Tabu pretraga

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Krivokuća Mina, Nićković Teodora, Popović Olivera, Zobenica Darinka
mina.krivokuca@gmail.com
nickovic97@gmail.com
popovic.olivera97@gmail.com
darinkazobenica@gmail.com

Sažetak

Problem pretrage je jedan od najznačajnijih i duboko izučavanih problema u teoriji računarstva. Algoritmi pretrage služe da dohvate element iz neke strukture podataka, odnosno unapred zadatog domena diskretnih ili kontinualnih vrednosti, tako da zadovoljava unapred zadate kriterijume. Efikasnost određenog algoritma pretrage zavisi od problema sa kojim se suočavamo. U daljem tekstu biće detaljno opisana tabu pretraga, metaheuristički metod koji je baziran na lokalnoj pretrazi. Koraci algoritma će biti razrađeni i približeni čitaocu kroz primere koda i nekoliko primena nad poznatim problemima kojima se bavi računarstvo.

Sadržaj

1 Uvod

Tabu pretraga je metaheuristika koja spada u algoritme lokalne pretrage, a koristi se za rešavanje problema kombinatorne optimizacije. [1] Razvio je Fred Glover u svom naučnom radu na temu celobrojnog programiranja 1986. godine. [2] Glover je imao cilj da izbegne zaglavljivanje pretrage u lokalnim minimumima, što je postigao konceptom tabua.

Reč tabu potiče iz tonganskog jezika, polinezijskog jezika koji se govori u Kraljevini Tonga. Izvorno reč označava svetinje, koje su u njihovoj kulturi bile zabranjene, nedodirljive, nedozvoljene za korišćenje. Prisvajanjem u engleski se izgubilo značenje svetinje, i reč označava samo zabranu. [3]

Tabu pretraga koristi koncept zabranjenih stanja da prevaziđe problem zaglavljivanja u lokalnim optimumima. Naime, ona što održava tzv. tabu listu, tj. listu stanja u kojima je pretraga već bila, te se u njih ne vraća ponovo. Uz to, tabu pretraga dozvoljava poteze koji pogoršavaju vrednost funkcije koju optimizujemo, pod uslovom da boljih dozvoljenih poteza nema. Ta dva koncepta zajedno dozvoljavaju joj da napusti lokalni optimum i nastavi potragu. [4]

Pošto tabu pretraga čuva zabranjena stanja, ne može se trivijalno primeniti na kontinualne probleme gde je korak u pretrazi takav da je mala verovatnoća da će pretraga uopšte i pokušati da se vrati u isto stanje. Zbog ovoga, mnogo češća primena je na diskretnim domenima. Međutim, kod kontinualnih problema se problem može diskretizovati, pustiti tabu pretragu da nađe vrednost blisku nekom optimumu, a zatim iz te tačke pustiti neki algoritam pogodan za kontinualnu pretragu, kao što je gradijentni spust. [4]

Tabu pretraga se nekad kombinuje sa rasejanom pretragom, čime se dobija hibridna metoda. Ova nova populaciona metoda pretrage se često koristi za rešavanje velikih nelinearnih problema. [5]

Cilj ovog rada je da čitaocu približi osnovne koncepte vezane za tabu pretragu na način koji je zanimljiv, edukativan, i precizan. Nakon definicije i objašnjenja osnovnih koncepata i njihove motivacije, izlažemo primere primene tabu pretrage na neke od poznatih NP problema, kao i zaključke iz drugih radova o kvalitetu primene ove metaheuristike u praksi.

2 Osnovni pojmovi i opis algoritma

Zadat je sledeći problem kombinatorne optimizacije, konkretno minimizacije funkcije f:

$$\min_{x \in X} f(x)$$

Funkcija f ovde može biti linearna ili nelinearna, a skup X predstavlja prostor dopustivih rešenja, i on mora biti konačan [6].

Svako $x \in X$ ima svoj skup $N(x) \subset X$, $x \notin N(x)$, i taj skup nazivamo **okolina** (eng. *neighborhood*) tačke x. $N(x)$ je definisan kao skup svih $y \in X$ koje se

mogu dobiti direktno na osnovu x modifikacijom koju nazivamo **pomeraj iz x u y** (eng. *move*) i obeležavamo $m(x, y)$. Rešenje iz okruženja obeležavaćemo sa x'. [6]

Jedno od mogućih rešenja ovog problema minimizacije je korišćenje pohlepne pretrage. [7] Takvo rešenje bilo bi jednostavno za implementaciju i često efikasno za izvršavanje. Međutim, ako se u svakom koraku bira najbolja raspoloživa akcija, takav algoritam će se često zaglavljivati u lokalnim minimumima, a u slučaju platoa se neće zaustavljati bez dodatnih ograničenja. Jedan od načina sprečavanja zaglavljivanja u lokalnom minimumu jeste uvođenje **tabu liste** (eng. *tabu list*).

2.1 Tabu lista i kratkoročna memorija

Tabu lista je lista **tabu čvorova** (stanja, konfiguracija, poteza...). Glavna ideja tabu liste je da čuva čvorove koji su posećeni, kako se tokom pretrage ne bismo vraćali na ono što je već istraženo [8]. Tabu čvorovi su oni čvorovi koji se ne mogu posetiti, čak i ako su u trenutnoj okolini $N(x)$. **Izmenjena okolina** (eng. *modified neighborhood*) $N^*(x)$ je rezultat ovoga, i ona čini skup dopustivih pomeraja. Izmenjena okolina $N^*(x)$ se u ovom slučaju formira kao $N(x) \setminus T$, gde T predstavlja tabu listu. Potom se iz nje bira sledeći korak x'. [4]

```
Ulaz: Konfiguracija pocetno_resenje
      Int max_iteracija
Izlaz: Konfiguracija najbolje_resenje

algoritam TabuPretraga:
    trenutno_resenje = pocetno_resenje
    najbolje_resenje = pocetno_resenje
    tabu_lista = [trenutno_resenje]
    za k = 1 do max_iteracija:
        ako je validno_resenje(trenutno_resenje) i
                f(trenutno_resenje) < f(najbolje_resenje):
            najbolje_resenje = trenutno_resenje;

        tabu_lista.dodaj(trenutno_resenje)
        okolina = generisi_okolinu(trenunto_resenje)
        izmenjena_okolina = [sve konfiguracije iz okoline
                             koje nisu u tabu listi]
        trenutno_resenje =
            selektuj_najbolje_ocenjeno_resenje(izmenjena_okolina);

    vrati najbolje_resenje
```

Prvi problem sa ovako koncipiranom tabu listom je to što je čuvanje svih obrađenih stanja jako skupo.[9] Rešenje koje se nameće je čuvanje samo određenih čvorova koji su nam relevantni, što može biti poslednjih k čvorova. Ovakvo rešenje ima smisla, jer je malo verovatno da će čvorovi koji su davno posećeni uopšte biti u trenutnoj okolini. Ovakva tabu lista naziva se **kratkoročna memorija**.[4]

Dužina liste u praksi može biti dinamički menjana, tako da s jedne strane spreči zaglavljivanje u lokalnom minimumu, a s druge ne uspori pretragu previše jer svaki potez moraju da se proveravaju svi elementi tabu liste. U nastavku ćemo smatrati da je dužina ove liste fiksna.

Drugi problem na koji nailazimo je da ovakvo čuvanje obrađenih stanja i dalje može biti skupo za neke probleme.[9] Razmatrana stanja mogu biti kompleksne konfiguracije koje su memorijski zahtevne za čuvanje, kao i za međusobno poređenje sa razmatranim stanjem u svakoj iteraciji. Zato se tabu lista često implementira tako da čuva neke apstrakcije skupa čvorova koji su skoro posećeni, na primer neke njihve osobine, ili pomeraje koji su nas doveli iz prethodnih stanja u tekuće. Ideja je da se takvi pomeraji ne prave opet, kako ne bismo ponovo došli u stanje sa lošijom vrednošću funkcije.

Ovakva implementacija tabu liste jasnija je na primerima primene. U rešavanju problema putujućeg trgovca, od početne konfiguracije [A, B, C, D, E], možemo dobiti novu tako što zamenimo gradove B i D, i dobijemo [A, D, C, B, E]. Ukoliko je ova konfiguracija lošija od početne, zamena gradova B i D će biti dodata u tabu listu.

2.2 Kriterijum aspiracije

U situaciji kada u tabu listi čuvamo pomeraje, a ne cela stanja, može se desiti da je za jednu konfiguraciju određen pomeraj loš, zbog čega on završava u tabu listi, ali da za drugu konfiguraciju on vodi u stanje sa boljom vrednošću funkcije. Drugim rečima, može se desiti da postoji pomeraj koji je tabu, ali u isto vreme i jako dobar. Zbog takvih situacija definišemo **kriterijum aspiracije** (eng. *aspiration criteria*). On modifikuje okolinu tako da izmenjena okolina sadrži i pomeraje koji su tabu, ali zadovoljavaju kriterijum aspiracije. Takav kriterijum se može definisati na mnogo načina, od čega je najjednostavniji - da li ovim pomerajem dobijamo konfiguraciju za koju je vrednost funkcije bolja od trenutne?

2.3 Dugoročna i srednjeročna memorija

Kratkoročna memorija je sama po sebi često dovoljna za postizanje rezultata koji su bolji od onih postignutih drugim metodama lokalne pretrage [10]. Međutim, često je potrebno pretragu dodatno usmeriti [1]. Usmeravanje se može izvršiti na dva načina:

- usmeravanje ka regijama koje su obećavajuće
- usmeravanje ka neistraženim delovima

U slučaju kada pretragu usmeravamo ka obećavajućim regijama, koristimo **pravila pojačavanja** (eng. *intensification rules*). Takva pravila mogu biti davanje prioriteta nekim karakteristikama dobre ciljne konfiguracije, na primer dobre vrednosti određenih promenljivih ili određenih segmenata konfiguracije. Struktura u kojoj se čuvaju takve karakteristike se naziva **srednjeročna memorija** (eng. *intermediate-term memory*).

Kada se pretraga usmerava ka neistraženim regijama, koriste se **pravila diversifikacije** (eng. *diversification rules*). Ova pravila se koriste u slučaju stagniranja pretrage, kada se ona mora resetovati. Jedan od načina za izvođenje ovoga je čuvanje konfiguracija koje su u ranijim iteracijama davale dobre rezultate, i u tom slučaju se pretraga vraća na neku od njih. Struktura u kojoj se čuvaju takve konfiguracije se naziva **dugoročna memorija** (eng. *long-term memory*). Važi da je $N^*(x) \subset N(x)$ kod strategija baziranih na kratkoročnoj memoriji, ali

ne nužno kod strategija baziranih na dugoročnoj. [4]

3 Primena

Tabu pretraga je postigla superiorne rezultate na širokom dijapazonu problema, koji je dodatno proširen kada se ona ukombinuje sa drugim algoritmima i tehnikama optimizacije. Uspešno rešava kombinatorne, kao i probleme optimizacije nad neprekidnim domenom [6]. U daljem tekstu ćemo prikazati primenu ovog algoritma na dva tipa problema:

- Kombinatorni problemi - **problem N-dama** (eng. *N-queens problem*)
- Celobrojno programiranje - **problem rasporeda objekata sa ograničenim resursima** (eng. *capacitated facility location problem*)

3.1 Primena tabu pretrage na problem N-dama

Treba rasporediti N dama na šahovskoj tabli dimenzije NxN, tako da se nikoje dve međusobno ne napadaju. Početna konfiguracija ovde može biti nasumičan raspored N dama, tako da je na svakoj vertikali tačno jedna. Pomeraj ćemo definisati kao pomeranje dame po svojoj vertikali, na poziciju različitu od trenutne. Cilj pomeraja će biti da se maksimalno smanji broj prekršaja koji postoje u toj konfiguraciji. Dakle, pomeraj je dodeljivanje vrednosti polja jednoj od dama, i jedan pomeraj ćemo zapisivati kao par (dama, polje). Broj prekršaja za jednu konfiguraciju je broj dama koje se međusobno napadaju. Okolinu za svaku damu čine svi njeni mogući pomeraji. U tabu listi čuvaćemo k prethodnih pomeraja, tj. k vrednosti (dama, polje), gde ovi parovi predstavljaju prethodne pozicije odgovarajućih dama. Time ćemo sprečiti da dama ponovo dođe na poziciju na kojoj je već bila.

```
Ulaz: Konfiguracija pocetna_konfiguracija
      Int max_iteracija
      Int max_duzina_tabu_list
Izlaz: Konfiguracija najbolja_konfiguracija

algoritam TabuPretraga:
    trenutna_konfiguracija = pocetna_konfiguracija
    najbolja_konfiguracija = pocetna_konfiguracija
    br_iteracija = 0
    tabu_lista = []

    dok god br_iteracija < max_iteracija i
            broj_prekrsaja(trenutna_konfiguracija) > 0 radi:
        okolina = generisi_okolinu(trenutna_konfiguracija)
        izmenjena_okolina = [sve konfiguracije iz okoline
                             koje nisu u tabu listi]
        ako je duzina(izmenjena_okolina) > 0:
            (dama, novo_polje) = selektuj_najbolji_potez(izmenjena_okolina)
            tabu_lista.dodaj((dama, staro_polje) iz trenutna_konfiguracija)
            zameni vrednost za dama u trenutna_konfiguracija

        ako je broj_prekrsaja(trenutna_konfiguracija)
               < broj_prekrsaja(najbolja_konfiguracija):
            najbolja_konfiguracija = trenutna_konfiguracija

        ako je duzina(tabu_lista) >= max_duzina_tabu_liste:
            izbaci_prvi_dodati_element(tabu_lista)

        brojac += 1
```

```
30
31      vrati najbolja_konfiguracija
```

Ovo je osnovni oblik algoritma tabu pretrage, primenjen na problem N-dama. Prikazana je samo kratkotočna memorija. U slučaju da želimo da dodamo kriterijum aspiracije, to možemo učiniti na sledeći način.

```
1       okolina = generisi_okolinu(trenutna_konfiguracija)
2       izmenjena_okolina = [potez iz okolina koji nije u tabu listi ili
3                       broj_prekrsaja(najbolja_konfiguracija) >
4                       broj_prekrsaja(trenutna_konfiguracija + potez)]
```

Moguće je dodati i dugoročnu memoriju, tako što pamtimo konfiguracije sa malo prekršaja, na koje se možemo vraćati nakon određenog broja iteracija ili male promene u poboljšanju, i time restartovati pretragu od tog trenutka. Implementacija ovog problema u programskom jeziku Python priložena je uz rad. U ovoj konkretnoj implementaciji, dovoljna diversifikacija je bila resetovanje pretrage i traženje optimalnog rešenja nad svim iteracijama. Parametri za broj resetovanja, broj iteracija svake pretrage i dužinu tabu liste određeni su ekperimentalno.

3.2 Primena tabu pretrage na CFLP

U svom najjednostavnijem obliku, **problem rasporeda objekata** (eng. *facility location problem - FLP*) je problem nalaženja tačke na ravni, takve da je suma rastojanja od ove tačke do N ciljnih tačaka najmanja (**Veberov problem**) [11]. Kompleksije varijante ovog problema bave se postavljanjem više objekata, zadovoljavanjem ograničenja oko njihovih lokacija (npr. radioaktivni objekti ne smeju biti blizu naseljenih mesta) i optimizacijom dodatnih kriterijuma (npr. blizina konkurentnih objekata).

Jedno od mogućih ograničenja može biti količina resursa u objektima za skladištenje koji treba da snabdevaju prodajne objekte. To je baš ono čine se bavi **problem rasporeda objekata sa ograničenim resursima** (eng. *capacitated facility location problem*). Formulacija problema je sledeća: pretpostavimo da imamo M mogućih lokacija za magacine i P prodavnica koje treba snabdevati robom. Za svaku od potencijalnih lokacija magacina poznat je njen kapacitet, kao i cena otvaranja. Za svaku od prodavnica, poznate su njene potrebe, kao i cena transporta robe od svakog od magacina (u rešavanju ovog problema, kao cenu smo uzeli Euklidsko rastojanje prodavnice i magacina). Potrebno je odrediti:

- koje magacine otvoriti,
- koja prodavnica će se snabdevati iz kog magacina.,

tako da robni zahtevi svih prodavnica budu ispunjeni, kao ukupna cena otvaranja magacina i transporta bude minimalna. Dakle, ciljna funkcija nam je ukupna cena, i nju treba minimizovati, uz dodatno zadovoljavanje ograničenja o resursima.

Magacini i prodavnice zovu se po indeksima. U listama su date odgovarajuće karakteristike lokacije, za lokaciju pod tim indeksom. Algoritam vraća najbolju dostignutu vrednosti ciljne funkcije (minimalni trošak), kao i najbolje rešenje, koje je dato listom istinitosnih vrednosti, koje označavaju da li se magacin sa tim indeksom otvara ili ne.

```
Ulaz: Int broj_magacina
      Lista<Int> kapaciteti_magacina
      Lista<Int> cene_otvaranja
      Int broj_prodavnica
      Lista<Int> robni_zahtevi
      Lista<Lista<Float>> cene_transporta
      Int max_iteracija
      Int max_duzina_tabu_liste
      Lista<Bool> pocetno_resenje
Izlaz: Float minimalni_trosak
       Lista<Bool> najbolje_resenje

algoritam TabuPretraga:
    trenutno_resenje = pocetno_resenje
    trenutni_trosak = odredi_vrednost_ciljne_funkcije(svi argumenti
    samog algoritma, sem max_iteracija i max_duzina tabu_liste)
    najbolje_resenje = trenutno_resenje
    minimalni_trosak = trenutni_trosak

    br_iteracija = 0
    tabu_lista = [trenutno_resenje]

    dok god br_iteracija < max_iteracija:
        okolina = [resenja dobijena invertovanjem tacno jedne istinitosne
                   vrednosti u trenutno_resenje]
        izmenjena_okolina = [svako resenje iz okolina koje ne pripada
                            tabu listi]

        novo_resenje, izmenjeni_magacin =
            odaberi_nasumicno_validno_resenje_iz_izmenjene_okoline()

        novi_trosak = izracunaj_vrednost_ciljne_funkcije(svi argumenti
            samog algoritma, sem max_iteracija i max_duzina tabu_liste,
            trenutno_resenje)
        tabu_lista.dodaj(novo_resenje)

        // Ukoliko nasumicna izmena pogorsava trosak, ignorisemo je
        // u potpunosti. Time je niz ocena koji se stvara uvek rastuci.
        ako novi_trosak < trenutni_trosak:
            trenutni_trosak = novi_trosak
            trenutno_resenje = novo_resenje

        ako trenutni_trosak < minimalni_trosak:
            minimalni_trosak = trenutni_trosak
            najbolje_resenje = trenutno_resenje

        ako duzina(tabu_lista) > max_duzina_tabu_liste:
           izbaci_prvi_dodati_element(tabu_lista)

    vrati minimalni_trosak, najbolje_resenje
```

Direktna implementacija ovog pseudokoda je uvek uspevala da poboljša početno rešenje, međutim često je bila jako neefikasna, i rešenja nisu mnogo napredovala. Diversifikacija nije mnogo pomogla, kao ni podešavanje parametara dužine tabu liste i broja iteracija. Međutim, uvođenje unutrašnje i spoljašnje petlje[CITATIONS] je značajno poboljšalo rezultate [UBACITI TABELU].

4 Poređenje tabu pretrage sa drugim algoritmima pretrage

Mnoge studije ispituju efikasnost i optimalnost različitih algoritama pretrage na različitim problemima. Kada je u pitanju tabu pretraga, neka istraživanja su pronašla da je ona za ispitivane probleme vidno bolja heuristika, [12] dok su druga zaključila da je kvalitet situacion i da zavisi od dimenzija i oblika problema. [13][14] Pri tome je nekoliko istraživača došlo do zaključka da tabu pretraga često daje nešto manje optimalna rešenja sa boljom efikasnošću, dok algoritmi kao što je simulirano kaljenje daju optimalnija rešenja, ali je vreme izvršavanja duže. [15] Postoje i istraživanja koja tvrde da je tabu pretraga manje efikasnija od drugih algoritama. [16]

Iz priloženog možemo zaključiti da je teško izmeriti efikasnost metaheuristika i da ona može biti jako situaciona, ali da se tabu pretraga definitivno pokazala uporedivom sa drugim kvalitetnim tehnikama lokalne pretrage, te da se sama ili u hibridnom obliku može primeniti na mnoge probleme sa jako dobrim rezultatima.

5 Zaključak

Ispod svakog naslova mora bar jedan paragraf, ne sme naslov pa podnaslov odmah

Literatura

[1] Glover, Fred: *Tabu Search: A Tutorial*. INFORMS Journal on Applied Analytics, 1990. https://www.researchgate.net/publication/242527226_Tabu_Search_A_Tutorial.

[2] Glover, F.: *Future Paths for Integer Programming and Links to Artificial Intelligence*. Computers and Operations Research, 1986.

[3] Dixon, Robert M. W.: *A Grammar of Boumaa Fijian*. University of Chicago Press, 1988.

[4] Fred Glover, Manuel Laguna: *Tabu Search*. Springer Science+Business Media, 1997.

[5] F. Glover, M. Laguna, R. Marti: *Fundamentals of Scatter Search and Path Relinking*. Control and Cybernetics, 2000.

[6] al., Mirjana M. Čangalović et: *Tabu Search: A Brief Survey and Some Real-Life Applications*. The Yugoslav Journal of Operations Research, 1996. http://elib.mi.sanu.ac.rs/files/journals/yjor/11/yujorn11p5-17.pdf.

[7] Predrag Janičić, Mladen Nikolić: *Veštačka inteligencija*. Matematički fakultet, 2019.

[8] Stefan Edelkamp, Stefan Schrödl: *Heuristic Search.* Morgan Kaufmann, 2012.

[9] Van Hentenryck, Pascal: *LS 9 - tabu search formalized, aspiration, car sequencing, n-queens.* https://www.coursera.org/lecture/discrete-optimization/ls-9-tabu-search-formalized-aspiration-car-sequencing-n-queens-CvYhL.

[10] Pandya, M. Malek; M. Huruswamy; H. Owens; M.: *Serial and parallel search techniques for the traveling salesman problem.* Annals of OR: Linkages with Artificial Intelligence, 1989.

[11] Michiel, Hazewinkel: *Encyclopedia of Mathematics.* Kluwer Academic Publishers, 2001, ISBN 978-1-55608-010-4.

[12] Abhay D Lidbe, Alexander M Hainen, Steven L Jones: *Comparative study of simulated annealing, tabu search, and the genetic algorithm for calibration of the microsimulation model.* SIMULATION, 93(1):21–33, 2017.

[13] Connor, A.M, Shea K.: *A COMPARISON OF SEMI-DETERMINISTIC AND STOCHASTIC SEARCH TECHNIQUES.* Springer, 2000.

[14] Arostegui, Marvin Jr. i Kadipasaoglu, Sukran N i Khumawala Basheer M.: *An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems.* International Journal of Production Economics, 103:742–754, 2006.

[15] E. Osaba, F. Díaz: *Comparison of a memetic algorithm and a tabu search algorithm for the traveling salesman problem.* U: *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, strane 131–136, 2012.

[16] Maninder Singh Kamboj, Jyotsna Sengupta: *Comparative Analysis of Simulated Annealing and Tabu Search Channel Allocation Algorithms.* International Journal of Computer Theory and Engineering, 1(5), 2009.

A Dodatak