Tabu pretraga

Seminarski rad u okviru kursa
Računarska Inteligencija
Matematički fakultet

Tatjana Kunić
mi17139@alas.matf.bg.ac.rs

Jun 2021.

Sardžaj

1 Uvod

1.1 Ukratko o Tabu pretrazi

Tabu pretraga (eng. *tabu search*) je metaheuristika koju je definisao Fred Glover 1986. godine. Motivacija iza ovog optimizacionog algoritma jeste preusmeravanje postupka lokalne pretrage iz lokalnih optimuma sa ciljem pronalaženja globalnog optimuma. Tabu pretraga koristi različite memorijske strukture u postizanju ovog cilja.

Reč *tabu* se može definisati kao „zabrana da se izvode neke stalne radnje". U kontekstu tabu pretrage, stavljaju se zabrane na već posećena rešenja čime sprečavamo zaglavljivanje u lokalnim optimumima.

1.2 Problem trgovačkog putnika

Problem trgovačkog putnika (eng. *travelling salesman problem*, u nastavku TSP) se može definisati na sledeći način: Neka je zadat skup C od n gradova i rastojanje $d_{ij} \in \mathbb{N}$ za svaki par gradova, $i, j = 1, \ldots, n$. Pronaći putanju S^* kojom bi trgovački putnik trebalo da se kreće takvu da poseti svaki grad tačno jedanput i da se vrati u onaj iz kog je krenuo, a da tom prilikom pređe najkraći put [9]. U ovom radu ćemo podrazumevati da postoji put između svaka dva grada i da su rastojanja između gradova zadata matricom rastojanja D oblika:

$$D = \begin{pmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{21} & 0 & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{pmatrix}$$

Problem bismo mogli da rešimo iscrpnom pretragom gde bismo prolazili kroz sve moguće permutacije dužine n i računali rastojanja između gradova. Međutim, s obzirom da je vremenska složenost problema $O(n!)$, nalaženje rešenja ovim postupkom je neupotrebljivo za probleme većih dimenzija.

U nastavku rada ćemo diskutovati moguću implementaciju rešenja za TSP algoritmom tabu pretrage.

2 Rešavanje problema trgovačkog putnika algoritmom tabu pretrage

2.1 Kodiranje rešenja i funkcija cilja

Rešenje S ćemo predstaviti kao niz od n brojeva, gde svaki broj predstavlja jedan grad iz skupa C. Ovaj niz označava putanju kojom bi trgovački putnik

trebalo da se kreće. Podrazumevaćemo da je prvi broj u nizu onaj iz kog trgovački putnik kreće i onaj u koji se vraća.

Funkciju cilja f računamo kao zbir rastojanja između svaka dva grada trenutnog rešenja. Rešenje problema S^* će biti ono za koje je funkcija cilja minimalna.

2.2 Inicijalno rešenje

Najjednostavnija inicijalizacija rešenja bi bila generisanje nasumične permutacije skupa gradova. Međutim, obično je dobra praksa da inicijalno rešenje bude već dosta kvalitetno. Jedan od najčešće korišćenih algoritama za inicijalizaciju rešenja je **algoritam najbližeg suseda** (*eng. nearest neighbour*). Algoritam najbližeg suseda jeste pohlepna procedura koja traži najbliži neposećen grad trenutnom. Grad iz kog krećemo ovu pohlepnu pretragu može biti izabran nasumično, ali jednostavnosti radi, počećemo pretragu iz prvog grada (grad označen brojem 1).

2.3 Okolina i potezi

U svakoj iteraciji algoritma modifikujemo trenutno rešenje u nadi da ćemo pronaći bolje rešenje. Ovakvu modifikaciju nazivamo **potez** ili **pomeraj** (*eng. move*). Za naš problem, jedan potez predstavlja zamena neka dva grada trenutne putanje. Na primer, neka je trenutno rešenje za problem od 5 gradova $S = [1, 2, 3, 4, 5]$. Jedan potez bi, na primer, bio kad bismo zamenili gradove 2 i 4 pri čemu dobijamo novo rešenje $S' = [1, 4, 3, 2, 5]$.

Sva rešenja koja mogu da se dobiju primenom jedne transformacije na trenutno rešenje S pripadaju njegovoj **okolini** (*eng. neighbourhood*) koju označavamo kao $N(S)$. Na osnovu prethodno definisanih poteza, možemo i da izračunamo veličinu okoline. Uzmajuci u obzir da je zamena gradova (i, j) ekvivalentna zameni gradova (j, i), dobijamo da je veličina okoline nekog rešenja jednaka

$$\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

U svakoj iteraciji algoritma pravimo potez ka najboljem rešenju iz okoline trenutnog i zamenjujemo ga ukoliko naiđemo na rešenje u kome funkcija cilja ima bolju vrednost od trenutno najbolje pronađene.

2.4 Tabu lista

Kao što je već ranije pomenuto, lokalna pretraga ima tendenciju da se zaglavi u lokalnim optimumima. U našem slučaju ovo se može desiti ukoliko se tokom svake modifikacije rešenja razmenjuju ista dva grada. Iz ovih razloga, tabu pretraga uvodi koncept tabu poteza kojim se sprečava ciklično korišćenje istih poteza, a samim tim i zaglavljivanje u lokalnim optimumima. Ovo se postiže tako što se ono rešenje ka kome je napravljen potez proglasi tabuom. Potrebno

je takođe da imamo negde zabeleženo koji potezi su tabu tokom izvršavanja algoritma. Tabu potezi se čuvaju u memorijskoj strukturi koju često nazivamo **kratkoročna memorija** (*eng. short term memory*) ili **tabu lista**. Za potez koji se trenutno nalazi u tabu listi kažemo da je **tabu aktivan** (*eng. tabu active*), a vreme, tačnije broj iteracija, za koje ostaje tabu aktivan zovemo **tabu period** (*eng. tabu tenure*). Vrednost tabu perioda se uz odgovarajući potez takođe čuva u tabu listi kako bismo u konstantnom vremenu mogli da odredimo da li je neki potez tabu aktivan i vrednost može biti **statička** (*eng. static tabu tenure*) ili **dinamička** (*eng. dynamic tabu tenure*). U mnogim radovima se statička vrednost uzima iz intervala $[5, 25]$[7] i ona ostaje ista tokom celog izvršavanja algoritma. Dinamičke vrednosti se mogu birati ili nasumično ili redom iz nekog unapred odabranog niza vrednosti u toku izvršavanja algoritma. Ne postoji neka univerzalna dobra vrednost za tabu period. Premala vrednost tabu perioda može imati za ishod da je pretraga slična lokalnoj pretrazi, dok velike vrednosti mogu značajno ograničiti okolinu rešenja koju trenutno ispitujemo. Iz ovih razloga se u literaturi često preporučuje da se vrednost tabu perioda menja u toku izvršavanja, međutim jednostavnosti radi, u ovom radu je korišćena fiksirana vrednost.

Tabu pretraga u svakoj iteraciji u stvari traži najbolje rešenje koje nije tabu iz okoline trenutnog rešenja $\tilde{N}(S) = N(S) \setminus \mathcal{T}$, gde je $\mathcal{T}$ skup rešenja koja mogu da se dobiju primenom poteza iz tabu liste.

2.5 Kriterijum aspiracije

Ponekad nema smisla zabranjivati određene poteze. Ukoliko je jedan potez postao tabu aktivan u nekoj iteraciji i i ukoliko je i dalje tabu aktivan u iteraciji $i + k$, a istovremeno popravlja vrednost funkcije cilja, onda praktično gubimo jedno dobro rešenje. Zbog toga je neophodno da postoji mogućnost da zanemarimo tabu status nekog poteza. **Kriterijum aspiracije** (*eng. aspiration criteria*) je strategija koja dopušta primenu poteza kojim dobijamo bolje rešenje od trenutno najboljeg.

2.6 Intenzifikacija

Iako tabu pretraga koju smo definisali do sad daje dosta dobre rezultate, želimo da dodatno razvijemo naš algoritam kako bismo dobijali još bolja rešenja. Jedna od dodatnih i često korišćenih strategija je **intenzifikacija** (*eng. intensification*) koja za cilj ima da dodatno poboljša kvalitet već pronađenih dobrih rešenja. Ovo se postiže tako što se određeni broj najboljih rešenja tokom pretrage čuva u **srednjoročnoj memoriji** (*eng. intermediate memory*) i na kraju pretrage pokušavamo da dodatno modifikujemo ta elitna rešenja kako bismo eventualno pronašli bolje.

2.7 Diverzifikacija

Kako bismo bili sigurni da smo pretragu što detaljnije izvršili, možemo takođe da usmeravamo pretragu ka retko posećenim regionima prostora pretrage kao i da proveravamo deo van granica okoline nekog rešenja. **Diverzifikacija** (*eng. diversification*) uzima u obzir rešenja do kojih možda ne bismo ni stigli lokalnom pretragom, a koja su potencijalno optimalna. Postoji više načina implementacije ove strategije od kojih je najčešće korišćena ona koja beleži frekvencije izvršavanja poteza. Frekvencije se čuvaju u takozvanoj **dugoročnoj memoriji** (*eng. long term memory*). Evaluaciju nekog rešenja izvršavamo na sledeći način:

$$f' = f + d \cdot frequency_{ij}$$

gde $frequency_{ij}$ označava koliko puta je do tog trenutka primenjen potez (i, j), a d je koeficijent diverzifikacije (*diversification parameter*). Veće vrednosti koeficijenta d označavaju veću potrebu za diverzifikacijom.

Kao što je već rečeno, možemo da uzimamo u obzir rešenja koja nisu u okolini trenutnog. U slučaju TSP, tokom pravljenja poteza možemo da izvršimo jednu ili više dodatnih permutacija gde se razmenjuje dodatni par (ili više) gradova. Posmatrajući primer od 5 gradova iz 2.3, pored zamene gradova 2 i 4 razmenjujemo i gradove 3 i 5 čime se dobija $S'' = [1, 4, 5, 2, 3]$ koje nije u okolini rešenja S.

2.8 Pseudokod

Sada kada smo razmotrili osnovne koncepte tabu pretrage, možemo da napišemo pseudokod:

Algoritam 1: Tabu pretraga

```
Inicijalizacija rešenja;
Postavi vrednost najboljeg rešenja na vrednost inicijalnog;
dok nije ispunjen kriterijum zaustavljanja radi
    Modifikuj trenutno rešenje S uzimanjem najboljeg rešenja S' iz
     njegove okoline uz primenu kriterijuma aspiracije i diverzifikacije;
    Zabeleži napravljeni potez u tabu listu;
    ako je novo rešenje bolje od trenutno najboljeg onda
        Postavi najbolje rešenje na novo;
        Zabeleži rešenje u srednjoročnu memoriju;
    kraj
kraj
Intenzifikacija;
vrati najbolje rešenje S* i njegovu vrednost f*
```

3 Rezultati

Algoritmi lokalne i tabu pretrage se zaustavljaju kada se izvrše zadati broj iteracija. Broj iteracija nakon kog se prekidaju je postavljen na 100. S obzirom da se na nekim mestima koriste nasumične razmene, algoritmi se dodatno izvršavaju u celosti 100 puta.

Skupovi podataka koji su korišćeni u testiranju su uglavnom iz biblioteke TSPLIB[10], ali i sa sajta Florida State University[11] i Stack Overflow[12].

Okruženje za testiranje

Operativni sistem	Windows 10 64-bit
Razvojno okruženje	JupyterLab
Verzija interpretera	Python 3.8.8
Procesor	AMD Ryzen 7 2700
RAM	16GB
Broj jezgara	8

3.1 Algoritam grube sile

Algoritam grube sile će uvek dati tačno rešenje, ali se izvršavanje ne isplati za probleme većih dimenzija usled velike složenosti. U tabeli se može uočiti koliko brzo raste vreme izvršavanja.

Naziv skupa	n	**Vreme izvršavanja (s)**	**Najbolje poznato rešenje**	**Najbolje pronađeno rešenje**	**Razlika (%)**
five_d	5	0.001011	19	19	0
eleven	11	111.081097	253	253	0

3.2 Lokalna pretraga

Naziv skupa	n	**Prosečno vreme izvršavanja (s)**	**Najbolje poznato rešenje**	**Najbolje pronađeno rešenje**	**Razlika (%)**
five_d	5	0.002241	19	19	0
p01_d	15	0.044450	291	291	0
gr17	17	0.063184	2085	2085	0
ulysses22	22	0.376645	7013	7639	8.926280
att48	48	3.418413	10628	12221	14.988709
berlin52	52	4.202848	7542	8614	14.213736
bier127	127	60.491724	118282	124804	5.513941
gr202	202	247.913220	40160	43700	8.814741

3.3 Tabu pretraga

Naziv skupa	n	**Prosečno vreme izvršavanja (s)**	**Najbolje poznato rešenje**	**Najbolje pronađeno rešenje**	**Razlika (%)**
five_d	5	0.007322	19	19	0
p01_d	15	0.098533	291	291	0
gr17	17	0.132200	2085	2085	0
ulysses22	22	0.623500	7013	7013	0
att48	48	5.080123	10628	11608	9.220926
berlin52	52	6.402325	7542	8227	9.082471
bier127	127	86.536863	118282	124804	5.513941
gr202	202	323.226017	40160	45342	12.903386

4 Zaključak

Iako smo značajno smanjili složenost sa $O(n!)$, primećujemo da ovo rešenje i nije baš najefikasnije. Uzimajući u obzir da moramo da ispitujemo celu okolinu veličine izračunate u 2.3 kao i činjecu da za svako rešenje iz okoline moramo da računamo vrednost funkcije cilja, dobijamo složenost čak $O(n^3)$ što je i dalje veoma neefikasno za probleme velikih dimenzija. U daljem radu na razvijanju algoritma bi trebalo razmotriti bolje načine pretrage okoline ili čak i redefinisanje okoline rešenja. U radu [4] se na osnovu teorije grupa pronalazi efikasniji način izračunavanja funkcije cilja nakon napravljenog poteza. Još potencijalnih poboljšanja možemo da napravimo ako se problem posmatra grafovski. Rad [8] navodi i opisuje različite heuristike i načine pravljenja liste kandidata rešenja koje su se pokazale dobro u praksi.

Dalje, iz tabele rezultata se primećuje da tabu pretraga nije dala bolje rešenje od lokalne pretrage za probleme dimenzija 127 i 202. Mogući razlozi za ovo su način implementacije tabu liste i/ili diverzifikacije. Umesto da tabu period bude fiksiran, bolja ideja bi bila da se menja dinamički tokom izvršavanja.

Postoje i razne naprednije strategije intenzifikacije i diverzifikacije. Jedna strategija intenzifikacije je, na primer, ona koja fiksira putanje koje su se pokazale kao dobre i modifikuje okolne putanje [1]. Još jedna dobra strategija intenzifikacije ali i diverzifikacije je prespajanje putanja (*eng. path relinking*) koja istražuje delove prostora pretrage koji vode od jednog elitnog rešenja do drugog [1].

Literatura

[1] Fred Glover, Manuel Laguna - *Tabu Search*, Springer Science+Business Media New York, 1997.

[2] Michel Gendreau, Jean-Yves Potvin - *Handbook of Metaheuristics (Third edition)*, Springer International Publishing AG, 2019.

[3] Michael Hahsler, Kurt Hornik - *TSP– Infrastructure for the Traveling Salesperson Problem*, https://epub.wu.ac.at/3990/1/TSP.pdf

[4] Shane N. Hall - *A Group Theoretic Tabu Search Approach to The Traveling Salesman Problem*, Air Force Institute of Technology, 2000. https://apps.dtic.mil/dtic/tr/fulltext/u2/a378321.pdf

[5] Johann Dréo, Alain Pétrowski, Patrick Siarry, Eric Taillard - *Metaheuristics for Hard Optimization: Methods and Case Studies*, Springer Berlin Heidelberg, 2006.

[6] Andries Engelbrecht - *Computational Intelligence - An Introduction*, John Willey & Sons, 2007.

[7] Sumanta Basu - *Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey*, American Journal of Operations Research, Vol. 2 No. 2, 2012, pp. 163-173. doi: 10.4236/ajor.2012.22019.

[8] Cesar Rego, Fred Glover - *Local Search and Metaheuristicsfor the Traveling Salesman Problem*, Hearin Center for Enterprise Science, School of Business Administration, Uni-versity of Mississippi, MS 38677 http://leeds-faculty.colorado.edu/glover/Publications/TSP.pdf

[9] Zoran Stanić - *Diskretne strukture 2*, Matematički fakultet - Beograd, 2018.

[10] TSPLIB95

[11] Florida State University

[12] Stack Overflow