu putanju koju je trgovac trebao da pređe, kao i ukupnu dužinu puta. Sa grafika uočavamo da jedna putanja može biti: 1 0 2 3, pri čemu je ukupna dužina puta koju trgovac treba da pređe 12.

5. Upoređivanje rezultata za različiti broj iteracija i dužine tabu liste

Primeri su testirani na virtuelnoj masini sa jednim jezgrom, brzine 2.20 GHz.

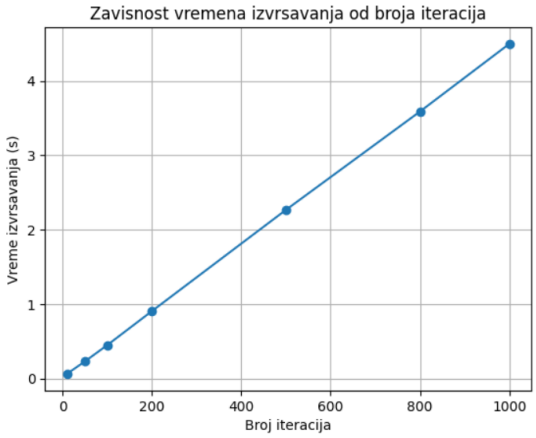
Tekst u nastavku se odnosi na simetričan TSP.

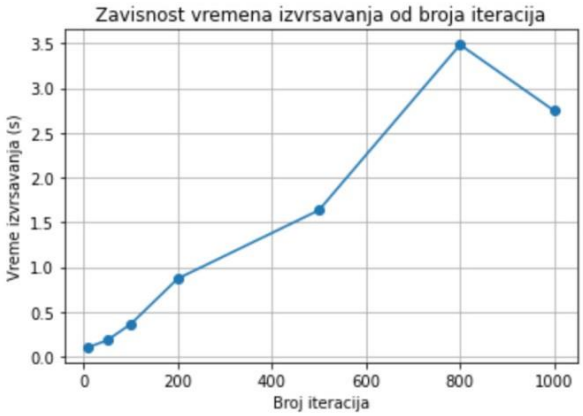
Specifikacije testiranja:

- ulazni podaci: skup od 50 gradova sa njihovim koordinatama
- broj iteracija: postavljeno je na 1000 iteracija kako bi se osiguralo dovoljno vremena za pretragu prostora rešenja
- dužina tabu liste: postavljena je na 50 kako bi se osiguralo raznovrsno istraživanje okoline

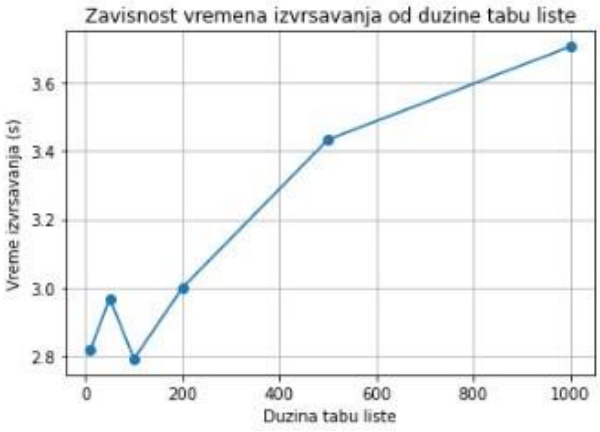
Primećujemo da je vreme izvršavanja algoritma u ovom slučaju 3.8603694438934326 sekundi.

Ukoliko sada izmenimo parametre tako što za broj iteracija postavimo 100, dobijamo da je vreme izvršavanja algoritma 0.5778748989105225 sekundi. Slično, za broj iteracija jednak 10000 vreme je čak 30.900516748428345 sekundi. U ovom slučaju možemo zaključiti da smanjene broja iteracija dovodi do toga da je vreme izvršavanja algoritma značajno smanjeno, što sugeriše da smanjenje broja iteracija može dovesti do efikasnijeg izvršavanja algoritma. Ovo važi za početne iteracije, zatim će se postepeno smanjivati kako se broj iteracija povećava, možda čak i postati konstantan ili blago opadati.





Razmotrićemo i kako se menja vreme izvršavanja algoritma u odnosu na promenu veličine tabu liste, ako je broj iteracija fiksiran i iznosi 1000. Za veličinu tabu liste 50 dobijamo da je vreme izvršavanja algoritma 3.8672590255737305 sekundi. Ako sada postavimo da je veličina tabu liste jednaka 100, vreme izvršavanja algoritma iznosi 3.9012460708618164 sekundi. Povećanjem dužine tabu liste na 1000 dobijamo 4.223812818527222 sekundi. Međutim, sa grafika možemo uočiti da povećanje dužine tabu liste ne uzrokuje uvek duže vreme izvršavanja algoritma.



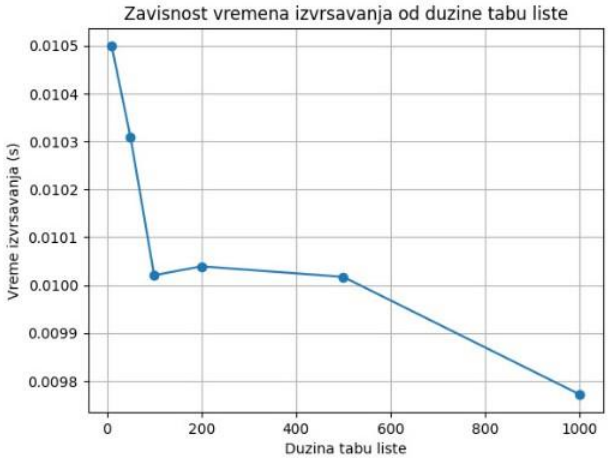
Sada ćemo uporediti rezultate za asimetrični TSP.

Specifikacije testiranja su u ovom slučaju iste kao i kod simetričnog TSP-a.

- ulazni podaci: skup od 50 gradova sa njihovim koordinatama
- broj iteracija: postavljeno je na 1000 iteracija
- dužina tabu liste: postavljena je na 50

Vreme izvršavanja algoritma jednako je 0.19046425819396973 sekundi u slučaju kada je broj iteracija 1000. Kao i malopre, ako broj iteracija postavimo na 100, dobijamo da je vreme izvršavanja algoritma 0.02624654769897461 sekundi. Slično, za broj iteracija jednak 10000 vreme je 1.3807868957519531 sekundi. Možemo zaključiti da smanjene broja iteracija dovodi do toga da je vreme izvršavanja algoritma značajno smanjeno.

Sada posmatramo vreme izvršavanja algoritma u odnosu na promenu veličine tabu liste, pri čemu ćemo za broj iteracija uzeti 1000 i on je fiksiran. Ukoliko za veličinu tabu liste uzmemo 50, vreme izvršavanja algoritma je jednako 0.19046425819396973 sekundi. U slučaju veličine tabu liste jednake 100, vreme iznosi 0.20088696479797363 sekundi. Povećanjem dužine tabu liste na 1000 dobijamo 0.4247422218322754 sekundi. Međutim, sa grafika možemo uočiti da ne postoji zavisnost između dužine tabu liste i vremena izvršavanja algoritma.



6. Literatura

Materijali sa kursa računarska inteligencija

“Metaheuristics: From Design to Implementation” - El-Ghazali Talbi